2016

Positioning Commuters And Shoppers Through Sensing And Correlation

Rufeng Meng
University of South Carolina

Positioning Commuters and Shoppers through Sensing and Correlation

by

Rufeng Meng

Bachelor of Engineering
University of Shanghai for Science & Technology, 2000

Master of Science
University College London, 2009

Submitted in Partial Fulfillment of the Requirements

for the Degree of Doctor of Philosphy in

Computer Science and Engineering

College of Engineering and Computing

University of South Carolina

2016

Dedication

This dissertation is dedicated to my family, especially my parents, Zhigen Meng and Kune Shen. All I have accomplished are only possible due to their love and sacrifices.

Acknowledgments

Pursuing a PhD degree is a long and tough journery, without the support from my advisors, colleagues and my family, the journey would have ended in the middle.

I want to express my deepest gratitude to my major advisor, Dr. Srihari Nelakuditi. Without his support, I would not have the chance to work on mobile computing for my degree. Dr. Nelakuditi gave me unliminted freedom in doing my work and managing my time. I really appreciate his patience and tolerance. Many times, I felt depressed and on the edge of giving up, Dr. Nelakuditi found all the ways to cheer me up and encourage me to keep going. He shared lots of his knowledge, experience with me and tried everything he could on helping me on my journey towards the degree. To me, Dr. Nelakuditi is not only my advisor, but also a good friend. With whom, I could discuss everything open-mindedly. I will miss all the moments when we discussed and even argued on ideas in his office and the nights we spent in the lab on polishing papers.

I have been lucky enough to have an opportunity of working under the supervision of Dr. Romit Roy Choudhury in his SyNRG lab at University of Illinois at Urbana-Champaign, where I had a wonderful and productive year. I was really impressed by the way Dr. Roy Choudhury was thinking about research ideas, organizing teams, instructing students and managing time. Dr. Roy Choudhury gave me invaluable help on my research work. Wheneven I had problems on my work, he could always offer a thorough discussion with me and guide me to the correct direction. One year at SyNRG is very short, but it becomes the most precious experience for me.

I am thankful for having opportunities of working with Dr. Song Wang on my projects. Dr. Wang treated me exactly the same as the students in his own group. On the projects we collaborated, Dr. Wang gave me countless effective instructions. He cared about me so much and had lots of conversation with me on research and life. He also shared much his own experience to motivate me.

I'd like to thank my collaborators: Dr. Yan Tong, Sheng Shen, Sanjeev Singh, Chengfeng Mao and David Grömling. Without the fruitful discussion with them and their talent, hard-working, I could not have completed my work.

I sincerely appreciate Dr. Manton M. Matthews for serving on my committee, and reviewing my proposal and dissertation. I am much obliged to Dr. Guoan Wang for serving on my committee for dissertation proposal.

As part of my journey, I have spent lots of my time with the colleagues at ARENA@USC lab and SyNRG@UIUC lab. I heartfully appreciate having the experience to work and live with these talented and kind people. We had wonderful time together, which will be part of my memory forever.

ABSTRACT

Positioning is a basic and important need in many scenarios of human daily activities. With position information, multifarious services could be vitalized to benefit all kinds of users, from individuals to organizations. Through positioning, people are able to obtain not only geo-location but also time related information. By aggregating position information from individuals, organizations could derive statistical knowledge about group behaviors, such as traffic, business, event, etc.

Although enormous effort has been invested in positioning related academic and industrial work, there are still many holes to be filled. This dissertation proposes solutions to address the need of positioning in people's daily life from two aspects: transportation and shopping. All the solutions are smart-device-based (e.g. smartphone, smartwatch), which could potentially benefit most users considering the prevalence of smart devices.

In positioning relevant activities, the components and their movement information could be sensed by different entities from diverse perspectives. The mechanisms presented in this dissertation treat the information collected from one perspective as reference and match it against the data collected from other perspectives to acquire absolute or relative position, in spatial as well as temporal dimension.

For transportation, both driver and passenger oriented solutions are proposed. To help drivers improve safety and ease the tension from driving, two correlated systems, *OmniView* [1] and *DriverTalk* [2], are provided. These systems infer the relative positions of the vehicles moving together by matching the appearance images of the vehicles seen by each other, which help drivers maintain safe distance

from surrounding vehicles and also give them opportunities to precisely convey driving related messages to targeted peer drivers.

To improve bus-riding experience for passengers of public transit systems, a system named *RideSense* [3] is developed. This system correlates the sensor traces collected by both passengers' smart devices and reference devices in buses to position passengers' bus-riding, spatially and temporally. With this system, passengers could be billed without any explicit interaction with conventional ticketing facilities in bus system, which makes the transportation system more efficient.

For shopping activities, *AutoLabel* [4, 5] comes into play, which could position customers with regard to stores. AutoLabel constructs a mapping between WiFi vectors and semantic names of stores through correlating the text decorated inside stores with those on stores' websites. Later, through WiFi scanning and a lookup in the mapping, customers' smart devices could automatically recognize the semantic names of the stores they are in or nearby. Therefore, AutoLabel-enabled smart device serves as a bridge for the information flow between business owners and customers, which could benefit both sides.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Positioning is needed everywhere in our daily life. It helps people learn about their circumstances, anticipate and analyze the behaviors of individuals, groups and all kinds of systems. The requirement of positioning comes from both spatial and temporal dimensions. Spatially, position information could be GPS coordinates on a map, a bus station, a road segment a passenger traveled, a store in a shopping mall, etc. Temporally, position information tells the time an event happens. Position information could be absolute as well as relative. For example, an address on a street, a moment in a day; the relative distance and direction between two buildings, the relative time between two events within a period. Based upon position information, various services could be built to improve the quality of people's life as well as the efficiency of many systems in our society.

This dissertation tackles the positioning problems in two major areas related to people's daily life: transportation and shopping, which are the fields need positioning function deadly but have not been addressed well. In vehicular transportation, although there are many devices/solutions helping drivers locate their vehicles on the road, drivers are still in the hazard of traffic accidents and suffering from the stress induced by various traffic situations happening around. The vehicles/drivers are only positioning themselves independently, there are no massively affordable solutions for drivers to obtain the relative positions of the surrounding traffic. Existing public transit systems are not efficient due to the defects in their pricing and ticketing mechanisms. Besides, they also lack functionality for system operators to track passengers and get fine-grained information about the running of the systems. Regarding shopping activity, in current commercial society, when people are shopping, customers and businesses are basically separate entities. There is no effective way positioning one with regard to the other, which limits the information flow between customers and business owners.

Each of the activities in transportation and shopping is composed of certain components and movement, such as vehicles and their motion, stores, etc. The characteristics of the components and movement are always presented in different ways and could be captured by different entities involved in these activities. This dissertation exhibits how the sensing data describing the same components or movement in activities, but collected from different perspectives by different entities, could be correlated to provide position information, thereby benefits people in the corresponding activities.

Nowadays more than 79% of mobile subscribers in the US are using smartphones [6]. People carry smartphones everywhere and use them in almost every activity. In transportation, smartphones are used for navigation, entertainment and other purposes; during shopping, smartphones are used for price comparison, payment, etc. Along with the advocacy of smartphone usage in automobile and commerce from giant companies, such as GM, Ford [7] and Apple [8] [9], we believe smartphone will be used more and more in people's daily life, especially transportation and shopping. The solutions presented in this dissertation are following this trend, which could benefit both the people and the organizations involved in the corresponding activities.

The work described hereinafter is categorized into three parts. The first two parts are for vehicular transportation. Among them, two correlated solutions are designed for improving safety and experience of driving, another solution is targeted at aiding passengers of public transit systems. The third part describes a shopping oriented solution.

1.1 Improving Experience of Driving

Drivers need to receive and process lots of information when they are moving on the road. Driving related information, including but not limited to road condition, traffic situation, traffic signs, the readings of the meters on dashboard, all kinds of 3rd-party devices used in vehicles, signals given by neighboring vehicles, is flowing to drivers continuously. Among the data stream of driving, positioning related information plays the most important role, which is closely related to the safety and experience of driving. Drivers need to maintain certain distance from their neighboring vehicles by continuously estimating the relative positions of each other. Drivers are experiencing nervousness stemming from their own maneuver and the stress from surrounding vehicles/drivers.

To ease the tension behind the steering wheel, we design two correlated systems, OmniView and DriverTalk, to help drivers learn about the relative positions of surrounding vehicles and allow neighboring drivers to directly communicate to precisely convey driving related message. OmniView and DriverTalk are both smartphone-based and vitalized through the collaboration among the vehicles moving together. The collaboration is realized based on the common information detected and exchanged by the smartphones of different participants. In these systems, the common information is the appearance image of the vehicles seen by each other. A vehicle x knows how itself looks like. Meanwhile, other participants moving around also see its appearance. When a participant y takes a picture for x and sends it the picture, vehicle x could infer the relative position of y by matching the information describing the same object, i.e. the images about the appearance of vehicle x, but collected by two different entities, i.e. vehicle x and y. In this way, the adjacent vehicles could learn about the relative positions of each other, which makes it possible to provide a map about the distance/direction of the neighboring counterparts for drivers to improve safety. This mechanism also enables targeted communication among neighboring drivers to convey necessary message, such as intent of maneuver, which could improve safety as well as ease the tension during driving.

1.2 Improving Experience of Bus-riding

Existing public bus system has certain defects. For example, the ticketing systems are not efficient and smart enough. Passengers still need to interact with staff or certain appliances on the buses to pay their fare. The system itself could at most track the boarding of a passenger, but has no knowledge about where a passenger gets off. Therefore, there is no way for the operators to learn about the fine-grained utilization of their systems, such as the occupancy of a bus at a moment.

To improve the experience of bus-riding and the efficiency of public transit system, a system, RideSense, is developed. It allows passengers to take bus without any explicit interaction with ticketing related facilities in bus. A passenger simply gets on and off, and her bus-riding information, including motion and the environment experienced during the trip, is collected by two different entities: the passenger's smart device (such as smartphone, smartwatch) and a reference device in bus. By correlating the two sources of data for the same activity, i.e. bus-riding, of a passenger, RideSense could derive fine-grained position information about the passenger's bus trip, such as the corresponding bus line, shift, source and destination stations. Thereby, it could help to improve the bus-riding experience and also allow authorities to understand the running of their bus systems better.

1.3 Improving Experience of Shopping

When people go shopping, if there is a system which could automatically position customers with regard to the stores where the customers are, many services could be built based upon this kind of positioning, which could potentially benefit both customers and business owners. For example, business owners could advertise their products accurately to the customers currently inside or near their stores; customers could get product information and coupon of the current stores. This kind of positioning mechanism needs to go beyond conventional map-coordinate mode, it should locate customers with stores semantically.

Although numerous researchers engaged in localization area, semantic positioning is not addressed well yet. We proposed our semantic positioning system, AutoLabel, which enables customers' smartphones to recognize the stores they are in or nearby automatically and semantically. Therefore, it could serve as a bridge between customers and stores to flow the information benefiting each other.

AutoLabel constructs a map which contains the relationship between stores' WiFi vectors and semantic names. The system connects WiFi vectors with semantic store names through correlating the text decorated inside stores with the text shown on stores' webpages. Both sources of text describes the property of the stores. The text decorated inside stores is easily connected with the corresponding WiFi vectors through smartphones' sensors, such as camera and WiFi module. The text on a store's webpages could be linked with the store's semantic name through search engine. By matching the two sources of text, the connection between WiFi vectors and semantic store names could be established. Later, during shopping, customers' smartphones could detect stores' WiFi vectors in-situ and consult the WiFi–Store name mapping to retrieve the semantic names of the stores they are in or nearby. With the positioning service provided by AutoLabel, customers could have better shopping experience and business owners are able to run their business more effectively.

1.4 Contributions

Through the work depicted in this dissertation, the following contributions are presented:

- We designed OmniView to help drivers maintain safe distance from their surrounding traffic through a mechanism of obtaining relative positions of neighboring vehicles based on matching vehicles' appearance images.
- We proposed DriverTalk to give drivers the opportunities to precisely convey driving related messages to their neighboring peers, which not only helps drivers improve safety but also ease their tension during driving.
- We developed RideSense for positioning passengers' bus trips by matching motional and environmental sensor traces with regard to the passengers' bus-riding activities, which could improve the efficiency of public transit system and provide a better insight into its running.
- We provided a semantic positioning mechanism to locate customers with regard to stores through mapping stores' WiFi vectors to store names based on the correlation between stores' in-store text and web text. A so designed AutoLabel system verified the feasibility of the mechanism and shows that this positioning mechanism could benefit both customers and business owners.

Part I

Improving Experience of Driving

When people are driving, they are continuously facing potential hazard and experiencing the tension from driving and peer vehicles moving around. Many solutions have been proposed to improve safety of driving and provide better driving experience, but most existing systems work in a standalone mode, which makes them hard to achieve the goal in many scenarios.

To improve safety and experience of driving, we designed two correlated systems. Both systems position the neighboring vehicles/drivers for each participant, which are implemented on smartphones and working in a collaborative way among the participants.

The first system is OmniView, which provides the drivers with a map about the surrounding traffic. The map tells the relative distance and direction between neighboring vehicles and alerts drivers to keep safe distance when it detects potential hazards. The second system is DriverTalk, which provides drivers with a way of talking to each other. Drivers could use this system to precisely convey information related to their driving to the peers in the surrounding vehicles.

This part elaborates the motivation, design, performance and also limitation, related work of these two systems.

Chapter 2

OmniView: A Collaborative System for Assisting Drivers with a Map of Surround Traffic

Every year, more than 32000 people died and more injured in traffic accidents in the US [10]. Traffic accidents also caused tens of billions of dollars' property damage. Most of those traffic accidents could be categorized into three types: forward collision, in which a vehicle hits another vehicle moving in front; rear-end collision, where one vehicle is hit by another vehicle moving behind; lane departure collision, which happens when a driver changes lane and hits another vehicle moving in the next lane. A common cause behind these collisions is that drivers fail to notice the presence of surrounding vehicles and/or maintain a safe distance from them.

Government and auto makers have invested enormous fund to make the vehicles and traffic system safe. As to driving safety, rules and facilities are only one side; another important factor is driver. Nowadays, we still need drivers to operate the vehicles. Drivers need to pay attention to their surroundings, to continuously estimate the traffic around and take proper maneuver to avoid involving in any dangerous situation. Drivers need to not only watch out for the vehicles moving in front of them, but also be aware of the vehicles in their next lanes and behind them. This is an onerous task. Some drivers are good at estimating distance, others are not. Drivers are easily looking at one side but neglecting other sides, even sophisticated drivers still make mistakes.

Therefore, it will be beneficial to drivers if the vehicles have a map telling the relative positions of the traffic around them. Such a traffic map, even without being made visible to the drivers, can help trigger acoustic alerts to prevent collisions. Indeed, some advanced driver assistant systems are being actively developed [11]. But, they tend to be pricey and available only in the latest models of middle/high-class vehicles. Furthermore, many of these assistant systems only monitor one side for the drivers. Our goal is to bring similar safety features to drivers of legacy and/or economy vehicles. Towards that end, we propose a system called OmniView that extends the vision of drivers in all directions using cameras of multiple collaborating smartphones.

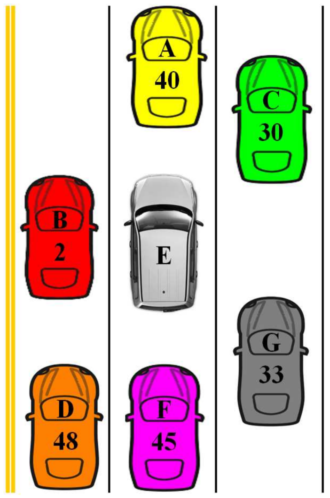


Figure 2.1: OmniView provides driver with a top-view map showing the relative positions of the neighboring vehicles. The numbers on the vehicles indicate the distances between the ego-vehicle (i.e. the vehicle in the center) and its neighbors.

OmniView is a smartphone-based system which provides driver with a real-time top-view map (as shown in Fig.2.1). The OmniView map shows the positions of all neighboring vehicles moving in the same direction as the ego-vehicle[1]. It contains distance information between the ego-vehicle and the surrounding vehicles, and it also tells relative direction of each neighboring vehicles (e.g. which lane a neigh-

[1]In a context which involves several vehicles, the one which is chosen as the subject in narration is called *ego-vehicle*.

boring vehicle is in). When it detects any potential hazard, e.g. the ego-vehicle is too close to the vehicle in front, it will alert driver to keep safe.

An obvious way to obtain such a map is to use GPS and have each vehicle report its position. Then, each vehicle can determine its distance to others. Unfortunately, GPS error for civilian usage could be up to 30 meters [12], which is too large to distinguish lane-level position. Although Differential GPS [13] could help reduce the error, it requires deploying reference stations all along the road. Furthermore, GPS based approach requires every vehicle's participation. If a vehicle does not report its GPS position, other vehicles would not even be aware of its presence. Therefore, we propose to utilize the cameras of smartphones and computer vision techniques to gather the traffic map, without requiring full participation from all surrounding vehicles.

Nowadays, more than 79% mobile subscribers are using smartphone in the US [6]. People carry and use smartphones everywhere in their daily life. Most smartphones have two cameras (one rear, one front) and many other sensors. Smartphones also have wireless modules (as shown in Fig.2.2 (left)) which enable them to communicate with each other.

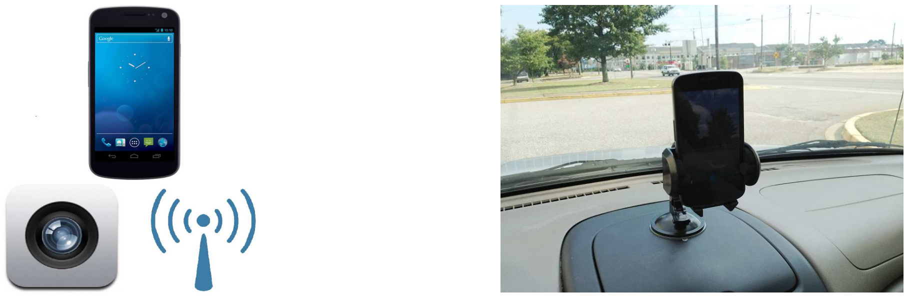


Figure 2.2: (Left) Smartphone has camera and wireless component. (Right) It is mounted on dashboard or windshield in vehicle when using OmniView.

OmniView system utilizes the capability of smartphone. The participant mounts her smartphone on the dashboard or windshield (as shown in Fig.2.2 (right)) with its rear camera facing forward. In this way, the smartphone can 'see' the vehicles (could be on the same lane as the ego-vehicle or on the left/right adjacent lanes) moving in front. These OmniView-installed smartphones also communicate with each other to share what they have seen. Every vehicle fuses its own vision with the vision received from others to form an OmniView map about the neighboring vehicles.

OmniView uses computer vision to detect vehicles and exchanges detected vehicles' appearance images over wireless network. After processing the images and calculating the positions of vehicles, it forms a map about surrounding traffic, which could show on the screen or run in background. OmniView does not require any road-side facilities, what it uses are only the ubiquitous smartphones. Users only need to install the OmniView app into their smartphones to gain the benefit of this driving assistant system without any extra operations.

To ensure a real-time traffic map, OmniView needs to process and communicate with each other as efficiently as possible. As we know, computer vision related work usually requires powerful machines to do intensive computation, but here OmniView does everything on smartphones. Recent years, smartphone has advanced a lot, but compared with PC, its processor is still slow and the memory is not large enough, which raises big challenges to our system.

In this work, we need to study how efficiently OmniView could do calculation and communication to detect vehicles and form a map; we also need to find out how reliably the OmniView-enabled vehicles could collaborate with each other. Our evaluation shows that OmniView could provide the surrounding-traffic map with reasonably high reliability in real-time.

2.1 System Overview

OmniView is a vision-based vehicular collaboration system. With OmniView, every vehicle detects the vehicles moving in front of it and calculates the relative positions (direction and distance) of the detected vehicles to form a local map about those vehicles. The participants also communicate with each other to share local maps. By fusing local map and the maps received from others, each participant could obtain a more complete view about her surrounding traffic. OmniView uses vehicle's appearance image as ID (herefrom, termed as *visual ID* or *VID*) to represent each vehicle. It uses VIDs to identify vehicles, calculate relative positions, communicate and fuse maps.

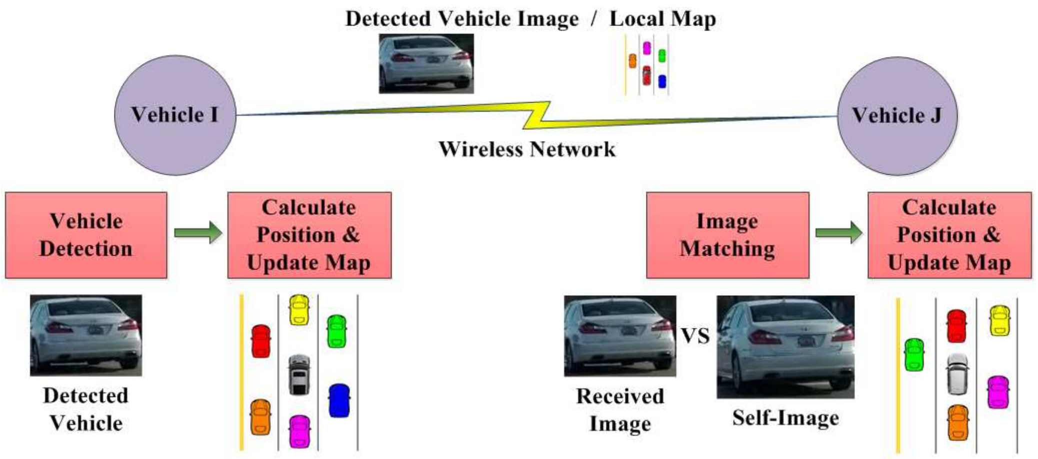


Figure 2.3: OmniView system workflow: Vehicle I detects vehicle J. It calculates the position of vehicle J and updates its local map to show vehicle J. Upon receiving the detected vehicle image from vehicle I, vehicle J estimates the relative position of vehicle I and shows vehicle I on its local map. The participants also regularly exchange their local maps to help each other form more complete and accurate map about neighboring vehicles.

Before using OmniView, every participant driver takes pictures for her own vehicle in different directions (left-side, rear-end, right-side) from different distances, as exemplified in Fig. 2.4. These pictures are stored locally in the smartphone and referred as *self image*. To use OmniView, each driver mounts her OmniView-enabled smartphones on dashboard or windshield with its rear-camera facing forward (as shown in Fig. 2.2 (right)). The sketch of OmniView is shown in Fig. 2.3.

Being a safety related system, OmniView must act as efficiently as possible. Every step in the work flow must complete in real-time. OmniView uses computer

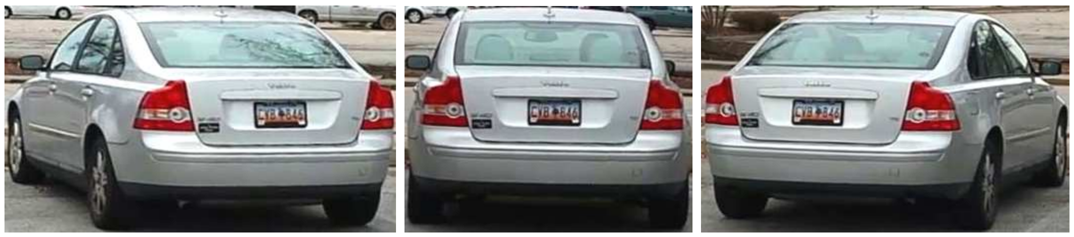


Figure 2.4: Each driver takes pictures for her own vehicle from 3 different directions: left-back (left), rear-end (middle) and right-back (right). These images are stored locally as self images in her DriverTalk-enabled smartphone.

vision to detect vehicle, match images and calculate relative positions of vehicles. It also relies on smartphone's wireless component to communicate with each other. But the computation of computer vision algorithms/technologies is usually time-consuming, especially for smartphone's CPU and memory; wireless communication between vehicles is not as reliable as wired network. Therefore OmniView needs to address the following challenges:

a) Whether could OmniView detect vehicle, communicate and do image matching efficiently enough to allow real-time performance?

b) Whether could the participants confidently conclude that the received image is someone detected for it, not for others?

c) How accurately can each vehicle calculate the positions of other vehicles?

2.2 Design & Implementation

As illustrated in Section 2.1, the whole OmniView system could be divided into five functional parts: vehicle detection, vehicular communication, visual ID recognition, position calculation and map construction. Here below we clarify the design and implementation of each part.

A. Vehicle Detection

When a smartphone is mounted with its camera facing forward, it could see the whole scene in front of the ego-vehicle. In OmniView, we only care about vehicles on the road. So the first thing OmniView needs to do is to detect vehicles in its camera view.

For OmniView to detect vehicles, we need to construct a vehicle detection model based on vehicles' appearance/shape, which enables OmniView to identify whether the objects in the scene are vehicles or not. We trained a Haar Cascade classifier [14,15] on vehicle images collected from [16–19] and frames from self-taken videos on highways near Columbia and Charleston in South Carolina. The whole collection contains 1600+ positive and 4300+ negative samples about vehicles' rear-end, left-side and right-side appearance/shape.

OmniView's vehicle detection works as shown in Fig. 2.5. Once OmniView detects a vehicle, it extracts the part of the image containing the detected vehicle (the segment in the green rectangle). The extracted image (which is in JPEG-format) becomes the VID for the vehicle being detected, which will be used in position calculation, map construction and communication.

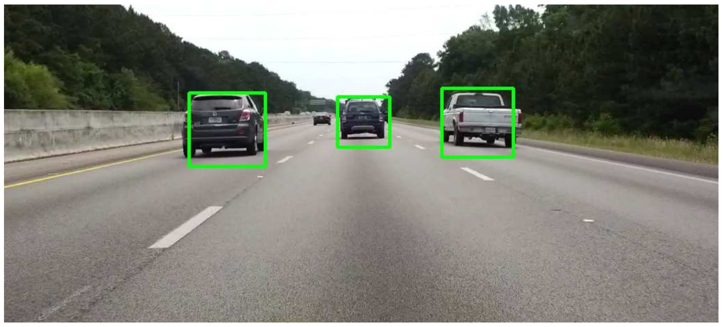


Figure 2.5: OmniView detects vehicles moving in front of the ego-vehicle and extracts the images about the vehicles being detected.

B. Vehicular Communication

In OmniView, vehicles collaborate with each other to exchange detected vehicle images and maps over wireless network. But the size of image might be too large for wireless communication. We need to choose proper communication technology and strategy, decide reasonable size for the data to be exchanged.

Communication Technology and Strategy

For vehicular communication, we adopt DSRC 802.11p [20], which is a wireless protocol drafted since 2005 and designed particularly for vehicular network. DSRC 802.11p supports transmission range up to 1000 meters based on the power setting, and it provides data rates from 3 to 27 Mbps. In OmniView, we choose 6 Mbps as the data rate, which is verified to be optimal for communication in vehicular network [21].

Current smartphone does not have standard DSRC module yet. But after-market DSRC component for smartphone is available now. Besides, IC manufacturers, such as QualComm [22], have already been developing and demonstrated DSRC-enabled reference design phones [23]. We can expect that DSRC will be a standard component in smartphone in the very near future.

On the road, vehicles come and go randomly. When two vehicles meet, they usually don't have any knowledge about each other in advance. They don't know each other's conventional address (such as IP or MAC address). Therefore, OmniView uses broadcast to communicate, which allows all neighboring vehicles to hear the transmitted vehicle images and maps.

When transmitting detected vehicle image, the sender only wants the peer being detected to recognize this message. For example, vehicle A moves behind vehicle B. When vehicle A detects vehicle B, it transmits the detected image of vehicle B to help vehicle B be aware of the presence of vehicle A. In such kind of scenario, vehicle

A only wants vehicle B to recognize what it sends. We need a mechanism to allow the participants to correctly identify the sender or receiver of a message. License number shown on the license plate might be an option for identifying the vehicles, but a vehicle's license number is only legible within a very limited distance and angle for smartphone's camera. OmniView uses images about vehicles' appearance as IDs, i.e. VIDs, to identify the vehicles in communication. Once the vehicle being detected recognizes the VID in the received message, it can conclude that some other vehicle is communicating with it (Section 2.2.C will explain the method for recognizing VID.).

Controlling Communication Overhead

OmniView needs to exchange detected vehicle images, which could be up to tens of KBs. If all the participants are frequently broadcasting vehicle images, we can expect that the network will be full of collisions, most of the messages will be lost and the performance of OmniView will be very poor.

By observing the traffic on highway, we can easily discover that although the vehicles are moving very fast, the relative speeds between vehicles is low, which are below 30 mph most of the time. When two vehicles meet, instead of coming and going ephemerally, they usually stay in the communication range of each other from a couple of seconds to several minutes, which is called the contact time. During the contact time, the vehicles do not need to communicate with each other by the detected image every time. Once one vehicle knows the other vehicle's VID, it does not need to frequently transmit the detected image.

In exchanging local map, if OmniView uses VID to represent each vehicle in the map, the map size will be too large for transmission because VID is still the vehicle's image. To keep every participant's local map up-to-date, the vehicles need to exchange maps efficiently. OmniView maps VID to text ID (from here on, we use

"TID" for "text ID") and then uses text ID to represent each vehicle in the map, which could significantly reduce the size of exchanged map message. We choose vehicle's license number as TID (e.g. SC77CD88), which is automatically unique and known to its driver. Each participant maintains the VID-TID mapping for the vehicles nearby (as shown in Fig. 2.6) and uses the TID to represent each vehicle in the map. In this way, although OmniView still needs to periodically transmit detected vehicle images to assure the correctness of the corresponding VID-TID mapping, the frequency of transmitting vehicle images could be significantly lowered. The VID-TID mapping is constructed through message exchange, which is explained next.

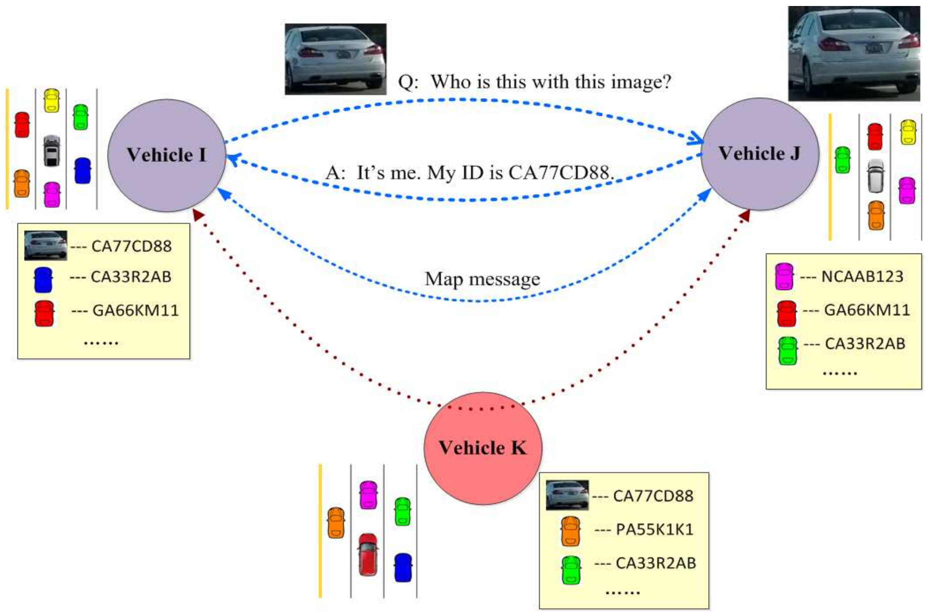


Figure 2.6: Message exchange in OmniView: Vehicle I detects J, sends an Identify message to J, and J responds with its TID. I and J then exchange Map messages. A vehicle K moving nearby could overhear these messages and update its local map. Each vehicle caches VID-TID mapping for the vehicles nearby.

Message Exchange

There are two types of messages being exchanged in OmniView system. One is Identify message, which is used to resolve the TID of a vehicle from an image (i.e.

VID); the other is Map message. Fig. 2.6 shows the message exchange. A vehicle, say I, detects vehicle J for the first time, it sends an Identify message (the question in the figure) with the VID detected for J, when it tries to resolve the TID of J. Vehicle J responds with its TID, along with its own local map. Then, vehicle I records the mapping between J's VID and TID, and also merges J's map with its own map. In case a vehicle does not receive a response to its query, after a few, say two, attempts, it assumes that the target vehicle is a non-participant and associates a random TID to that image, to minimize futile queries.

When Identify messages are being exchanged between vehicles I and J, other surrounding vehicles, such as K, that overhear these messages, will also record VID-TID mapping for J and update their maps.

Each participant also periodically broadcasts its local map. Every other vehicle which hears the map broadcast by others, compares the TIDs in the received map and its local map. If common TIDs exist, it merges the two maps to obtain a more complete and accurate map.

C. Visual ID Recognition

When one vehicle detects another vehicle moving in front of it, it compares the detected vehicle image with the images stored in the maintained VID-TID mapping. If exists, it directly uses the corresponding TID; otherwise, it sends Identify message to ask for the TID. When one vehicle receives an Identify message from others, it needs to decide whether the embedded VID is for it (i.e. whether some other vehicle has detected it and is asking for the TID). It does this by comparing the received image with its self image. If it recognizes itself in the received image, it answers to announce its TID.

In OmniView, all the visual ID recognition is realized by image matching. The flow is shown in Fig. 2.7. After the matching process, if there are still enough

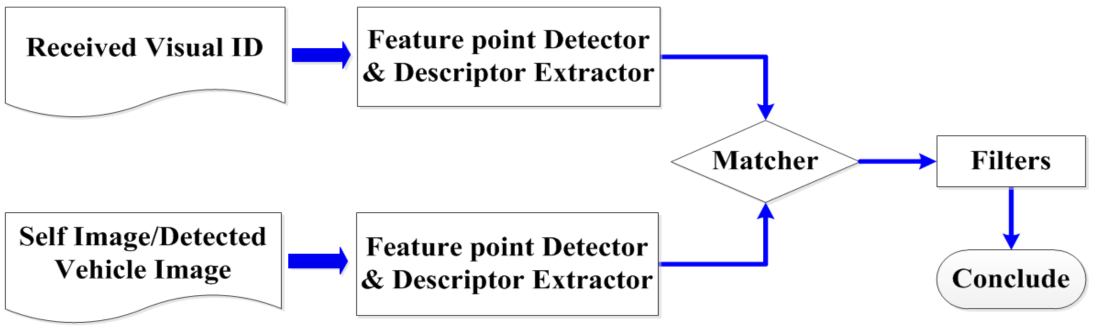

Received Visual ID
Feature point Detector & Descriptor Extractor
Matcher
Filters
Self Image/Detected Vehicle Image
Feature point Detector & Descriptor Extractor
Conclude

Figure 2.7: Flow of visual ID recognition: The feature points and the corresponding descriptors are extracted from the images being compared, which will flow through the matcher to find out the matched points. A set of filters are used to remove the false matching before making conclusion.

number of matched points left, OmniView concludes that these two images matched (i.e. the vehicles in the two images are the same). Fig. 2.8 exemplifies the image matching in different scenarios.

Table 2.1: Vehicle Images for Image Matching Test

Image	Size (KB)	2D Size
A	25.8	333×270
B	16.9	238×194
C	11.4	185×149
D	6.7	128×102

Table 2.2: Time of Image Matching with Different Feature Point Detectors and Descriptor Extractors

Image	SIFT (Unit: S)	SURF (Unit: S)	SURF/BRISK (Unit: S)	SURF/FREAK (Unit: S)	ORB (Unit: S)
A vs. A	26.3	18.3	9.9	3.8	0.637
A vs. B	19.9	12.6	9.0	3.0	0.497
A vs. C	16.9	9.3	8.6	3.0	0.371
A vs. D	14.7	7.1	8.6	2.4	0.256
B vs. B	13.8	7.9	8.3	2.3	0.374
B vs. C	11.0	5.5	7.9	2.3	0.262
B vs. D	8.8	3.9	7.9	1.7	0.494
C vs. C	8.4	3.3	7.5	1.6	0.173
C vs. D	6.2	2.2	7.5	1.4	0.429
D vs. D	4,1	1.2	7.0	1.2	0.055

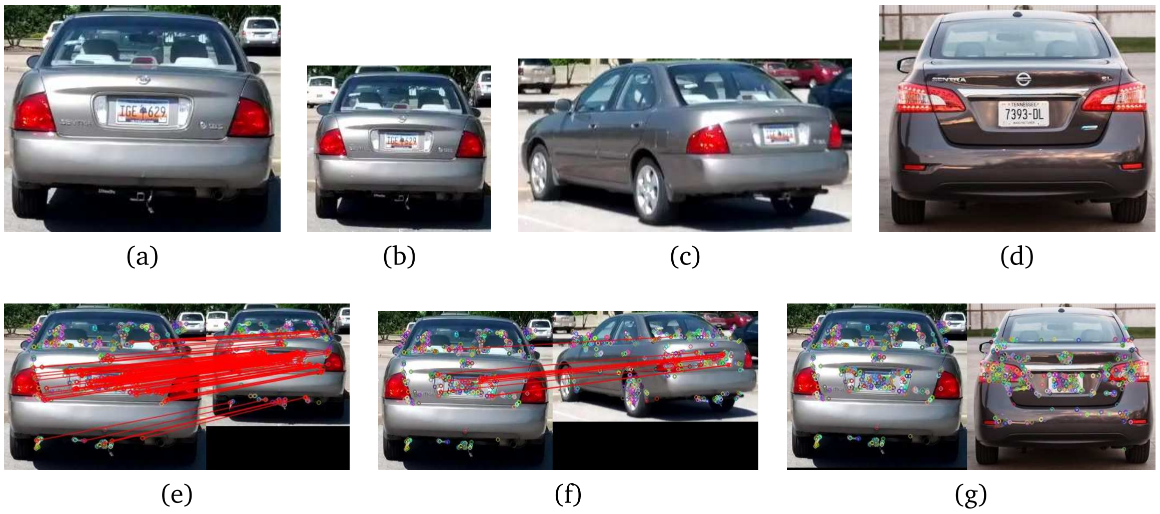


Figure 2.8: Image matching between images of same vehicle and different vehicles. Image (a) (b) (c) are for the same vehicle. The first two (a, b) are detected in the same direction (i.e. rear-end), while (c) is detected in a different direction (left-back). The vehicle in image (d) is different. We use (a) as a reference and (b)(c)(d) match with this reference. The bottom row shows the result of image matching between these images. (e) shows that if the two images are for the same vehicle and in the same direction, they could have lots of matched points. (f) shows that if the two images are for the same vehicle but in different directions, image matching could get some matched points, but fewer than the same-direction matching. (g) shows that if the two images are for different vehicles, although they are in the same direction, they have very few, even zero matched points.

There are many algorithms for feature point detector and descriptor extractor, such as SIFT/SIFT [24], SURF/SURF [25], SURF/BRISK [26], SURF/FREAK [27], ORB/ORB [28], etc. OmniView is a safety related system, which needs to run on smartphone, so the image matching task should perform as quickly as possible on smartphone. We did a test with the above options for feature point detector and descriptor extractor on four vehicle images as shown in Table 2.1. This test was carried out on a Galaxy Nexus (Android 4.2.1, CPU: 1.2GHz Dual-core; Memory: 1GB) and the result (in Table 2.2) shows that, ORB could detect feature points and extract descriptors within 1 second, while SIFT/SIFT, SURF/SURF, SURF/BRISK and SURF/FREAK are too inefficient for a real-time system. Therefore, ORB is chosen for visual ID recognition in OmniView.

As an image about a vehicle, on one hand, the larger the image is, the more details it can catch about this vehicle, thereby more feature points could be extracted. As a result, it will lead to a more confident conclusion from image matching. On the other hand, we need to transmit image over wireless network. Large image will not only occupy more channel time and incur more collisions, but also require more time to match, which could decrease the overall efficiency of OmniView. Therefore a proper image size which is friendly for transmission and meanwhile guarantees high confidence in making conclusion from image matching is very important.

To study the trade-off between image size and matching confidence, we collect two sets of vehicle images. The first set contains 530 pairs of images and each pair contains two images for the same vehicle. The second set contains 1090 pairs and the images in each pair are for different vehicles. The results of image matching for these two sets are shown in Fig. 2.9. It is evident that if two images are for different vehicles, the number of matched feature points is rarely above 10. On the other hand, if two images are for the same vehicle and the image size is larger than 10 KB, more than 20 features points match. Therefore, by choosing a threshold like 15 for the number of matched points and image sizes 10$\sim$14 KB, OmniView could identify a vehicle with high accuracy through image matching. When OmniView detects a vehicle, it scales down the detected vehicle image to this range, to reduce communication overhead.

D. Position Calculation

The relative position of a vehicle could be specified with <lane, distance>. Below, we describe how one vehicle determines another vehicle's lane and estimates distance to it.

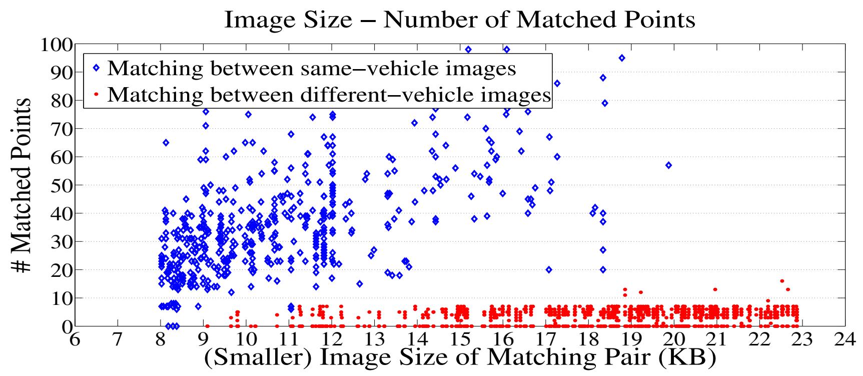


Figure 2.9: Image matching between pairs of same-vehicle and different-vehicle images (x-axis shows the size of the smaller image in each pair).

Lane Determination

The OmniView system needs to determine the lane-level position of vehicles, i.e. who is in front and who is on the left adjacent lane, etc. We adopt two strategies to get the lane position information.

When OmniView is doing vehicle detection, it extracts lane markings (as in Fig. 2.10) at the same time, which could be used to determine the relative lane positions of ego-vehicle and the vehicles being detected. This lane position information is included in Identify message, which not only helps the receiver get the sender's position, but also helps a receiving vehicle filter Identify messages not meant for it without performing image matching, which saves significant computational overhead.

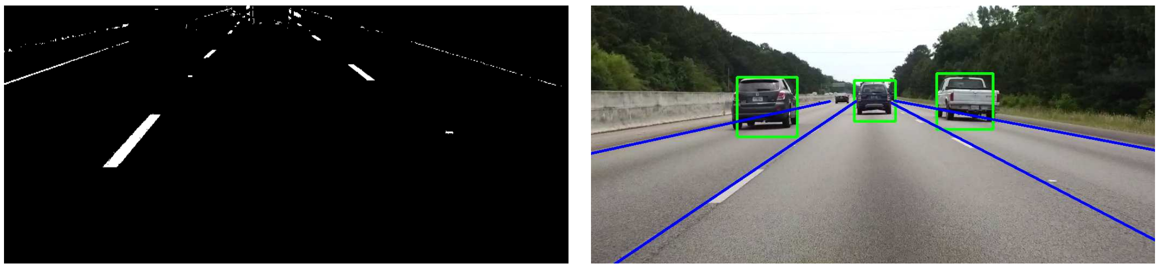


Figure 2.10: (Left) Extracted lane markings on the road. (Right) From the lane markings, OmniView identifies the relative positions between the ego-vehicle and the detected vehicles in its field of view.

The second strategy for determining relative positions of vehicles is based on image matching. When one vehicle receives an image, it matches that image with its self images in different directions. The best match tells which side the sender is. Table 2.3 shows the number of matched feature points between two sets of images taken at different distances and from two directions (left-side and rear-end) by two different phones. Clearly, same-direction images have many more matched points than different-direction images.

Table 2.3: Matching Images from Different Directions

Distance(m)	Rear vs. Rear	Rear vs. Left	Left vs. Left
10	386	75	542
20	191	50	256

When lane markings are not identifiable, and the exact lane positions of vehicles can not be determined, OmniView uses image matching based method as a fallback option for estimating the relative directions.

Distance Estimation

We compute the distance between vehicles in two ways: when a vehicle detects another, it estimates the distance to the vehicle being detected; when a vehicle receives an Identify message and itself is the target, it estimates the distance to the sender.

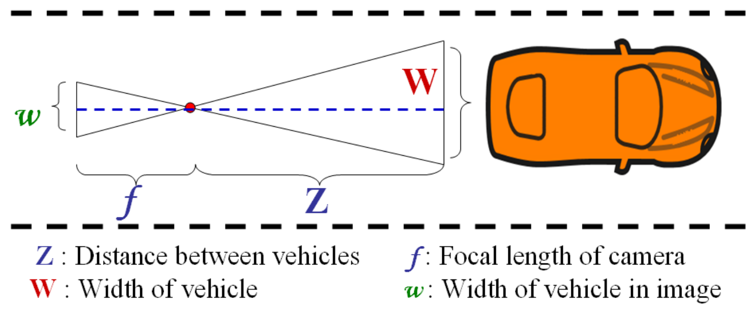

w
W
f
Z
Z : Distance between vehicles
f: Focal length of camera
W : Width of vehicle
w: Width of vehicle in image

Figure 2.11: Relationship between vehicle and its image in camera.

When a vehicle detects another vehicle, the distance (Z) from the ego-vehicle to the detected vehicle, the focal length (f), the width of the vehicle in the camera view (w) and the actual width of the vehicle (W) satisfy the relationship (see Fig. 2.11):

$$\frac{W}{Z} = \frac{w}{f} \tag{2.1}$$

Here, focal length f of the smartphone's camera could easily be obtained through SDK, whereas w depends on the pixel size of the camera sensor, which is hard to obtain. We overcome this problem by having the driver take an image of her own vehicle, with width W_0, from a known distance Z_0, for calibrating the camera. From this self-image, we have Eq. (2.2). When using the system, once it detects other vehicle in front, we have Eq. (2.3). In (2.3), W_{new} is width of the detected vehicle and Z_{new} is the distance between the ego-vehicle and the detected vehicle. w_{new} is the width of the detected vehicle in the camera sensor.

$$\frac{W_0}{Z_0} = \frac{w_0}{f} \tag{2.2}$$

$$\frac{W_{new}}{Z_{new}} = \frac{w_{new}}{f} \tag{2.3}$$

From Eq. (2.2) and Eq. (2.3), we could calculate the distance Z_{new}:

$$Z_{new} = \frac{W_{new}}{W_0} \times \frac{w_0}{w_{new}} \times Z_0 \tag{2.4}$$

In Eq. (2.4), $\frac{w_0}{w_{new}}$ is the ratio of the pixel numbers of ego-vehicle and the detected vehicle, regardless of the pixel size. The only unknown is W_{new}. When the dimensions of the detected vehicle are not known, OmniView chooses a default value for W_{new}. Once the detected vehicle responds to an Identify message for announcing its TID, its actual width is also declared and can be used to calculate the distance more accurately.

When a vehicle receives Identify message with its image from another vehicle, it can estimate the distance to the sender using a method based on image matching.

Initially, when a driver takes her own vehicle's self-images from known distances, the self-images satisfy Eq. (2.5). Here w_0 is the width of the vehicle in a self-image and Z_0 is the known distance when driver takes the self-image. f_A is the focal length of the corresponding smartphone camera.

$$\frac{w_o}{f_A} = \frac{W}{Z_0} \tag{2.5}$$

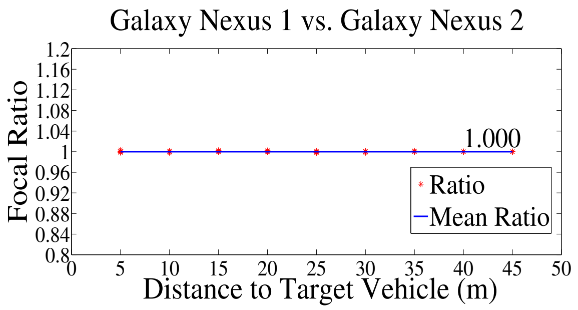
$$\frac{w_{new}}{f_B} = \frac{W}{Z_{new}} \tag{2.6}$$

When the vehicle is detected by another vehicle, we have Eq. (2.6). Here w_{new} is the width of the vehicle in the detected image; f_B is the focal length of the smartphone camera in the vehicle doing the detection; Z_{new} is the distance between the two vehicles.

From equation (2.5) and (2.6), we have:

$$Z_{new} = \frac{w_0}{w_{new}} \times \frac{f_B}{f_A} \times Z_0 \tag{2.7}$$

In this equation, $\frac{w_0}{w_{new}}$ could be derived from image matching. When matching two images, the scale ratio of the two matched objects (here vehicles) could be calculated. f_A and f_B are usually different, unless the models of the two smartphones are identical. For different smartphone models, we could pre-measure $\frac{f_B}{f_A}$ for all the popular smartphone pairs on the market, and then pre-install these ratios in OmniView. To do that, we collect images taken at same distances with different phones, and then match these same-distance images to get the ratios between different phone models. Fig. 2.12 exemplifies the focal ratio of different phone models. In communication, sender tells the counterpart the smartphone model it is using. Therefore the receiver could select proper $\frac{f_B}{f_A}$ to calculate the distance.



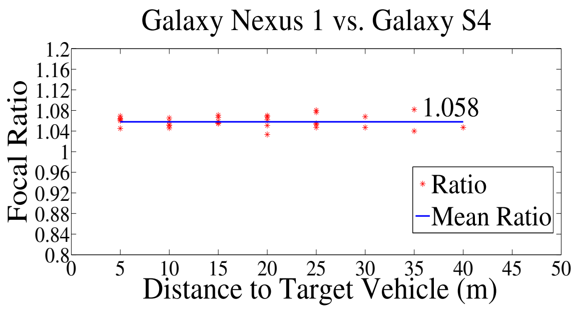


Figure 2.12: Focal ratio between phones. Pairs of images for the same vehicle are taken by compared phones at each distance (10 to 45m) and direction (Rear, Left, Right). Each pair of images are matched to calculate the focal ratio between different phones. Left shows the ratio between two Galaxy Nexus. These two are same model phones, so the focal ratio is 1.0; Right shows the ratio between Galaxy Nexus and Galaxy S4.

For communication purpose, we need to scale down the original image to be a smaller size in case it is too large. For instance, when sender detects receiver at distance $Z_{original}$, the size of the receiver vehicle in the original image is $w_{original}$, we have Eq. (2.8).

$$\frac{w_{original}}{f} = \frac{W}{Z_{original}} \tag{2.8}$$

After scaling down, the size of the receiver vehicle in the resulting image becomes w_{scaled} (which is the w_{new} in (2.7) from the receiver's point of view) and the distance $Z_{original}$ becomes Z_{scaled} which meets Eq. (2.9) (here the scaling down is carried out at the sender, so f is unchanged, which is the focal length of the sender's smartphone camera):

$$\frac{w_{scaled}}{f} = \frac{W}{Z_{scaled}} \tag{2.9}$$

Hence, Z_{new} computed by equation (2.7) is indeed Z_{scaled} in equation (2.9). From equations (2.8) and (2.9), we get:

$$Z_{original} = Z_{scaled} \times \frac{w_{scaled}}{w_{original}} \tag{2.10}$$

In this equation, $\frac{w_{scaled}}{w_{original}}$ is the scale factor S, which is transmitted along with the image in Identify message.

By combining equation (2.7) and (2.10), the actual distance could be derived from the following formula:

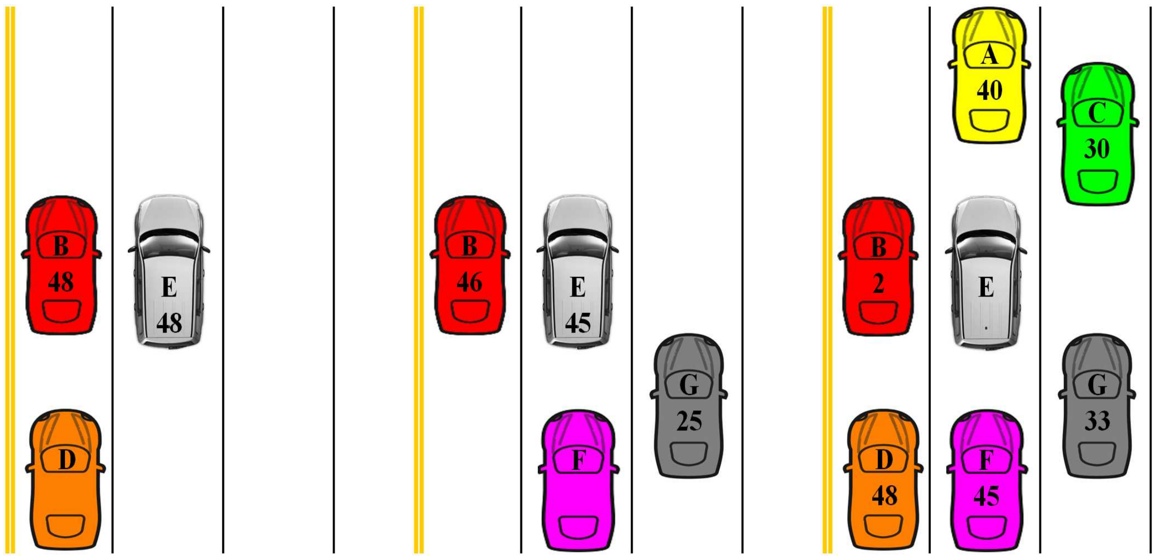
$$Z = \frac{w_0}{w_{new}} \times \frac{f_B}{f_A} \times Z_0 \times S \tag{2.11}$$

E. Map Construction

From vehicle detection, OmniView could obtain the lane position and distance of the detected vehicle, and then put the detected vehicle in its local map. Besides, participants periodically exchange local maps to help each other form more complete and accurate maps. Each time a vehicle receives a map from another vehicle, it searches for common nodes between its local map and the received map. If a common node exists, OmniView fuses the two maps by adding the distinct nodes from the received map into its local map. For instance, in Fig. 2.13(a),(b), vehicle D and F detect the vehicles moving in front and form their local maps. Vehicle E in (c) detects vehicle A and C.

After receiving the map from vehicle F, OmniView in vehicle E identifies the common node (which is vehicle E) in both maps. It fuses the maps through geometric computing based on the common node and the position information provided in the received map. Fig. 2.14 illustrates how vehicle F and B from F's map are merged into E's map. Here, d_{EF} and d_{BF} are derived from F's map, and w is the standard lane width, which could easily be known or estimated. After locating vehicle F directly by the direction and distance between vehicle E and F, OmniView can compute the value of h. Whereafter, by referring to d_{EF}, OmniView could position vehicle B into E's map.

After fusing the maps from vehicle D and F, vehicle E gets to know the presence



(a) Map of Vechiel D (b) Map of Vechiel F (c) Map of Vechiel E

Figure 2.13: Local maps at vehicle D (a) and vehicle F (b). The map at E (c) after merging its local map with the received maps from D and F. When E broadcasts its map, D and F also learn about the surrounding traffic. The numbers shown on the vehicles tell the distance in meters.

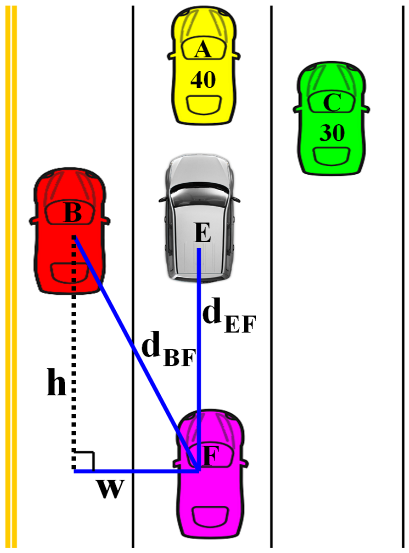


Figure 2.14: Locate vehicles from received map into local map.

and positions of B, D, F, G (as shown in Fig. 2.13(c)), which could not be seen by E itself. Therefore, vehicle E obtains an omni-view about its surrounding traffic, even some of its neighboring vehicles might not be participants, e.g. vehicle B.

Each time OmniView updates the local map, it checks whether any potential hazards exist. Different levels of alerts are defined for different hazards, from flashing icon on the screen to acoustic warning when the danger of collision increases based on the positions of neighboring vehicles.

2.3 Evaluation

As a safety related system, OmniView must provide the map about the neighboring vehicles in real-time, which requires all functional components to perform as efficiently as possible. Besides, the communication among OmniView-enabled vehicles must perform reliably to make sure the messages could be received by the peer participants, otherwise, it could miss out some vehicles on the map or provide driver with an out-of-date map.

In this section, we study the performance of the most important aspects of OmniView: The accuracy of visual ID recognition, the accuracy of distance estimation, the reliability of vehicular communication and the computational overhead as well as overall efficiency of OmniView.

A. Performance of Visual ID Recognition

OmniView uses visual ID to represent a vehicle. When a participant broadcasts an Identify message, every receiver needs to check whether the embedded VID is itself, a vehicle in its view or an unknown vehicle. When the corresponding target replies the Identify message with its TID, nearby participants add an entry in their cached mapping between the VID and TID.

To make the solution feasible and the information in the map correct, the system should be able to confidently distinguish whether a received VID matches one of its self images and currently detected vehicle images. The problem could be divided into two aspects: On one side, if two VIDs are for the same vehicle, whether the system could accurately conclude that they are the same. On the other side, if two

VIDs are for two different vehicles, whether OmniView could reliably tell that they are different.

To show the performance of OmniView in identifying VID for the same vehicle, we collect 50 groups of vehicle images. Each group contains images for one vehicle but taken at different distances. By taking image size into consideration (i.e. the image size should be above 10 KB as discussed in Section 2.2.C), we carried out image matching on 913 pairs of same-vehicle images.

Fig. 2.15 shows the accuracy of OmniView in recognizing two same-vehicle images with certain thresholds (only if the number of matched points between two images are above the threshold, the two images are concluded to be the same; otherwise, they are considered to be different). By choosing proper threshold for the number of matched points, our system could recognize two VIDs are for the same vehicle with a high probability (e.g. 99% for threshold = 15).

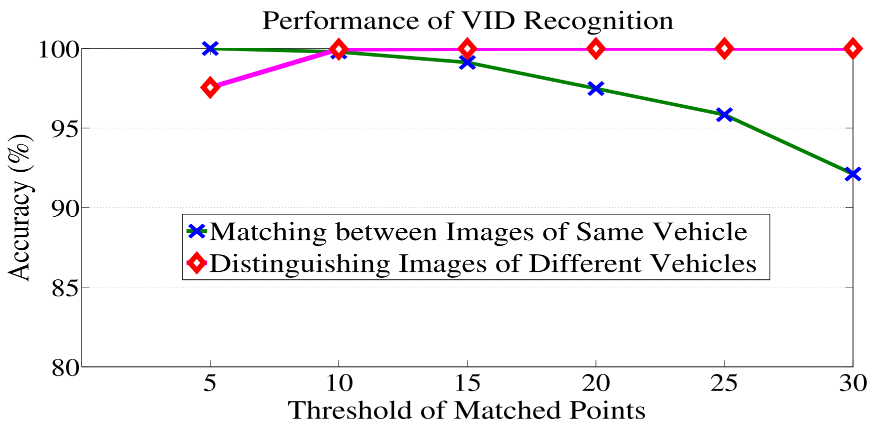


Figure 2.15: Performance of OmniView in recognizing Visual IDs for same vehicle and distinguishing Visual IDs for different vehicles.

In order to check how well the system could tell two VIDs are different in case they are really for different vehicles (here “different” means the appearance is different), we perform image matching on the image set used in Fig. 2.9. By taking image size limitation into consideration, we get 28245 pairs of images. From Fig. 2.15, we could see that OmniView could distinguish two different vehicles very accurately (close to 100% for threshold ≥ 10).

Along with the above study, one natural question arises: What if there are two same-appearance vehicles within the communication range? In case two same-appearance vehicles show up within a third vehicle's communication range, if the third vehicle wants to communicate with one of them, the other one might be confused. So we study to see how diverse the vehicles are within a certain range in real-life traffic and how often two same-appearance vehicles occur in the same range of an OmniView-enabled vehicle.

We took videos for the traffic of a multi-lane highway both in rush hour and off-peak hour. From these videos, we randomly extract 1000 "traffic snapshots". Each snapshot starts when a randomly selected vehicle (i.e. the first vehicle in the snapshot) passes a pre-defined baseline (as shown in Fig. 2.16). We count all the vehicles that pass the baseline after the first vehicle until the first vehicle has traveled about 80 meters away. We extract the images for all the vehicles in the snapshot at the moment they pass the baseline. These vehicles form an 80-meter-spanning snapshot of the traffic.

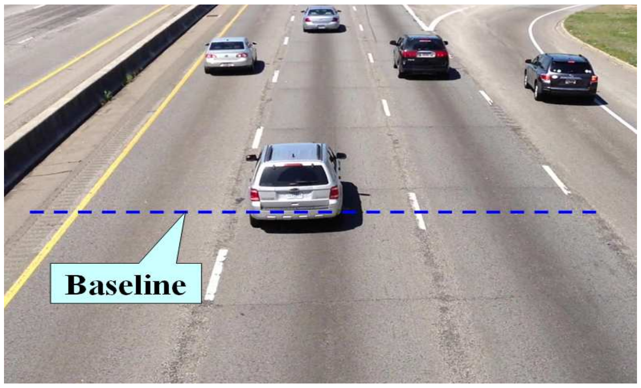


Figure 2.16: Snapshot of traffic.

To learn about the diversity of vehicles in the snapshots, we checked both manually and through image matching. From manual checking, we found that within the 1000 snapshots, only 3 of them contain two same-appearance vehicles. So we believe that the probability of two same-appearance vehicles show up in the same communication range is low enough for our system to work in reality.

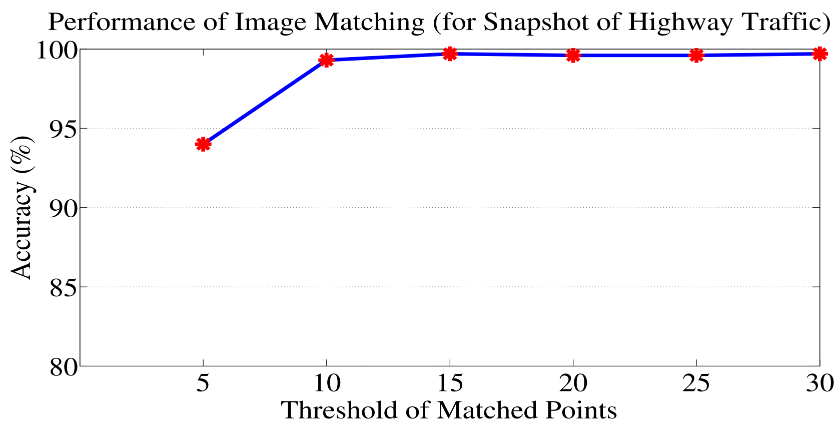


Figure 2.17: Performance of OmniView in recognizing Visual IDs for snapshots of highway traffic.

During image matching checking, we treat the result from manual checking as ground truth. Fig. 2.17 shows the accuracy of image matching for these 1000 snapshots with certain thresholds. From the figure we could see, by choosing proper threshold (e.g. 15), the system could identify the VIDs with very high probability.

One thing we need to point out is, when we took the videos for the diversity study, we were on an overpass bridge over highway. Although the bridge is not high, the angle at which the camera sees these vehicles are still slightly different from the actual situation the OmniView system will see. When OmniView system is used in a vehicle, it sees exactly the left-back, rear-end or right-back sides of other vehicles. While in our videos, we also see the roof of the vehicles due to the angle from the bridge. This might slightly affect the result. But considering the appearance of vehicles, for two vehicles, their similarities and differences are usually consistent from their roof to their left/rear-end/right sides.

Even in the case there are two exactly same-appearance vehicles in one communication range, the relative lane position between sender and target provided in the message could help to identify which one the sender is referring to. For example, there are two same-appearance vehicles moving closely on a multi-lane road, one is on the left-most lane, the other is not. If they receive a message telling that the target should be on the right lane of the sender, the vehicle on the left-most lane could easily skip processing the VID in the message, because it could not be on someone's right lane.

With the above study, we believe that OmniView could reliably identify vehicles to form the map about surrounding traffic.

B. Distance Estimation

In order to form a useful map for improving safety, besides the correctness of the vehicle information, the distance between the vehicles should also be estimated correctly.

To assess the accuracy of distance estimation by OmniView, we took pictures of 10 vehicles, with four phones (Galaxy S4, iPhone 4S and two Galaxy Nexus) from 3 directions (left, right, and rear) and different distances (ranging from 1 to 50 meters). We kept the vehicles stationary in this preliminary evaluation, since it is not easy to obtain the ground truth on distance between vehicles, when both are moving. The images taken by one Galaxy Nexus (named as Galaxy Nexus 1 here) are treated as self-images. The focal ratio between phones is precomputed as described earlier in Section 2.2.D. The measured distance vs. ground truth is plotted in Fig. 2.18. It shows that OmniView can estimate the distance accurately, particularly up to 45 meters. Admittedly, these results, while very encouraging, are obtained when the vehicles are stationary. A part of our on-going work is to assess the accuracy of OmniView in real driving scenarios.

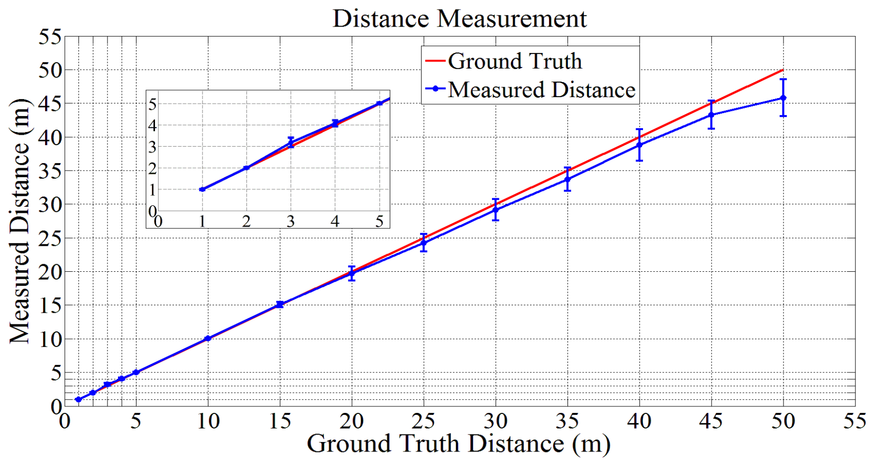


Figure 2.18: Distance estimated by OmniView compared with the ground truth.

C. Vehicular Communication

In OmniView, vehicles need to transmit detected vehicle images and map information. As discussed in Section 2.2.C, the sizes of images will be in the range of 10∼14 KB. OmniView slices each image into 1-KB packets for transmission. Although we need to add other information, such as sequence number, into the Identify message, the size of these information is negligible compared with the image itself, so we just treat them as a little portion of the image in studying the communication performance. As to Map message, which only contains text, therefore we use 512 Bytes to convey the map.

We use simulation to study the performance of the vehicular communication of OmniView system. As a vehicular network, what is different from conventional networks is that every node (i.e. vehicle) in the network is moving. We need to simulate both the mobility of vehicles and the communication among vehicles. SUMO [29] (sumo-0.12.3) is used to generate the vehicle mobility, which is fed into NS2 [30] (NS2.34, which supports DSRC 802.11p) to carry out the network communication.

We simulated three transmission ranges: 60 m, 80 m and 100 m, we believe 60∼100 m is a reasonable range which allows enough time for drivers to take proper maneuver to keep safe. We also studied the performance of OmniView under two different traffic conditions: one is dense mode, in which the traffic flow is relatively heavy and each vehicle has many neighboring vehicles; the other is sparse mode, in which every vehicle has fewer neighbors than in dense mode. The details of the parameters used in our simulation are listed in Table 2.4.

Table 2.4: Setting of Simulation for OmniView Vehicular Communication

Parameter	Remark
Simulation Period	200 s
Number of Vehicles	400 (6 types of vehicles with different length/speed/acceleration/deceleration; vehicles' speeds are dynamically changing in speed range.)
Speed	22∼36 m/s (50∼80 mph)
Traffic Density	Sparse (58.8 vehicles/km) Dense (102.7 vehicles/km)
Wireless Protocol	DSRC 802.11p
Antenna Type	OmniAntenna
Radio Propagation Model	Two Ray Ground
Data Rate	6 Mbps (QPSK) DSRC could support up to 27 Mbps data rate, but 6 Mbps is the optimal data rate which achieve good performance [21]
Message	Identify message (10∼14 KB, sliced into 1-KB small packets) Map message (512 B)
Message Life Time	Image message: 0.6 s Map message: 0.4 s
Transmission Method	Periodic Broadcast
Transmission Frequency	Identify message: 0∼3 images every 3.5±2.0 s Map message: Once every 0.4±0.2 s
Transmission Range	60, 80, 100 meters

We measure the message reception rate, which is defined as:

$$\frac{\#Nodes\ in\ range\ \&\ received\ the\ message}{\#Nodes\ in\ sender's\ transmission\ range} \tag{2.12}$$

The message reception rate for Identify and Map messages in two traffic modes are shown in Fig. 2.19. We can see that Map message could be exchanged very reliably for all the three transmission ranges in both sparse and dense traffic modes. In case of Identify message, with short transmission distance, OmniView could achieve about 78∼84% reception rate in dense mode, while in sparse mode, the number goes up to 87∼90%. When the transmission range increases, the reception rate decreases, especially in dense mode. The reason behind this is that every vehicle will be in the transmission ranges of more other vehicles, hence more network collisions will happen. But to a vehicle, the other vehicles which are far from it are less dangerous than the ones in short distance, so in many cases, we could stick to the shorter transmission range. Here we also set a relatively strict message life time for Identify message (i.e. 0.6s). We could allow longer life time for Identify message (because its main purpose is to get the corresponding TID), then we will have higher reception rate.

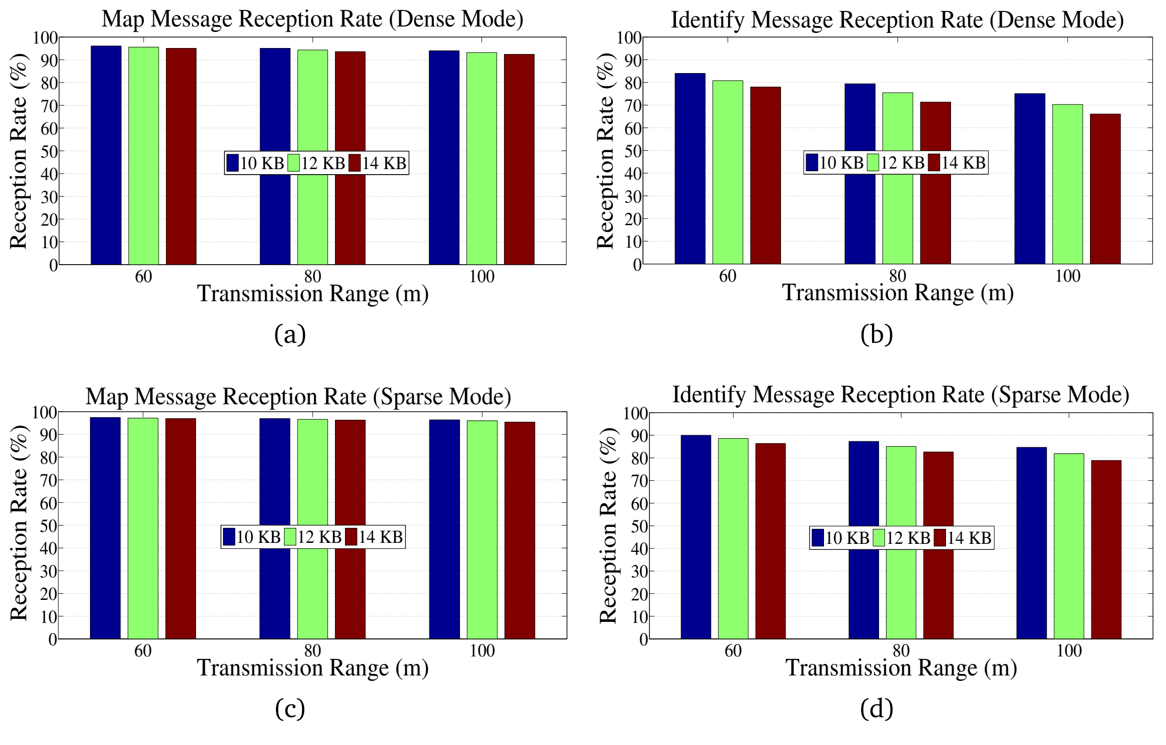


Figure 2.19: (a) Map message reception rate in Dense Traffic Mode (three types of image sizes and transmission ranges are compared; Map message is fixed to 512 Bytes). (b) Identify message reception rate in Dense Traffic Mode. (c) Map message reception rate in Sparse Traffic Mode. (d) Identify message reception rate in Sparse Traffic Mode.

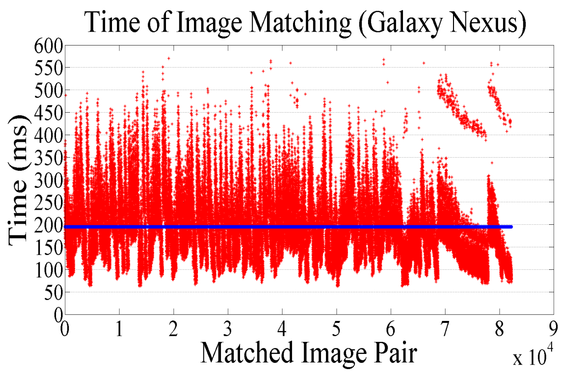
As mentioned earlier, before the sender sends an Identify message, it calculates position for the detected vehicle, thereby the sender should be safe. For the receiver, the probability of receiving a single Identify message might not be as high as Map message. But OmniView is a collaborative system, every vehicle will likely be detected by more than one vehicle moving behind it (in the same lane or different lanes). All those vehicles will send Identify messages to it, therefore the receiver will have more chances than what is shown in Fig. 2.19 to receive at least one of those Identify messages. Once it receives one and announces its TID (which is a Map message and its reception rate is high), the vehicles behind it could construct the VID-TID mapping. The subsequent Identify messages related to this receiver will turn into Map messages, which could be received more reliably. We can expect that in OmniView system, every vehicle will have high probability to obtain the positions of its neighboring vehicles.

D. Efficiency

OmniView must update the local map as quickly as possible to let the driver know what is happening around her in real time. In the OmniView system, time is mainly consumed in three parts: vehicle detection, message transmission and visual ID recognition.

We measured the vehicle detection in real traffic. When a vehicle occurs in the video frame, OmniView could detect the vehicle within 97 $\pm$ 17 ms (measured on a Galaxy S4).

To measure the time of visual ID recognition, we prepared 406 vehicle images. Among the 406 images, some of them are for same vehicles, others are for different vehicles. We measured the time of image matching with all possible image pairs on a Galaxy Nexus, Fig. 2.20 (left) shows the time. The image matching could complete in 195$\pm$74 ms.



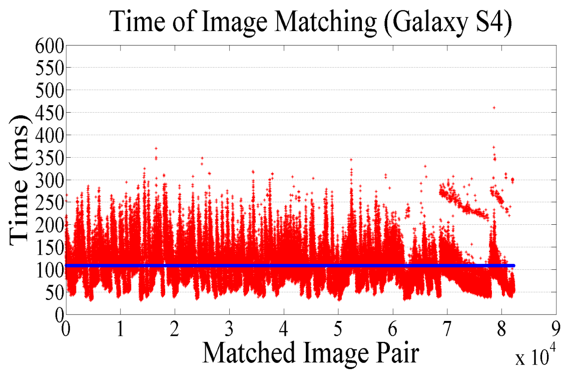


Figure 2.20: (Left) Time of image matching measured on Galaxy Nexus. OmniView could match images on Galaxy Nexus in 195±74 ms. (Right) Time of image matching measured on Galaxy S4. OmniView could match images on Galaxy S4 in 108±44 ms.

From the simulation, we also measured the end-to-end communication latency for Identify message and Map message. The results are exhibited in Fig. 2.21.

From the above measurement, we can expect that in OmniView, the total time spent on vehicle detection, communication and visual ID recognition is about 400 ms, which is real-time. Considering the relative speed between vehicles on highway is mostly less than 30 mph ($\approx$ 13.5 m/s), the distance between any two vehicles changes less than 5 meters from a message is initialized at the sender to the the message is processed at the receiver.

Here the time is measured on a Galaxy Nexus phone, which is an old model (released in 2011). When more powerful smartphones come out, we could expect that OmniView will perform more and more efficiently. For example, if we use Galaxy S4 (which is released in March, 2013 and more powerful than Galaxy Nexus), OmniView could match images in 108±44 ms, as shown in Fig. 2.20 (right). Thus OmniView could provide and update the map about neighboring vehicles more efficiently and more accurately as time goes on.

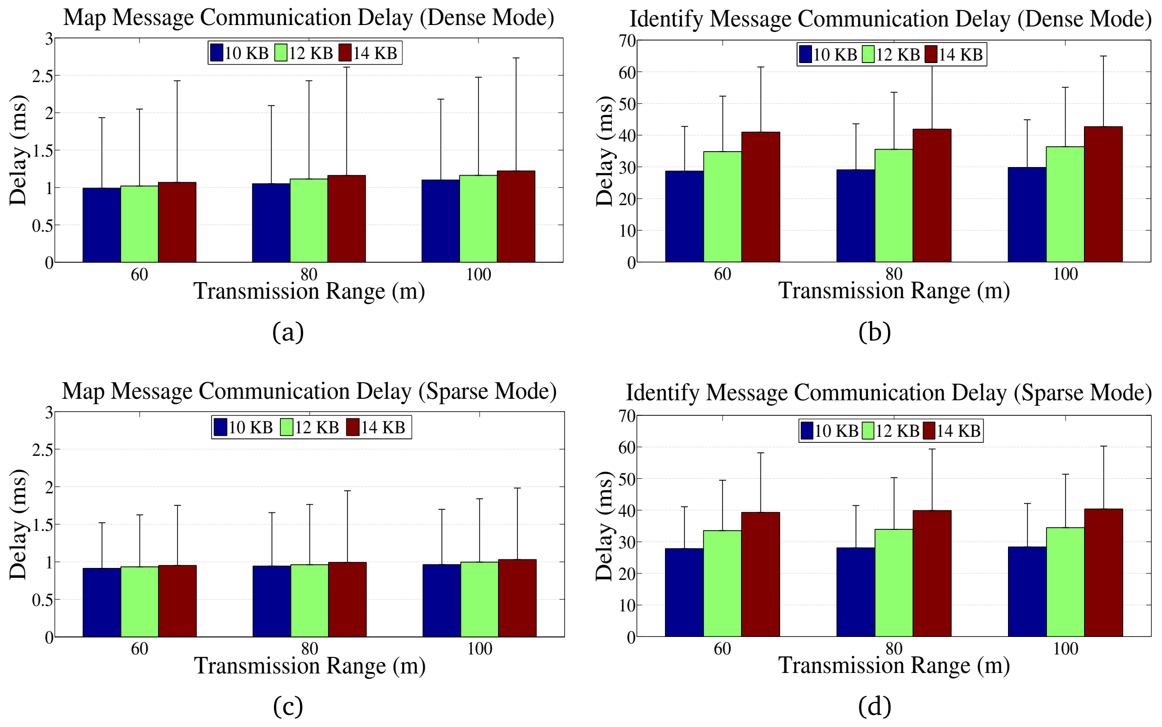


Figure 2.21: (a) Communication delay of Map message in Dense Traffic Mode (three types of image sizes and transmission ranges are compared; Map message is fixed to 512 Bytes). (b) Communication delay of Identify message in Dense Traffic Mode. (c) Communication delay of Map message in Sparse Traffic Mode. (d) Communication delay of Identify message in Sparse Traffic Mode.

E. Computational Overhead

In the OmniView system, image matching is the most computation-intensive work. If OmniView has to spend all or most of its time on image matching, the system might not be able to reflect what is happening around in real-time. Here we study the computational overhead of OmniView.

Fig. 3.7 shows how many Identify messages each vehicle will receive every second on average, which corresponds to how much image matching task each OmniView-enabled vehicle needs to carry out. Although each OmniView-enabled vehicle receives 4$\sim$6.5 images per second, with the knowledge of lane position (as described in Section 2.2.D), OmniView could easily filter out the Identify messages which are not targeting at it. It does not need to do image matching on all the

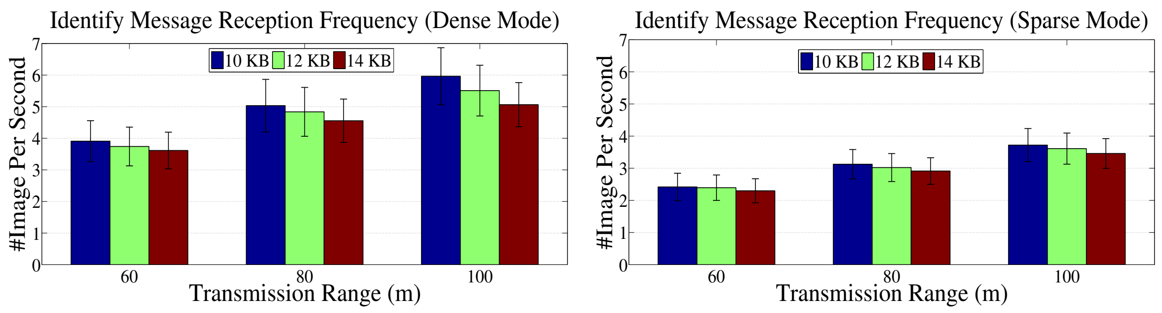


Figure 2.22: (Left) The number of Identify messages each OmniView-enabled vehicle receives every second in Dense Traffic Mode. (Right) The number of Identify messages each OmniView-enabled vehicle receives every second in Sparse Traffic Mode.

received images. Besides, smartphones' processors are becoming more and more powerful. Many of them now have multi-cores, the image matching task could be carried out in parallel on multiple cores, thereby the time on image matching will become even less.

2.4 Related Work

Driving safety has attracted attention of both industrial developers and academic researchers for a long time. Many collision-preventing systems have been proposed and deployed. In industry, many vehicles are now equipped with radars and cameras, which help drivers detect the objects around the vehicle. But most of these systems are only usable in a short distance and low speed, usually help the driver in backing or packing [11]. Some new-model vehicles, especially luxury vehicles, such as Mercedes-Benz, start to provide systems to help driver detect the objects in a relatively long distance. These components are not only expensive (usually more than $1000 [31]), but also usually only cover part of the scene around the vehicle.

In academia, researchers have studied many methods to improve driving safety by helping drivers to know about other vehicles surrounding them. Authors in [32] mount two omni-cameras near the side mirrors, which uses pure computer vision to detect vehicles moving in front and on two sides of the ego vehicle. The work in [33] also suggests omni-directional vision-based system to discover surrounding vehicles and obstacles. It requires a multi-camera system to be mounted on top of the vehicle. These systems require particular hardwares to be installed at some unusual place in vehicle and purely work on their own. They could only see up to 40 meters which makes them unsuitable for scenario like driving on highway.

To know the positions of vehicles, one nature way is using GPS. Every vehicle obtains its own position and tells others. In this way, the drivers could know the positions of their neighboring vehicles. In [34–38], each vehicle uses GPS to get the location of its own and communicates over wireless to help each other know about the positions. But GPS has its innate weakness. First, the accuracy of GPS for civilian usage is not high. Its error could be up to 30 meters [12], which is too large to distinguish lane-level position, considering the lane width is only about 2.7$\sim$3.6 meters [39]. Although Differential GPS [13] could help to reduce the error, but it requires deploying base stations all along the road, which is very costly. Second, GPS outage occurs in many places. When traveling in urban area, mountains, valleys, tunnels, etc., GPS satellites are partially or totally blocked, which makes GPS system totally unusable. Another problem of these GPS-based solutions is the penetration rate. It requires every vehicle's participation. If some of them do not have GPS or do not share the location information, other vehicles have no way to get those vehicles' positions, which limits their usage.

OmniView system adopts a different strategy. It uses ubiquitous smartphone as the only required device, which incurs no extra cost since smartphone has become a standard equipment for almost every driver. Each participant detects other vehicles and estimates the relative positions of the detected vehicles. Even some vehicles do not participate, the participants could still obtain the positions of those vehicles. OmniView-enabled vehicles collaborate with each other to exchange all the position information they got about other vehicles, which helps each participant to obtain the positions of the vehicles which can not be observed by its own. OmniView also estimates the lane position of every vehicle. Compared with the omni-camera or radar-based standalone systems and GPS-based collaboration systems, OmniView could see further and provide more fine-grained position information of neighboring vehicles. Meanwhile it requires lower penetration rate than GPS-based systems.

2.5 Limitations, Future Work and Expansion

OmniView uses computer vision related algorithms/technologies to detect vehicles, match images and calculate the positions of vehicles. Computer vision algorithms/technologies are easily affected by light condition. When the light condition is poor, the appearance of the vehicle might not be easily observed, hence vehicle detection will be difficult. The light condition could also affect the feature point extraction in image matching. When it is dark (for example, at night), current OmniView could not work. The proposed OmniView could be a complementary mechanism to other light-condition free solutions, such as GPS-based system. One possible way to make OmniView also work under poor light condition is to detect vehicles by the vehicle lights (different vehicle models usually have differently-shaped lights) and use the distance between the left and right rear-end lights to infer the distance to the vehicle being detected.

The second limitation of OmniView is that it relies on the diversity of the neighboring vehicles. If two vehicles are exactly the same, in case there is a third vehicle detecting one of them and broadcasting the detected image, it is hard for these same-looking vehicles to tell which one the third vehicle is detecting. We have partially solved this problem by using lane position. When detecting a vehicle, both the lane positions of the ego-vehicle and the target vehicle are identified. The lane position information is transmitted along with the image. If the receiver's lane position does not match the lane position transmitted in the Identify message, it ignores it. This could solve the problem when two same-looking vehicles are on different lanes. If there are two same-looking vehicles moving together on exactly the same lane (one behind another), the lane position information does not help to distinguish them.

Most highways have two directions and the lanes for opposite directions are very close (usually only a barrier between them). Consequently the transmission from the vehicles in one direction could also be received by the vehicles in the other direction. We could utilize smartphone's motion sensors to identify the moving direction of the vehicles. In the message, we could attach the direction information to help the receiver identify the moving direction of the sender. If the sender is in the opposite direction, the receiver could directly ignore this message.

The OmniView system could easily expand to show a map not only about the neighboring vehicles, but also the vehicles multi-hop away. With this expanded map, each driver could learn about the traffic situation on her way ahead. If there is congestion ahead, the driver could select a different route to travel. The driver could also learn about the traffic situation behind her from the map, which is also useful. For example, if one driver finds out that the traffic on her lane behind her is congested, it is very likely that this driver herself is moving too slow and blocking the vehicles moving behind. She could change to the next lane to release the traffic on her lane.

2.6 Summary

In this work, we introduced OmniView, a smartphone-based collaborative system for assisting drivers. With OmniView, each participant detects other vehicles and estimates their positions to form a local traffic map. The vehicles exchange the detected vehicle images as well as their local maps to help each other form a more complete and accurate map with the positions of neighboring vehicles. Our evaluation shows that OmniView could work reliably and provide a map of the traffic surrounding a vehicle in real-time.

Chapter 3

DriverTalk: Enabling Targeted Communication Between Drivers

When driving on the road, there are many situations where drivers have the desire to communicate with other drivers nearby. For example, a vehicle moving behind is tailgating, which exerts high pressure on the driver in the front vehicle and easily causes accidents [40]. The driver in front wants the tailgating vehicle to leave some space. When a driver intends to change lane, she wants other drivers in the target lane to be aware of her intent and give her space. On highway, a slow-moving vehicle occupies the passing lane and blocks the traffic, the driver behind wants to remind the driver in the slow vehicle to move away to release the traffic. Another common desire and behavior during driving is inquiring information from others. For example, when driving in an unfamiliar area, a driver wants to know the highway exit to a local landmark. In the traffic flow, especially on highway, drivers might not be able to stop to search or ask a local resident. If she could ask the drivers moving around, it's very likely that they have the information.

Automobile industry has been advancing for a very long time, but the function allowing drivers to talk to each in-situ is still not there. Current vehicles have been equipped with some components which allow drivers to convey some intent of driving. When changing lane, a driver could give signal by blinking lights. When a driver wants to slow down, she depresses her brake and then the vehicle will illuminate the brake lights. But these kinds of light indicators only work in passive mode, if other drivers do not notice these, the function of the indicators is immediately limited.

Horn is another equipment which helps a driver to tell others something about her intent or remind other drivers about something related to their driving. For example, at an intersection, the traffic light has changed from red to green, but the leader of a queue doesn't move. The drivers behind may honk to remind her. Horn has several practical problems. First, when a driver honks to alert someone, all other drivers around also hear it and have to figure out whether they are the intended target and what the intended message is, causing unnecessary cognitive load on them. Second, although horn is a standard equipment in vehicles, honking to other vehicles is usually considered as an impolite behavior. Besides, in many countries, for instance India, honking has become a serious noise pollution source [41]. In recent years, some countries, especially in cities, honking is not encouraged or even has been banned [42]. To say the least, honking could not express more detailed intents besides simple reminding.

The core limitation of existing modes of communication between drivers through horns and lights is that they are inadequate in conveying a message to and only to the intended target. To address the need of talking to the neighboring drivers and convey intents of maneuver actively and precisely, as well as a safe and efficient way for drivers to seek and provide necessary information from/to neighboring drivers, we propose a smartphone-based system, *DriverTalk*, leveraging the smartphones of drivers. DriverTalk enables a driver to talk to a particular driver, some, or all surrounding drivers, and clearly express her driving intents and/or query necessary information.

DriverTalk is an ad-hoc based vehicular network system, which does not rely on any centralized infrastructure. The system allows drivers on the road, with no prior acquaintance, to talk to each other. Similar to OmniView (Chapter. 2), it utilizes the appearance images of vehicles as their Visual ID (or termed as VID) to refer to the senders and receivers of messsages. DriverTalk could help most drivers, from high-end vehicle owners to legacy and/or economy vehicle owners, in improving safety and reducing stress while driving.

With the feasibility study of DriverTalk by measuring its efficiency and evaluating its communication performance in both highway and city traffic scenarios, this work shows the potential to introduce new features into vehicles for assisting drivers in a user/environment friendly manner.

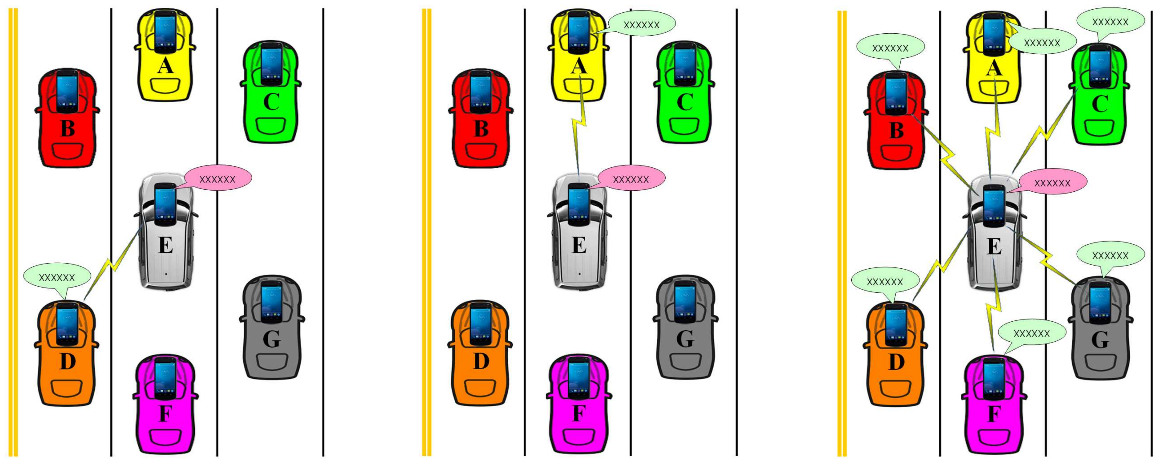
3.1 Application Scenario

DriverTalk uses the same setup as OmniView. Before using the system, drivers take self images for their vehicles as shown in Fig. 2.4 and mount the their smartphones on dashboard or windshield inside their vehicles as illustrated in Fig. 2.2.

When talking to other drivers in other vehicles, a driver instructs DriverTalk via voice command to convey the information she wants to express. Based on what the driver says, DriverTalk infers who should be the receiver and embeds proper VID in the message to identify who is sending the message, or whom the message is targeting at. The VID could be a self-image or an image of the vehicles moving in front of the ego-vehicle which is currently seen by DriverTalk in the ego-vehicle.

Based on whom a driver wants to talk to, DriverTalk system works in three typical scenarios:

a) Talking to driver(s) moving behind: Suppose a driver (say in vehicle E in Fig. 3.1 (a)) wants to talk to the drivers moving behind her (i.e. D, F, G). This could be to notify target drivers of the traffic situation ahead (e.g. accident), convey intent of maneuver (e.g. changing lane) or even request to the driver behind (e.g. request F not to tailgate). Depending on which vehicle moving behind a driver wants to talk



(a) (b) (c)

Figure 3.1: (a) Talk to a driver behind the ego-vehicle. (b) Talk to a driver in front of the ego-vehicle. (c) Talk to all drivers moving around.

to[1], DriverTalk broadcasts the self-image(s) taken in the corresponding direction as VID, along with the information the driver wants the receiver to know. When the vehicle moving behind receives the message, it compares the received VID with the vehicle images in its view to find out whether one of the vehicles moving in front is talking to it and which one is the sender.

b) Talking to driver(s) moving in front: Driver of E may want to talk to the drivers moving in front of her (i.e. A, B or C, as shown in Fig. 3.1 (b)). This could be to notify the driver(s) moving in front about the situation behind, e.g. vehicle A is blocking the traffic. Then, DriverTalk extracts the image of the target vehicle in its view, and uses it as VID along with the information the driver wants to convey. Upon receiving the message, DriverTalk in the target vehicle compares its self-images with the received VID to check whether someone is talking to it.

c) Talking to all drivers around: A driver may want to talk to all surrounding drivers (as illustrated in Fig. 3.1 (c)) to inquire about some information or alert them about a hazardous situation. In this case, DriverTalk broadcasts the self-

[1]In this work, talking to a vehicle means talking to the driver in that vehicle.

images or the vehicle images it currently sees along with the information the driver wants to convey. Surrounding vehicles which receive this message compare the received VIDs with their self-images or vehicle images in their own view to infer the sender. If the message is a question and the receiver happens to know the answer, the receiver could send a message back to answer. In the answer message, the questioner's VID (which is received in the question message) is used to tell the target of the answer.

3.2 System Overview

The overall system flow of DriverTalk is illustrated in Fig. 3.2.

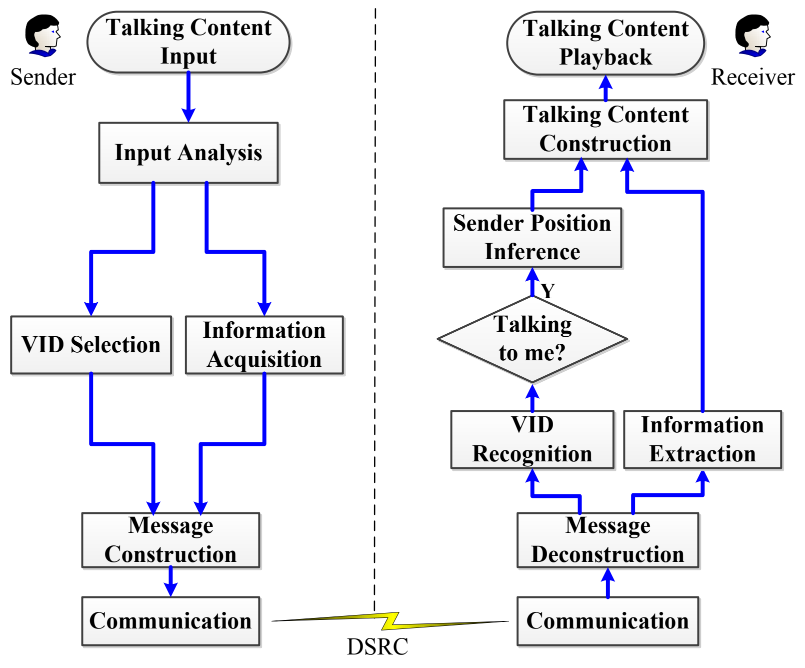


Figure 3.2: DriverTalk system flow. At the sender side, the system analyzes a driver's voice input and constructs message by combining the information input by the driver and the image selected as VID based on the driver's intent. At the receiver side, the VID is extracted and checked against the self images of receiver or the vehicle images seen by the receiver to decide whether the message is for this driver. If this driver is the target receiver, the message and the inferred sender's position is combined and played back to the driver.

At the sender side, a driver inputs via speech. DriverTalk analyzes what the driver says by identifying pre-defined keywords. Once any keywords are recognized, DriverTalk infers the target of this talking, i.e. which neighboring driver the sender driver wants to talk to. Based on this information, proper VID is selected from self images of the sender vehicle and other vehicle images currently seen by DriverTalk. Finally a message is organized by combining VID and the information the driver wants to convey. In DriverTalk, all messages are broadcast over DSRC [20]-based vehicular network.

Once the vehicles moving around the sender receive the message, DriverTalk system at those vehicles deconstructs the message to get the VID and the content of talking. DriverTalk compares the received VID with the receiver's self images and the vehicle images in its view. If one of them is matched, DriverTalk recognizes that this message is for this receiver, hence it infers the relative position of the sender. DriverTalk combines the sender's position and the message content to play back for the receiver driver.

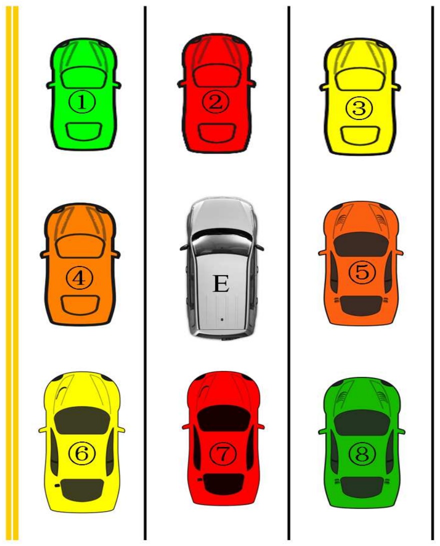


Figure 3.3: The relative positions of the vehicles moving around the ego-vehicle.

In DriverTalk, we define eight types of relative positions between a vehicle and its neighboring vehicles, as depicted in Fig. 3.3. Note that these 8 positions just indicate the relative direction from the ego-vehicle, while their distances to the ego-vehicle can vary. The only requirement is that they are all at one visual-hop from the ego-vehicle, without any other vehicles in between.

To realize DriverTalk, the following issues should be addressed:

- How can the system allow drivers to convey the information in a natural and simple way?
- How can a driver effectively recognize that another driver is talking to her?
- How efficiently and reliably could the system perform?

These questions are answered in the upcoming sections.

3.3 Design & Implementation

As exhibited in Fig. 3.2, DriverTalk needs to implement following components in order to effectuate end-to-end talking between drivers:

1) Vehicle Detection: Detect the vehicles moving in front and extract the appearance images of the detected vehicles, which will be used as VID along with self images in messages.

2) Input Analysis: Analyze what a driver says and infer the target of talking by identifying keywords.

3) Message Construction: Construct message based on the selected VID corresponding to the target of talking and the information the driver wants to convey.

4) Vehicular Communication: Send message to the target vehicle(s) moving around the sender.

5) Visual ID Recognition: Check VID in the received message and identify sender and the target receiver.

6) Inference of Sender's Position: Infer the relative position of the sender from the received message.

7) Message Playback: Play back the received talking content.

DriverTalk adopts the vehicle detection model trained in OmniView to perceive the presence of the vehicles moving in front. DriverTalk also uses ORB [28] for image matching in visual ID recognition, which could perform efficiently on smartphone. When the size of the detected image is controlled within 10$\sim$14 KB, the system could confidently identify whether two images are for the same vehicle or different vehicles. DriverTalk exchanges messages among participants through broadcasting over DSRC as depicted in Section 2.2.B.

In this section, we clarify how DriverTalk analyzes driver's input, constructs message, infers the position of message sender and also describe how the message is played back at the receiver. Besides, several strategies are also discussed to regulate the usage of the system.

A. Input Analysis

Operating on smartphone during driving is dangerous. To let the drivers concentrate on their driving when using the DriverTalk system, we adopt voice input in DriverTalk. Drivers are allowed to talk in a natural way, i.e. speech, without any hand operation on the smartphone.

To discriminate the talking over DriverTalk from regular talking that happens inside the vehicle, e.g. a driver is talking with a passenger, we define a foreword, "OK Driver", for the system, just like the way Google Glass is working. When a driver wants to talk to other drivers over DriverTalk, she says "OK Driver" first, and then expresses what she wants to convey to surrounding drivers. The system must recognize the foreword, and thereby understand what the driver is talking and whom the driver is talking to. Fig. 3.4 shows the process of analyzing driver's input in DriverTalk.

Nowadays, speech recognition module is available on all the popular smart-

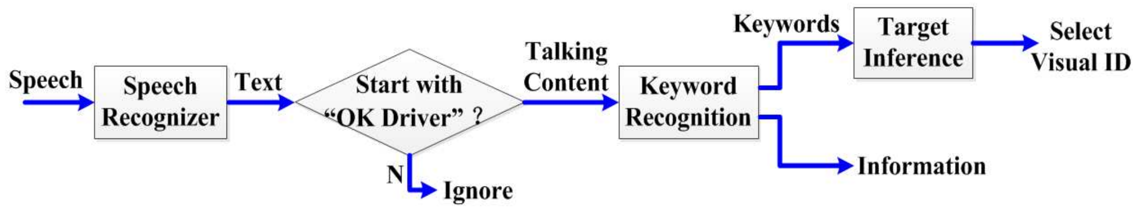


Figure 3.4: Process of input analysis in DriverTalk. This phase extracts the keywords and infers the target of talking.

phone platforms, for instance Siri on iOS. In DriverTalk, we use Android SpeechRecognizer to recognize what the driver says.

DriverTalk uses wireless communication to send what a driver says to another driver. If the system directly transmits driver's speech, the data size will be too large (could be tens to hundreds KBs even for a several-second sentence) to achieve reliable and efficient performance. DriverTalk converts driver's speech into text through SpeechRecognizer. Later, it transmits text instead of acoustic content to the receivers. In this way, the communication overhead is significantly reduced.

To ease the driver's effort when talking to the drivers around her and make the speech recognition module work more efficiently, we define a set of keywords, which map to a set of pre-defined common messages[2]. The keywords and their mapping to the pre-defined messages, the corresponding target vehicles of talking when using the messages are listed in Table 3.1. With this definition, instead of saying a long sentences (3rd column in the table), drivers only need to speak the keywords, which could improve the performance of speech recognition.

After recognizing the keywords defined in the table (2nd column), if the keywords are associated with some pre-defined message, DriverTalk fetches the index of the corresponding pre-defined message. The system only needs to transmit the index instead of the complete message to the receiver. At the receiver side,

[2]In this work, based on the context in narration, the word "message" could be one of the two kinds of meaning: a) The content of talking. b) The communication/network term for data packet, which includes network address, talking content and other necessary information.

Table 3.1: Mapping of Keywords–Predefined Message–Target Vehicle

Index	Keyword	Message	Target Vehicle(s) of Talking
1	Change Left	I want to change to the left lane.	Vehicles on left lane moving parallel and behind (at position ④ ⑥).
2	Change Right	I want to change to the right lane.	Vehicles on right lane moving parallel and behind (at position ⑤ ⑧).
3	Tailgate	Don't tailgate me.	Vehicle moving behind on the same lane (at position ⑦).
4	Too Slow	You are moving too slow and blocking the traffic.	Vehicle moving in front of ego-vehicle on the same lane (at position ②).
5	Lane Closed	This lane is closed ahead.	Vehicle moving behind on the same lane (at position ⑦).
6	Accident	There is accident ahead.	All vehicles moving behind on the same or adjacent lanes (at position ⑥ ⑦ ⑧).
7	Problem	There is something wrong with my car.	All vehicles moving around.
8	Slow Down	I will slow down.	Vehicle moving behind on the same lane (at position ⑦).
9	Move	Please move.	Vehicle moving in front of ego-vehicle on the same lane (at position ②).
10	Light On	Your light is on.	Vehicle moving in front of ego-vehicle on the same lane (at position ②).
11	Question	Question:	All vehicles moving around.
12	Answer	Answer:	The sender of the corresponding question message.

DriverTalk brings back the corresponding message and plays back (which will be described in Section 3.3.D). At the same time, the system infers which target vehicle the ego-driver is talking to based on the relationship depicted in this table. The knowledge of target vehicle's position is used to select proper VID for the message to be transmitted, which will be clarified in Section 3.3.B.

Different from the regulated messages defined as 1$\sim$10 in Table 3.1, the keywords “Question” and “Answer” give more flexibility, which could be used to ask questions and answer questions. Once DriverTalk detects one of these two keywords, it converts all what the driver says after the keyword into text as the content of question or answer. The target of a question message is all vehicles moving around the questioner, while the answer message is only targeted at the corresponding questioner.

In the future, we will define more pre-defined messages and possibly provide drivers the opportunity to talk in a more flexible way.

B. Message Construction

The message exchanged between DriverTalk-enabled vehicles contains two major parts: VID and the content of talking. Table 3.1 defines the regulated talking content. When the keyword “Question” or “Answer” is detected, DriverTalk converts everything following into the content of talking. VID is used to identify the sender or receiver of the message. Here below, we describe how the system selects VID.

Based on the intent of talking, at the sender side, different images should be selected as VID. It is inconvenient and unsafe for drivers to operate on the screen to manually select target vehicles. Following the relationship between keywords and target vehicle of talking defined in Table 3.1, DriverTalk could infer the position of target vehicle after recognizing keywords, whereafter it automatically selects proper images in order to talk to the target vehicle.

Table 3.2: Rules for VID Selection

Target Vehicle	Selected VID
Vehicle at position ①	Detected vehicle image of vehicle at position ①
Vehicle at position ②	Detected vehicle image of vehicle at position ②
Vehicle at position ③	Detected vehicle image of vehicle at position ③
Vehicle at position ④	Detected vehicle image of vehicle at position ① (if exist); otherwise the detected vehicle image of the vehicle at position ②
Vehicle at position ⑤	Detected vehicle image of vehicle at position ③ (if exist); otherwise the detected vehicle image of vehicle at position ②
Vehicle at position ⑥	Left-back self image
Vehicle at position ⑦	Rear-end self image
Vehicle at position ⑧	Right-back self image
Vehicle at position ⑥ ⑦ ⑧	Self images at all directions
All vehicles moving around	Vehicle Problem Warning: All detected vehicle images and rear-end self image Question: One self image
Questioner	Image received in the related question message

Table 3.2 defines what images will be selected as VID corresponding to the position of the target vehicle. Here a special case happens when the target vehicle is at position ④ or ⑤. Smartphone camera has a limited field of view. When two vehicles are moving parallel, they cannot detect each other, hence they are not able to get the VID of each other. At this moment, if one of them wants to change lane, even the driver gives light indicator, the other driver cannot see it, so collision might happen. To avoid this hazard, DriverTalk utilizes a third-party VID, which is the image of the vehicle moving in front of these two vehicles. The front vehicle could be detected by both of these two parallel vehicles. Once one vehicle finds out that someone else is sharing the same scene (i.e. the vehicle being detected in its field of view), it could learn that another vehicle is moving parallel to it.

When a driver speaks the keywords related to lane changing and DriverTalk recognizes that the driver wants to change lane, it sends out the message containing the related left-back or right-back self image as well as the third-party VID which is detected by both the ego-vehicle and the vehicle moving parallel to the ego-vehicle in the target lane. In this way, DriverTalk helps the driver to convey the lane changing intent to all the possibly affected vehicles. For example, in Fig. 3.3, if the ego-vehicle wants to change to the left adjacent lane, both the vehicles at position ④ and ⑥ will receive the notification. Although the vehicle corresponding to the third-party VID (① or ②) also receives this notification, the VID implies that the lane-changing vehicle is behind it, it could easily ignore this message.

DriverTalk allows driver to pose questions to other drivers. The questions usually don't go to a particular driver, instead they are targeted at everyone around. The questioner only needs to identify itself by using its self image as VID in the question message.

Upon receiving a question message, DriverTalk saves the received VID temporarily. Here, from the driver's perspective, identifying who is asking the question is not critical. The driver who knows the answer does not necessarily see the questioner. For example, the questioner is moving behind the one who knows the answer, in this case, the answerer might not see the questioner, but she could still help the questioner.

When answering a question, the answer should go to the questioner, so the VID received in the corresponding question message is used to represent the receiver of the answer message (i.e. the vehicle which asked that question).

When the questioner receives the answer message and matches the VID in the answer message with its self image, it could conclude that someone answers her question and hence extract the answer.

The specification of the message exchanged within DriverTalk is exhibited in Table 3.3.

Table 3.3: DriverTalk Message

Field	Remark
VID	Selected image used to identify sender or receiver.
VID Type	Tell the receiver what the VID is. It could be: • Self image of sender • Detected vehicle image of sender • Detected third-party vehicle image of sender • Image of the questioner
Relative Lane Position	The relative lane position between sender and VID related vehicle if VID is a detected vehicle image.
Timestamp	Time of this message.
Level	The level of importance and criticality of the message: • Warning • Notification • Q & A
Message Type	Information type in this message: • P: Pre-defined message • Q: Question • A: Answer
Message Index	Index of pre-defined message information. Only used when *Message Type* = *P*.
Direction	Which direction the message is targeted at: • A: All vehicles moving around • F: Vehicle moving in front on the same lane • B: Vehicle moving behind on the same lane • L: Vehicle(s) moving on the left adjacent lane • R: Vehicle(s) moving on the right adjacent lane
Info	If *Message Type* = *Q* or *A*, here will be the text information used to convey the content of question or answer.

C. Inference of Sender's Position

When DriverTalk receives a message from someone else moving around, it needs to identify the relative position (left, right, front, behind, etc.) of the sender and tell its driver where the peer is. Especially in case the message conveys information related to some maneuver intent of the sender, e.g. the sender wants to change lane, it would be better for the receiver to be aware of the relative position of the sender in order to avoid hazards.

Each time DriverTalk receives a message, it compares the VID in the message with its self images as well as the vehicle images currently being detected in its field of view. But even when the senders are at different positions, they could possibly use similar VID. For example, when both vehicles at position ⑥ and ⑦ talk to the ego-vehicle, they will both use the detected vehicle images of the ego-vehicle as the VID in the message. These two vehicles are both moving behind the ego-vehicle, so very likely they both could see the rear-end of the ego-vehicle, i.e. their visual IDs both contain the rear-end part of the ego-vehicle. The only difference is that one vehicle is moving on the same lane as the ego-vehicle, the other is on different lane. Therefore one VID contains only the rear-end of the ego-vehicle, while the other VID contains the rear-end part from a different angle and it also contains the left side of the ego-vehicle (just like the third images of the first row in Fig. 2.8).

Image matching helps to distinguish the direction of sender. When DriverTalk receives a message and the VID is the image the sender detected for it, it matches the VID with its self images at different directions. The self image which has the most matched points with the received VID tells the direction of the sender. For example, if the rear-end self image matches more with the received VID than self images at other directions, the sender must be moving behind the ego-vehicle on the same lane (as exemplified in Fig. 2.8).

Based on which of its self images and the detected vehicle images matched with the received VID, DriverTalk infers where the sender is. The following Table 3.4 shows the relationship between the position of the sender and image matching result.

Besides, in the message, direction and relative lane position information between the detected VID and the ego-vehicle (i.e. the sender) is included. These information could also help the receiver infer the position of the sender. For example, if the message tells that the VID is on the left lane of the sender, the receiver only needs to find out where (i.e. in front, behind or parallel) the sender is on its

Table 3.4: Rules for Inferring Sender's Position

Matched Image Pair	Sender's Position
Received VID matched with detected vehicle image at position ① in Fig. 3.3	Position ①
Received VID matched with detected vehicle image at position ②	Position ②
Received VID matched with detected vehicle image at position ③	Position ③
Received VID matched with detected image at position ① or ②; VID Type = Detected third-party vehicle image	Position ④
Received VID matched with detected image at position ③ or ②; VID Type = Detected third-party vehicle image	Position ⑤
Received VID matched with the left-back self image	Position ⑥
Received VID matched with the rear-end self image	Position ⑦
Received VID matched with the right-back self image	Position ⑧

right lane.

With the help of relative lane position and direction information, the computational overhead on image matching is also reduced. Now instead of comparing the received VID with its self images and the detected vehicle images in all directions, the receiver only needs to check a subset of them in some particular direction.

A special case is the question and answer exchange. When asking a question, the questioner does not have a preferred receiver, her purpose is to get the answer, where the answer comes from is not critical. Similarly, the driver who receives a question does not care much about which driver is asking the question. Although DriverTalk could still infer the position of the sender asking the question, the importance of this information is low.

D. Message Playback

After receiving a message from other drivers, if it is a pre-defined one, DriverTalk fetches the complete pre-defined message content corresponding to the index received in the message; otherwise it extracts the talking content from the message itself, which is the content of question or answer. Besides, to make the receiver driver understand the message better, the inferred sender's position is also attached to augment the message. So the final talking content for playback is in the form like:

Sender's Position: Message Content

For instance, "The driver in front of you says: Don't tailgate me."

The talking content is in text format, DriverTalk uses Android text-to-speech engine to convert text back into voice and plays back to complete the whole talking between two drivers in a natural way.

E. Avoiding Abuse

DriverTalk provides drivers opportunities to talk to each other during driving. It benefits drivers, but it is also possible that some unruly drivers abuse this system. In DriverTalk, several mechanisms are adopted to prevent abuse.

1) Regulate the way driver talks: With the definition of a set of keywords and pre-defined messages, DriverTalk regulates how and what the drivers could talk.

2) Set different levels for message: We define 3 levels for the importance and criticality of the messages (see Table 3.3). The first level is for warning. Every driver should be aware of it because this level of message is related to immediate safety. For example, someone's car has a problem it might lose control or stop dead. The second level is for reminding. This level of message is not directly related to any immediate driving safety, but it is used to reduce the pressure from others or make the driving more comfortable. For example, asking the vehicle moving behind not to tailgate. The last level is for question and answer.

In DriverTalk, the first level of messages are always enabled, but we allow drivers to configure for the other two levels. The drivers could decide whether to send/receive messages at the other two levels or not. The system filters out the messages correspondingly.

3) Regulate message frequency: DriverTalk defines maximal times of messages each DriverTalk-enabled vehicle could send within a certain period. In addition, the system also allows drivers to configure the maximal message frequencies they are happy to receive from other drivers. Besides, between a particular sender and a particular receiver, DriverTalk also regulates the message frequency. For example, when DriverTalk detects that messages are being received too frequently from a particular counterpart, it automatically filters to reduce the messages being played back to its driver.

3.4 Evaluation

To make DriverTalk a usable system, the following aspects should be studied.

First, DriverTalk should correctly convey message from sender to the target driver. When a driver talks to some other driver, the target driver should correctly recognize it and others should not be confused.

Second, the system uses peer-to-peer wireless communication between vehicles to enable talking. We need to study the feasibility of the vehicular communication at some certain traffic situations to make sure that the talking message could be exchanged reliably.

Last, to make the talking experience smooth and close to the way in real life, the whole system should work very efficiently.

We have verified in OmniView that with properly selected threshold for matched points, image matching algorithm could confidently recognize images for the same vehicle and distinguish images for different vehicles. Correspondingly, DriverTalk could correctly recognize the sender and receiver of each message, i.e. who is talking to whom. As a talking system, here below we focus on the evaluation of message exchange related performance as well as efficiency of DriverTalk.

A. Simulation of Vehicular Communication

In DriverTalk system, vehicles need to transmit VID embedded message. As mentioned before, the sizes of images are controlled in the range of 10$\sim$14 KB, which is friendly for communication. Meanwhile these image sizes allow the system to identify vehicles in images correctly. Although we need to add other information into message, such as VID type, message type as shown in Table 3.3, the size of those information is negligible compared with VID, so they are ignored in evaluating the communication performance.

To learn about the performance of the DriverTalk system in large scale deployment, we use simulation to study the vehicular communication among DriverTalk-enabled vehicles. We use SUMO [29] to simulate the vehicular movement and NS2 [30] to carry out the network communication.

We simulated two different layouts, one is highway, the other is city environment. For the highway traffic, we also studied with two different traffic modes. One is dense mode, in which the traffic flow is relatively heavy and talking would happen more frequently, thereby the communication between vehicles will be more intensive. The second is sparse mode, in which every vehicle has fewer neighbors than dense mode and the message transmission in the network will be less.

We simulated three different transmission ranges: 60 m, 80 m and 100 m. Although the speed of vehicles on highway is high, the relative speed between vehicles is low. Once two vehicles meet, they will at least stay within the communication range of each other for a while (i.e. be able to talk with each other).

For the city scenario, we injected 1200 vehicles into an 1km x 1km Manhattan-like area (as shown in Fig. 3.5 (left)). The 1200 vehicles are randomly moving around in this area during the simulation period. Besides, traffic lights are deployed at each intersection regulating the traffic flow, which lead to the pile-up of vehicles (see Fig. 3.5 (right)). The vehicles are moving at lower speed in a city environment

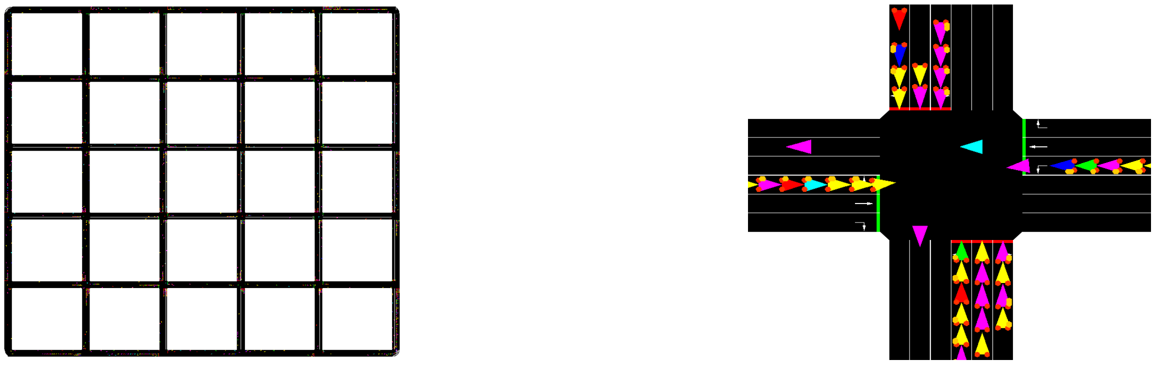


Figure 3.5: (Left) The city layout in the simulation. It is an 1km x 1km area (each segment is 200 meters). (Right) The detail of an intersection in the city layout. Each intersection has traffic lights enabled, the vehicles (here the triangle-like icons stand for vehicles) pile up there.

than highway, so we choose shorter transmission ranges: 40 m, 60 m and 80 m.

In DriverTalk system, talking could be in a unicast or multicast way, i.e. a driver talks to a particular driver (e.g. the driver moving in front on the same lane), or talk to two drivers (e.g. when changing lane, should notify the two drivers at position ④ and ⑥). Talking could also be broadcast, i.e. a driver talks to everyone moving around (e.g. asking question). We simulate these kinds of talking scenarios for each sender by randomly choosing target receivers moving around the sender. In each scenario, different number of images should be sent. We believe that in each talking, up to 4 images are enough. Four images are only needed when a driver tells others that something is wrong with her vehicle (i.e. pre-defined message information 7 in Table 3.1). If the vehicle has problem, there are cases that it is out of control or could not stop, then the vehicles in front of it or moving parallel will be potentially collided; or the vehicle might not be able to move, then the vehicle behind it will be affected. When the driver sends out this message, DriverTalk will broadcast the message with the VID of its rear-end self image as well as the three detected images of the vehicles in front of it (i.e. VIDs of vehicles at position ①②③).

The detailed setting in our simulation is listed in Table 3.5. In our simulation, to stress test the system, we assume every driver will talk once every 60±30 seconds.

Table 3.5: Setting of Simulation for DriverTalk Vehicular Communication

Parameter	Value
Simulation Period	Highway: 1800 s; City: 400 s
Number of Vehicles	Highway: 400 (8 types of vehicles with different length,speed,acceleration and deceleration; vehicles' speeds are dynamically changing.)
	City: 1200 (10 types of vehicles used.)
Speed	Highway: 22∼36 m/s (50∼80 mph)
	City: ∼16 m/s (35 mph)
Traffic Density	Highway: Sparse (57 vehicles/km); Dense (110 vehicles/km)
	City: 1200 vehicles cycle around.
Wireless Protocol	DSRC 802.11p
Antenna Type	OmniAntenna
Propagation Model	Two Ray Ground
Data Rate	6 Mbps (QPSK) DSRC could support up to 27 Mbps data rate, but 6 Mbps is the optimal data rate which provides good performance [21].
Message Size	Each VID is 10/12/14 KB, which is sliced into 1-KB small packets; size of other information in message is too small and ignored.
Message Life Time	2 s
Transmission Method	Broadcast
Transmission Frequency	1∼4 images every 60±30 s
Transmission Range	Highway: 60, 80, 100 meters
	City: 40, 60, 80 meters

We measure the message reception rate to see how reliably the system could exchange message from sender to targeted receivers. In unicast/multicast talking, we check whether the particularly targeted driver(s) could receive the message. In broadcast talking, we calculate the probability of every potentially targeted driver receiving the message.

The overall message reception rate for highway and city scenarios are shown in Fig. 3.6. On highway, DriverTalk can achieve about 97% reception rate both in dense mode and sparse mode. In city environment, due to the pile-up of vehicles at intersections, the reception rate decreases slightly to 95%. From this evaluation,

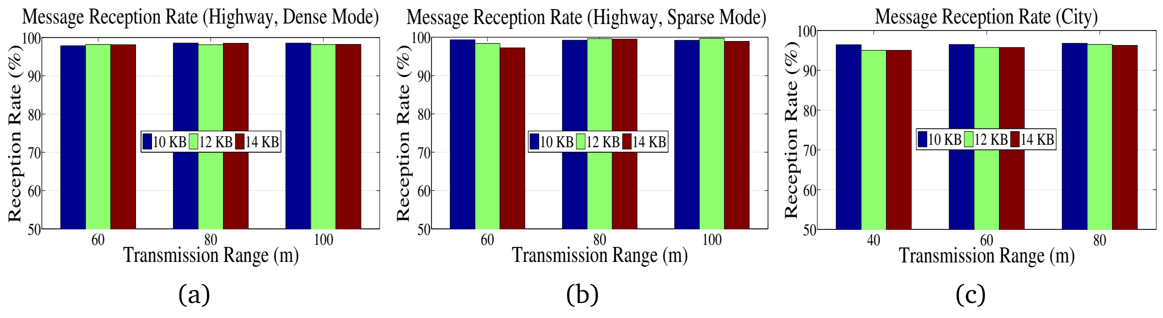


Figure 3.6: (a) Message reception rate in dense highway traffic (three types of image sizes and transmission ranges are compared). (b) Message reception rate in sparse highway traffic. (c) Message reception rate in city.

we believe DriverTalk could reliably deliver what a driver says to the target driver.

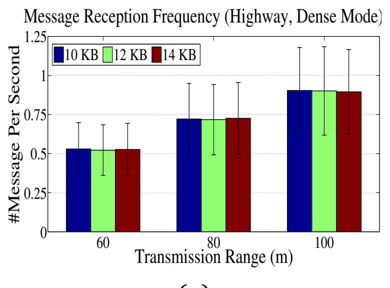
B. Efficiency

As a talking system, to provide good user experience, DriverTalk should perform efficiently, i.e. drivers should be able to talk via the system smoothly, like the way they talk in real life. We need to study the end-to-end latency of the whole system from a sender driver says something to the receiver driver hears it.

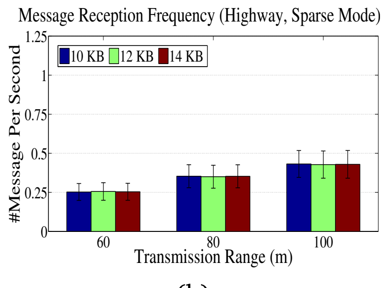
In the DriverTalk system, time is mainly consumed in the following steps: vehicle detection, message transmission, visual ID recognition, speech and text conversion. In this section, we measure the times consumed in each of these steps to give an idea about the end-to-end delay of talking.

DriverTalk uses the same classifier for vehicle detection, as measured in OmniView, it could detect vehicles in its view within 97 ± 17 ms (measured on a Galaxy S4).

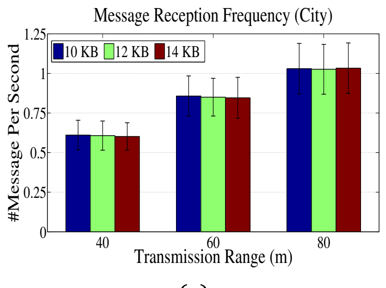
Upon receiving a message, the system needs to identify whether itself is the target receiver by checking the VID in the message. To do this, the system compares it with its self-images and the currently detected vehicle images. As measured in OmniView, on average, the task of one-time image matching could complete within 108±44 ms when experimented with Galaxy S4. Therefore in the worst case, the system may need about 650 ms on matching with a received VID.



(a)



(b)



(c)

Figure 3.7: The number of messages each DriverTalk-enabled vehicle receives every second in dense highway traffic (a), sparse highway traffic (b) and in city (c).

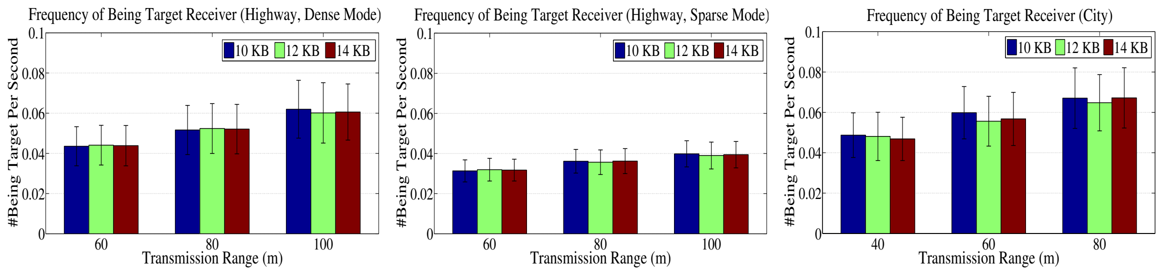
To estimate the total effort on visual ID recognition, i.e. how many times every DriverTalk-enabled vehicle needs to do image matching, we estimate the number of messages each one will receive, which corresponds to the number of received visual IDs need to be checked. Fig. 3.7 illustrates the message reception frequency. In most of these settings, on average, every vehicle will receive no more than one message per second. Therefore the overhead of computation and time on visual ID recognition is acceptable, which allows the smartphone to carry out other tasks .

Clearly, larger transmission range incurs more message reception. Comparing the same transmission range on highway and in city, due to the traffic lights and congestion, vehicles will pile up at intersections or on the road, which makes the vehicles in city receive more messages than on highway. In a real deployment, we can choose a short transmission range (e.g. 60 meters for highway, 40 meters for city) by tuning the power to allow the reception of messages only from the drivers in short distances. The interaction with the drivers moving closely is more important than with those moving far away considering the safety and driving experience to the ego-driver.

As mentioned before, with the knowledge of the relative lane position and direction information from the message, in many cases DriverTalk does not need to compare all its self-images and detected vehicle images with the received VID, the actual time on image matching will be much less.

Besides, smartphones's CPU is becoming more and more powerful. Many of them now have multi-cores, the image matching task could be carried out in parallel on multiple cores, thereby the time on image matching will become even less along with the emergence of new-model smartphones.

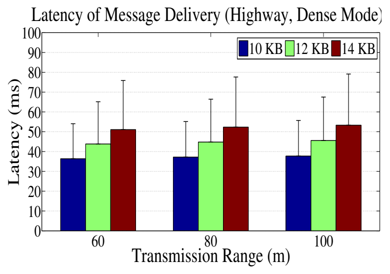
Although the visual ID recognition task will not occupy much of the running time, DriverTalk should allow time for the message to be played back. Here we check how often every participant is chosen as the target driver by others. Fig. 3.8 illustrates the frequency of each vehicle being the target receiver. It shows that on average, about every 12$\sim$38 seconds across all transmission ranges, one driver will become the talking target of some other drivers. Considering the normal speed of talking, people could easily speak 20$\sim$30 words within 10 seconds, providing sufficient time for a message to be played back before another message could arrive. Furthermore, the system could adjust the speed of playing back, which makes sure the talking could finish in time.



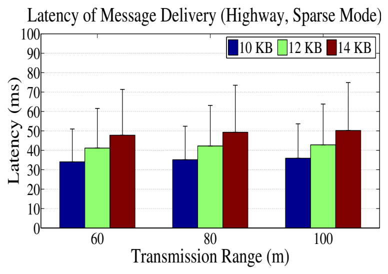
(a) (b) (c)

Figure 3.8: The frequency of each DriverTalk-enabled vehicle is chosen as target receiver by others in dense highway traffic (a), sparse highway traffic (b) and in city (c).

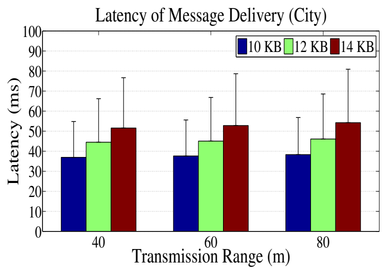
From the simulation, we also measured the end-to-end latency for message delivery. Fig. 3.9 shows the latency in both highway and city scenarios. In all cases, message could be delivered within 55 ms on average.



(a)



(b)



(c)

Figure 3.9: Latency of messages in dense highway traffic (a), sparse highway traffic (b) and in city (c).

In DriverTalk, at the sender and receiver sides, the system needs to convert the talking content between speech and text. We measured the efficiency of Speech to Text and Text to Speech conversion. For Speech to Text, we measured the time from the end of speech to the completion of conversion into text. For Text to Speech, we measured the time from the end of the acquisition of text to the completion of the conversion into voice. The time of recording voice during speech-to-text conversion and the time of playing back is not counted, because that time depends on what user speaks and how fast the user is speaking.

Now all the popular smartphone platforms provide speech-to-text service and text-to-speech engine. Android has embedded speech-to-text service and most Android phones have text-to-speech engine installed. Text-to-speech usually could work offline on Android. Speech-to-text could work both in offline and online modes. When it works in offline mode, it responds quickly. When working with a server in the cloud, the response time of speech-to-text will be larger than the offline mode, but its accuracy on recognition will be higher.

We drove around in downtown as well as on highway to measure the time[3] of speech-to-text conversion. The sentences we used varied from one word to thirty words. We collected data with a Samsung Galaxy S4 with AT&T 4G LTE enabled.

[3]The accuracy of speech recognition is not our focus, which varies from person to person.

The Table 3.6 lists the time of speech & text conversion. We can see that the conversion is very efficient.

Table 3.6: Time of Speech & Text Conversion

Speech $\rightarrow$ Text	Avg. 251 ms (95th percentile: 554 ms)
Text $\rightarrow$ Speech	Avg. 12 ms (95th percentile: 22 ms)

By summarizing all the above measurements, we can expect that in DriverTalk, when a driver says something, the target driver could start hearing it within 1 second, which could guarantee smooth user experience with this talking system.

3.5 Related Work

In both academia and industry, vehicle related systems are being developed to improve driving experience and safety.

Ford is known to be working on a “Talking Cars" project [43], which allows cars to talk to each other and expect this could help cars avoid crashes and reduce fuel consumption. Drivers can be alerted to potential safety hazards using loud noise and flashing lights. Yet the detail about how it works and when the system will be available is not clear.

CarSpeak [44] is a vehicle-to-vehicle collaborative system. Each CarSpeak-enabled vehicle senses the environment along the road. Every participant could query and access information captured by others for interested regions. In this way, the driver could know about the obstacle, pedestrians in the region which is out of her line of sight, which helps to improve safety.

RoadSpeak [45] and Social Vehicle Navigation [46] are two client-server based vehicular social network systems which allow drivers to do voice chatting during their commuting. Drivers join certain voice chatting groups based on interests, timely location [45] or routes [46]. The participants could talk on common topics and report road situations. Although these systems allow drivers to talk to each other, all the messages are aggregated by central servers. A driver could not identify any particular neighbors to have in-situ talking.

Authors in [47] proposed a GPS, CAN, radar and camera based cooperative safety system, which tries to infer drivers' intents. It expects to allow the participants exchange intents via V2V and V2I communication. This work only analyzes the potential benefit in reducing collision, how accurately the system could infer driver's intent is unknown. Due to the GPS error and camera view range, it is hard for a driver in the proposed system to convey intent to another particular driver. In contrast, our system allows drivers to directly talk and uses visual IDs to identify particular drivers moving around.

FleaNet [48] is also a DSRC-based vehicular network which sets up a virtual market for vehicles to exchange information. Each vehicle and road-side store in the system could sell and buy goods. Each participant tells what goods she possesses or seeks by broadcasting query into a DSRC-based ad-hoc vehicular network. The vehicles which receive queries help to disseminate the queries to the entire network. Once a node receives queries from others, it tries to resolve the queries in its local database. If a match exists between queries (i.e. two queries contain common interest), it will notifies the buyer. The buyer could contact the seller to make a transaction.

Different from all existing systems, DriverTalk aims to improve safety and driving experience by allowing drivers to talk to each other directly and in a more natural way. To the best of our knowledge, DriverTalk is the first system that enables in-situ talking between unacquainted drivers. As correlated systems, OmniView and DriverTalk offer complementary services and work in synergy to assist drivers.

3.6 Limitations and Future Work

As a next step, we plan to refine DriverTalk by tackling more particular cases, such as design a mechanism to allow a driver to selectively answer one of several question messages arrived in a very short time.

The second work we will do is studying the characteristics of feature points in vehicle images. If we could find some feature points which could uniquely represent a vehicle, instead of sending the whole vehicle image, we may only send the feature points, which could potentially reduce the overhead of communication and visual ID recognition.

We also plan to have a real deployment with some after-market DSRC component and arrange several vehicles to do real talking when moving both on highway and in city.

3.7 Summary

In this work, we explored the feasibility of enabling targeted communication between drivers, as a better alternative to the existing modes of coarse-grain signaling through horns and lights. Towards that end, we proposed a smartphone-based system, called DriverTalk, that allows neighboring drivers on the road, with no prior acquaintance, to talk to each other. We have presented the details of DriverTalk system, specifically, how it identifies the senders and receivers of messages in different communication scenarios, utilizing appearance images of vehicles as their Visual IDs. We have evaluated the system by simulating the traffic in both highway and city scenarios and shown that it could perform efficiently and reliably.

Part II

Improving Experience of Bus-riding

Imagine a transportation system in which passengers simply get on/off without any explicit ticketing operation. Yet, the system tracks usage and charges passengers. Such a system will not only be more convenient and efficient, but also be more conducive for analytics, than existing systems. Towards that goal, we exploit the opportunity that people are carrying sensor-equipped smart devices (e.g. smartphone, smartwatch), and their motion trajectories/patterns and experienced environment can be measured continuously. Assuming that vehicles are also equipped with such sensors (perhaps fixed devices or smart devices carried by drivers), the vehicles' motion and the experienced environment characteristics can also be recorded and uploaded to cloud. Under these assumptions, we hypothesize that the motion/environment sensed by a passenger's smart device correlates strongly with that of the vehicle she is traveling in and is distinct from that of other vehicles and/or other traces of the same vehicle. In this part, we expand on this intuition and develop a system, called RideSense, that matches a passenger's sensor trace against the traces of buses in that area, to determine which bus, when she has taken and where she gets on/off. This work offers confidence that ticketless public transportation may indeed be a possibility in smart cities of the future.

Chapter 4

RideSense: Towards Ticketless Transportation

In current public transit system, there are two major ways for paying the fare. One is paying by cash. Passengers need to prepare and carry cash with them and pay the fare when they get on a bus. A more convenient way is IC-card based ticketing system. A passenger obtains a physical IC card from transit system operator and deposits fees in it. Later when she gets on a bus, she inserts or scans the IC card to pay. Besides, in recent years, we are witnessing the emergence of smart-device-based e-payment in public transit. For example, smartphone users in Japan use Felica [49]-embedded smartphones to take bus and subway. Apple Pay is now being used in London buses [50]. Both these are NFC-based solutions; passengers tap their phones (which contain NFC chip) on readers and go.

While the existing pricing, ticketing and billing process in public transportation systems generates significant revenue, it does not come free of financial and experiential hurdles. For instance, installing, upgrading, and maintaining the end-to-end accounting infrastructure (including ticketing kiosks, manned booths, ticket-checking gateways, card readers, etc.) require substantial investment. The core notion of buying tickets, at the right price, for the right destination, with the right monetary change, and just in time to catch the train, is often a source of frustration/anxiety, making the overall experience less seamless. Finally, ticketing often creates queues at purchase kiosks and at bus stations, because every passenger boarding the bus needs to pay for (or verify) her ticket. In sum, the process of gathering revenue in public transportation systems imposes operational burdens, both to the operators and the customers. Apart from the above drawbacks, in current bus systems, the operators at most know where each passenger gets on the bus, but they have no knowledge or record of where a passenger gets off, making it hard to perform analytics such as determining the occupancy of each bus at each road segment.

To eliminate the operational burdens from both customers and operators, and also provide an insight about the running of the traffic system, we propose a smart-device-based system, RideSense. Our core idea is simple and exploits the opportunity that people are carrying sensor-equipped smart devices everywhere, and their motion trajectories/patterns and the environment they experience can be measured continuously. Now, assuming that public vehicles can also be equipped with such sensors (perhaps installed explicitly or carried by the drivers), the vehicles' sensor data can also be recorded and uploaded to the cloud. Under these assumptions, we hypothesize that if Alice takes a bus, her sensor trace can be correlated against the sensor trace of the bus to precisely position her bus trip - which bus she takes, when she boards, where from and to she rides, etc. We envisage matching Alice's sensor data with the vehicle at fine granularities, including similar *pot hole jerks* that both Alice and the vehicle experienced, the *stops*, *turns* and the number of *lane changes*, precise times of *braking*, decreased *atmospheric pressure* due to increased altitude, etc. Fig. 4.1 illustrates RideSense.

In a RideSense-enabled bus, the reference device collects sensor data all the time when the bus is on duty. When the bus is off duty, the sensor traces are uploaded into cloud for matching. Passengers run a RideSense app in their smart devices and could take buses freely without any explicit interaction with any ticketing/billing facilities when they get on/off. RideSense utilizes the built-in sensors to record the motion and environment information (e.g. atmospheric pressure) along the passenger's trip. After the passenger gets off the bus, the app finds an opportunity

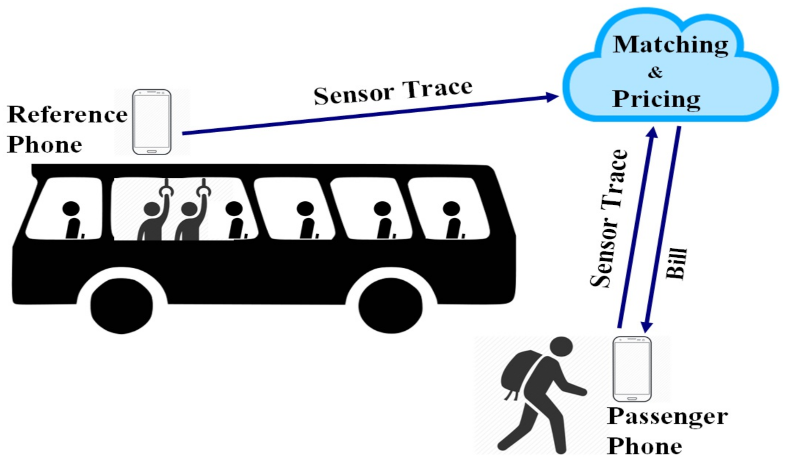


Figure 4.1: An illustration of the RideSense system. Sensors on passenger's phone and on the bus collect the sensor data along the travel; later these sensor traces are uploaded to cloud for matching. The system could learn which bus line a passenger took, when she boarded, and also where the passenger has traveled, therefore the operator could bill the passenger.

(for example, when the user is at home with WiFi connected and the phone is not busy) to upload the recorded sensor trace into cloud.

The cloud matches a passenger's sensor trace with reference trace to find out: *i)* **Which** bus line the passenger has taken. *ii)* **Where** the passenger boarded and disembarked, i.e. the source and destination stations of a passenger's trip. *iii)* **When** the passenger took the bus. We use *Which/Where/When* to refer to these three aspects in the rest of this paper.

Designed correctly, RideSense can bill the users accurately afterwards. It depends on how well our intuition holds, i.e. one sensor trace from the reference device in bus, the other from the passenger's smart device, should correlate strongly. As a preliminary check, we collected motion data from three passengers' phones each on a different bus. We gathered motion data from each bus too and matched them against the passengers' motion data. Fig. 4.2 shows that the highest correlation values are along the diagonal indicating strong correlation between motion of a passenger and her bus. Though this is a toy experiment, it does affirm the core intuition.

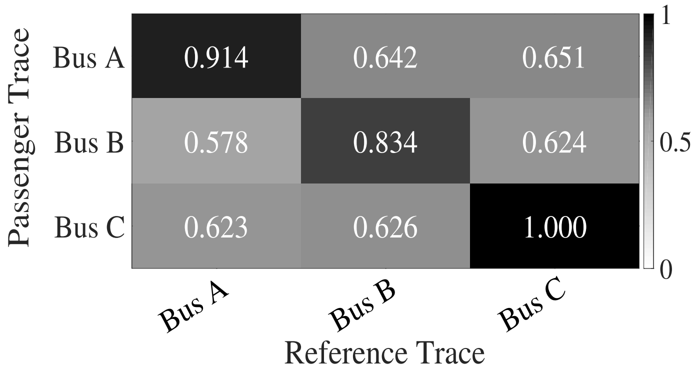


Figure 4.2: The sensor trace collected by a passenger phone shows higher correlation with the corresponding bus trace.

Apart from billing the users correctly, perhaps more importantly, the overall operation of the public transit system can become far more seamless. For example, users can board buses from any of the doors and conventional ticket verification facilities are no longer needed, which reduces traffic backlogs at the stations, and administrators can attain entire programmability on pricing and billing. Perhaps other opportunities will arise, given the disruptions in the transportation industry with Uber-like services becoming popular. Finally, the data from the vehicles and users can be amenable to valuable analytics, offering insights into city planning, human mobility models, traffic control, pricing, etc. Of course, realizing such a vision will entail a variety of challenges, including sensor data processing, mechanisms to thwart cheating, location privacy for users, appropriate user interfaces, policies, etc. The tangible outcome of this research is expected to be a convincing, data-driven argument on the viability/practicality of this vision.

Currently, we target RideSense at bus system and assume that each bus will be fitted with a device (e.g. smartphone) for sensing motion and environment. We focus on implementing and evaluating the algorithms for matching passengers' sensor traces against the traces of the buses in that region, to study how accurately RideSense could tell the Which/Where/When information about passengers' bus trips. Considering that passengers' sensor traces are their tickets, we need to be concerned about the potential for cheating and tampering. Also, users' normal usage of the smart devices should not adversely affect the RideSense accuracy. While there are several such important issues that need to be tackled prior to deployment of RideSense, in this work, we assume a non-malicious passenger and conduct a study to assess the feasibility of RideSense.

Before we proceed with system design, it is important to clarify a question one may have: Why not match the GPS data from the user's phone with that of buses in that region? A major limitation of this GPS-based approach is that users are unlikely to always turn on the power-hungry GPS. Furthermore, if another vehicle also takes the same route between "source" and "destination" at that time, GPS-based approach can not correctly place the passenger on the right bus. The error of GPS readings in urban area (where tall buildings and/or tunnels exist) could also make the approach infeasible. On the other hand, RideSense can offer better overall performance using cheaper sensors (such as accelerometer, gyroscope, and barometer) on a smart device.

4.1 System Design

We now present the design of RideSense system. As mentioned earlier, RideSense app in passengers' smart devices collect readings from accelerometer, gyroscope and barometer. We refer to this sensor data as passenger trace. A smart device plugged in each bus records readings from GPS too in addition to the above set of sensors. We refer to this sensor data as reference bus trace. Considering that GPS coordinates of bus stations could be obtained in advance, we can extract the reference bus trace for each segment (see Fig. 4.3 for terminology). The objective of RideSense is to identify the sequence of bus segments whose sensor trace matches closely with the passenger sensor trace.

Note that passenger trace may include data from other daily activities, as the

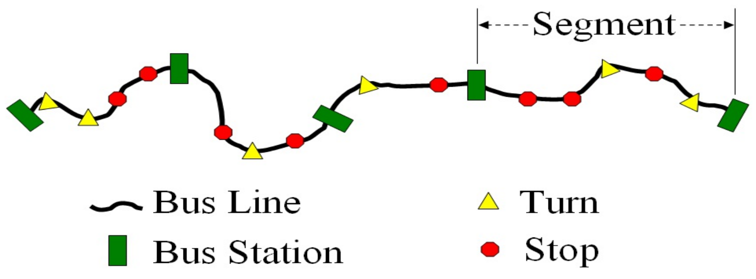


Figure 4.3: A bus line is made up of segments, which are delimited by stations. Stations are the regulated places where the bus stops and allows passengers to get on/off. When a bus is moving on the road, it could experience many stops and turns.

user may have the app on even when she is not traveling on the bus. We need to make sure a passenger's smart device will only upload the sensor trace related to her travel in bus system. Although RideSense could require passengers to explicitly turn on the app only when they get on a bus and turn off immediately after they get off a bus, this might sacrifice some convenience and people might forget to turn on or off. To provide the best user experience, user only needs to turn on the app once and does nothing else. So the system needs to distinguish a user's bus traveling from all other daily activities. Lots of solutions [51–56] exist which could distinguish vehicle transportation from other daily activities through smartphone sensors. Even for vehicle transportation, there are smartphone-based solutions [53] [55] to distinguish bus from car. We borrow the result from the existing approaches on distinguishing bus traveling from other activities.

Given a passenger trace, RideSense only knows it starts and ends at stations, but does not know the number of stations/segments in-between. To find the corresponding reference bus trace, RideSense needs to compare all possible combinations of consecutive segments from all bus lines. Considering the number of bus lines and passengers in a city, enormous number of reference and passenger traces are produced everyday, RideSense needs to search a huge space to match a given passenger trace. Fig. 4.4 shows the pipeline of the matching algorithm of RideSense

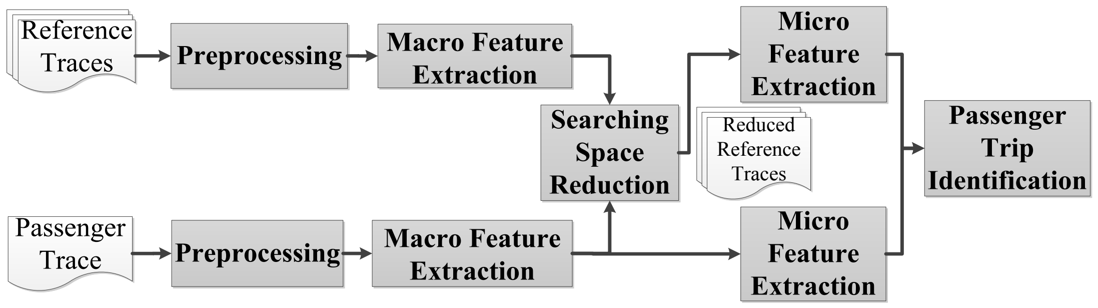


Figure 4.4: RideSense extracts macro features from traces after preprocessing. Based on macro features, the searching space is reduced for each passenger trace. The system uses micro features to do trace matching to identify a passenger's trip.

in cloud. The system extracts macro features and micro features from both reference and passenger traces. Macro features are the basic properties of the trace, such as turn, stop information, which are used to filter out reference traces and reduce the searching space for each passenger trace. Micro features are the fine-grained characteristics of the motion and atmospheric pressure information of the travel, which are used for trace matching once the searching space is reduced. We elaborate on the pipeline steps below.

A. Data Preprocessing

The data collected with smartphone's built-in sensors, i.e. accelerometer, gyroscope and barometer, is very noisy. Before extracting features from these sensor data, RideSense goes through a preprocessing stage to clean the data, which contains two steps: smoothness and interpolation.

We run an exponential moving average on the raw sensor data to remove noise. Due to the low-quality of smartphone sensors and also the limitation of operation system, the collected sensor data does not have fixed intervals between samples. We use cubic spline interpolation [57] to interpolate the sample data and then resample to obtain sample data with needed sample rate.

B. Searching Space Reduction

Searching space reduction is carried out based on macro features. Macro features define the basic properties of a bus line and its segments. To reduce the searching space of reference traces for passenger traces, RideSense needs to extract macro features both from passenger traces and reference traces. By comparing the macro features of both sides, the reference traces which show irrelevant properties to the passenger traces could be eliminated from the candidate set.

In RideSense, following macro features are extracted and used: cellular network information, duration of travel, turn information, altitude information. We present the details of each macro feature, its usage and the way to extract in following paragraphs.

Cellular Network Cellular network contains lots of cellular towers, which are distributed to provide good coverage. Each tower provides service to a limited area. A bus usually travels through many cells of the network, which are covered by different towers. Every cellular tower has a unique ID, which could be detected by the phone. By using the cellular network ID, RideSense could limit the reference traces for a passenger trace to a certain area, the searching space could be significantly reduced.

Duration of Travel Both the reference traces and passenger traces are formed by segments which are delimited by stations. In the reference trace, by knowing the GPS coordinates of each bus station, we could easily figure out the duration at each station and also the duration between two consecutive stations. In passenger trace, RideSense does not have GPS information, so it doesn't know how many stations/segments a trace covers. Fortunately, the places where a passenger gets on and gets off are stations inherently. Furthermore, the segments and stations covered by a passenger trace are continuous. Therefore, the end-to-end duration of a passenger trace should correspond to the duration from one station to another station (including the inbetween segments and other stations), although RideSense does not know exactly which stations they are.

To find out the corresponding reference sub-trace for a passenger trace (a passenger usually only travels a part of a bus line), RideSense only needs to search sub-traces which start and terminate at stations. It does not need to search the reference traces in the middle of a segment. For a passenger trace, if the duration of a sub-trace in reference space is far beyond the duration of the passenger trace, it could not be the reference trace corresponding to the passenger trace.

From motion's perspective, every trace is formed by stop and move. So an end-to-end duration of a trip is composed of many stop-durations and move-durations. Different bus lines and even different traces of the same bus line exhibit different stops and moves. Thereby the proportion of stop and move durations within an end-to-end duration could also be macro features for a trace.

In reference trace, the stop and move could easily be identified by the speed measured by GPS. While in passenger trace, no GPS speed information is available. We developed a stop detector to identify the stop and move. After running the stop detector, we summarize the stop and move proportion of a passenger trace as macro features.

Stop Detection: The stop detection in RideSense is based on accelerometer and gyroscope readings. We run a sliding window with 1-second length and 0.5-second step size on the sensor data to extract features (as listed in Table. 4.1) for stop detection. These features are selected experimentally. We trained a random forest which contains 50 decision trees for stop detection.

Table 4.1: Features for Stop Detection

Sensor	Feature
Accelerometer	*Time Domain*: (Magnitude of Linear Acceleration) Mean, Median, Variance, Standard Deviation
	Frequency Domain: (Magnitude of Linear Acceleration) Maximal amplitude, Energy, Mean coefficient magnitude,Root Mean Square of bucket 0$\sim$20 Hz
Gyroscope	*Time Domain*: (Magnitude of Gyroscope Readings) Mean, Median, Variance, Standard Deviation
	Frequency Domain: (Magnitude of Gyroscope Readings) Maximal amplitude, Energy, Mean coefficient magnitude, Root Mean Square of bucket 0$\sim$20 Hz

Turn Each bus line and its segments have different numbers of turns. A turn could be detected by continuously integrating the gyroscope readings over a window. We experimentally select 6 seconds as the window length for a turn. If it is a turn, the bus should be able to complete a continuous movement within the window and experience large enough turn degrees. Although gyroscope suffers from the problem of drifting [58], which usually leads to problems with a long duration. In a 6-second window, it is reliable to tell it is a turn or not from the integrated gyroscope readings.

Turns have different degrees and directions. To compare turn degree accurately, the passenger phone and reference phone must be aligned accurately. Reference phone could be fixed in a bus, but passenger phone has unlimited flexibility, which makes it less reliable in turn degree comparison.

When a bus makes a turn, it could have one of the two directions: left and right. A bus turns around its Z-axis (as shown in Fig. 4.5 (b)). The Z-axis of bus aligns with the Z-axis of the earth coordinate system (as shown in Fig. 4.5 (a)).

While the passenger and reference phones could be in any attitude, their coordinate systems do not align with the earth coordinate system (as shown in Fig. 4.5 (c)). In order to measure the turn around the earth's Z-axis, the smartphone's Z-axis and the Z-axis of bus (i.e. Z-axis in earth coordinate system) must be aligned.

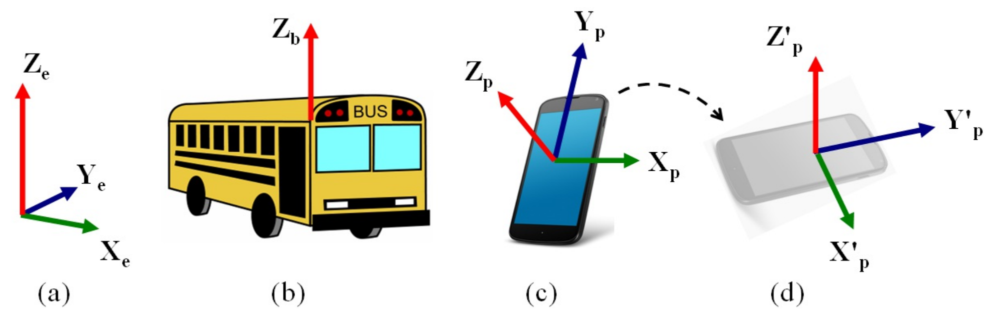


Figure 4.5: (a) Earth coordinate system. Smartphone's coordinate system (c) usually does not align with bus coordinate system (a, b). To learn the turn direction, their Z axes must be aligned. RideSense utilizes the accelerometer readings when the phone is in static status to measure the relative attitude between smartphone and the earth coordinate system, and then a rotation is applied to align smartphone's Z-axis with the Z-axis in earth coordinate system (d).

To align the passenger phone's Z-axis with bus, RideSense needs to do a rotation on the smartphone's gyroscope readings, which could eliminate the relative difference between the two Z axes. The relative difference could be learned from accelerometer when the passenger phone is in a static status. If a phone is static, what the accelerometer measures is only the gravity. The gravity is distributed on its 3 axes, which tells the attitude of the phone in the earth coordinate system.

For reference trace, it is easy to find the static status from the GPS information, where the GPS speed is shown as 0. It is more difficult to identify the static status in passenger trace, which does not include GPS information. Although we could use stop detector to identify stops, which is unlikely 100% accurate in recognizing all stops. To reduce the impact of error propagation caused by stop detector, RideSense uses the accelerometer readings in a short period when the stop detector reports stop continuously for 3 times. Once 3 consecutive stops are reported, it is safer to conclude that the passenger phone is static, then RideSense learns the distribution of acceleration (i.e. gravity) on the 3 axes to obtain the attitude of the passenger phone.

From the attitude value, RideSense derives a rotation matrix, which is used to rotate the Z-axis in smartphone's sensor readings into the Z-axis of the earth coordinate system. Therefore, the system could measure the rotation around earth's Z-axis, i.e. how the bus is turning.

To tell the turn direction, RideSense only needs to look at the gyroscope readings on Z-axis. If the gyroscope reading on Z-axis is positive, the bus is turning left, otherwise it is turning right. We use a voting mechanism within the turn window to decide whether the gyroscope reading is positive or not.

In RideSense, we use the number of turns and turn directions as macro features.

Altitude RideSense collects barometer readings which correspond to the altitude of the road. But barometer is easily affected by climate factors, such as temperature, humidity. It could only reflect relative relationship in altitude, e.g. place A is higher than place B. RideSense utilizes barometer to eliminate candidate reference traces for a given passenger trace. For instance, if the barometer reading at source station of a passenger trace is higher (with a threshold) than the destination station, the reference sub-trace which has opposite relationship in barometer readings between its source and destination stations could be ignored.

After reducing the searching space based on the macro features, the remaining reference traces are further compared with the passenger traces based on micro features, which is explained in next section.

C. Passenger Trip Identification

When a passenger takes a bus, the passenger phone should experience same or similar motion and environment as the reference device in the bus. RideSense characterizes the motion through micro features extracted from motion sensor data. Besides, the system also extracts micro features from the barometer readings which represent the atmospheric pressure along the trip. By comparing a sequence of micro features of passenger trace with a sequence of micro features extracted from reference trace, RideSense finds out the reference trace which corresponds to a passenger's travel.

When the reference phone and passenger phone are in static status, the trace data does not contain meaningful and comparable motion characteristics, which also wastes resource on computation. Therefore RideSense extracts micro features only from the sensor readings when the phone is experiencing movement.

The system extracts micro features from accelerometer, gyroscope and barometer both in time and frequency domain. In feature extraction, a 1-second sliding window is applied on sensor data, which moves every 0.5 seconds. The sequence of micro features of reference traces are delimited by stations. RideSense Z-normalizes the micro features and uses Dynamic Time Warping to do matching. In this system, normalized DTW distance is used to represent the similarity between traces. The reference trace which achieves minimal normalized DTW distance to the passenger trace is regarded as the corresponding reference trace.

To learn the effectiveness of the features, we collected experimental sensor traces along some randomly selected bus lines in our area, and then we carried out the study with each single feature on the experimental traces. Base on the experiment result and also taking the cost of computation into consideration, following micro features are finally used in RideSense for identifying detailed information about which/where/when of passenger trips:

Table 4.2: Micro Features for Passenger Trip Identification

Sensor	Feature (all based on the magnitude, time domain)
Motion Sensor	Median, Root Mean Square of Linear Acceleration; Log Energy of Gyroscope Readings
Barometer	Mean, Median, Variance, Standard Deviation, Range, Mean Crossing Rate, Mean Absolute Deviation, Skew, Root Mean Square, Signal Magnitude Area

4.2 Evaluation

We evaluate RideSense with the public transit in our university area to study how well the system could identify the bus line/shift (*which/when*), source and destination stations (*where*), corresponding to a passenger's bus trip. Before presenting the detailed results, we summarize our findings below.

- RideSense can identify *which/when/where* about a passenger's travel with an overall accuracy of 85% using motion sensors and an accuracy of 91% based on barometer.
- It can determine the number of segments a passenger has travelled (fare for the bus-riding may depend on this) correctly in 96% and 99% instances with motion sensors and barometer respectively.
- The more segments a passenger travels, the higher the accuracy of RideSense in identifying her trip.
- Motion-sensor-based RideSense performs better when a passenger has the phone in his pocket than the phone in hand, whereas barometer-based version is irrelevant to phone positions.

A. Data Collection

We recruited two volunteers for data collection. The volunteers traveled on 5 bus lines, which have 5 to 8 bus stations (correspondingly, 4 to 7 segments). For each bus line, they rode 3 times. The volunteers spent 20+ hours and collected more than 30G sensor data. The data collection was carried out in different time of different days, including rush hours and other time, and also experienced different weather conditions.

Each time, the volunteers use 3 phones. One acts as reference phone, the other two phones act as passenger phones. The phones are synchronized before the volunteers get on the bus, which allows us to get the ground truth for evaluation. After the volunteers get on the bus, the one carrying the reference phone sits close to the driver. The reference phone is fixed on the seat close to the driver. The two passenger phones are carried by the other volunteer who randomly selects available seat to sit. One of two passenger phones is in pants pocket, the other is in hand.

The reference phone collects sensor data from GPS, cellular network component (cellular network ID), accelerometer, gyroscope and barometer. Before the bus starts to move from the first station, the volunteer taking care of the reference phone marks a “start" on the reference phone; and then after the bus stops at the final station, this volunteer marks a “stop" on the reference phone. The start and stop markers in the reference trace tells the ground truth of the beginning and end of the bus trip. In the middle of the travel, the reference phone is not touched to avoid involving any motion from human.

The phone in the pants pocket of the passenger also collects the same sensor data as the reference phone. But the GPS information is only used for ground truth, it is not used in trace matching.

The phone in passenger’s hand also collects same sensors as reference phone. Same as the pants pocket-phone, the GPS information here is only used as ground truth, which is not used in trace matching. The phone in hand is used for two purposes. First, the passenger marks the bus stations on this phone, which provides further ground truth about the stations of the bus line when combined with the reference trace, because we don’t have detailed GPS location information about the bus stations in advance. Second, typing on the phone to mark the ground truth of stations will introduce the motion of the passenger, which allows us to mimic normal user operations on smartphone, although the operations are not intensive.

The collected data are processed and then used in evaluating the performance of RideSense.

B. Performance

To be a practical and useful system, RideSense should be able to achieve two goals: 1) It searches a candidate space as efficient as possible for a given passenger trace. 2) It identifies passenger trip with high accuracy. In this section, we verify our design and study the performance of RideSense.

In current implementation, we set relatively conservative conditions for macro-feature based searching space reduction. Fig. 4.6 shows the effectiveness of macro features in reducing searching space. On average, RideSense filters 91% of the candidate reference traces for each passenger trace before it goes into the micro-feature-based matching stage.

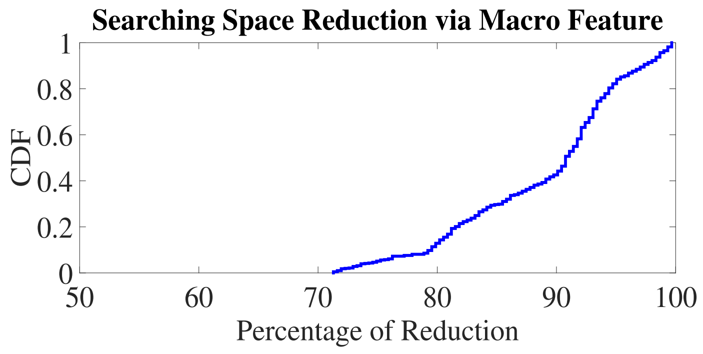


Figure 4.6: Based on macro features, on average 91% candidate reference traces are filtered for each passenger trace.

Next, we study whether RideSense could accurately identify: (i) *Which* bus line a passenger has taken; (ii) *When*, i.e., the shift of the bus she rides; (iii) *Where* she has traveled, i.e. the source and destination stations.

From the perspective of applications, RideSense could be used to bill a passenger. It could also help bus system operators analyze the traffic flows and the travel patterns of passengers.

For billing, knowing *which/where* is enough in most cases. Besides, we also want to find out how correctly the system will charge passengers and in what scenario, to what extent a passenger might be charged less or more than needed.

For traffic analysis, we consider the following fine-grained to coarse-grained measurements: i) *which/where/when*: It tells detailed travel information about each passenger, which helps the operators get fine-grained knowledge about the traffic system and travel pattern of each passenger. ii) *which/when*: It tells the bus line/shift a passenger has traveled, which could help the operator learn about the load on each bus line in a particular duration. iii) *which*: It enables the transit operator to be aware of the overall usage of its bus lines.

Therefore, RideSense is evaluated from four aspects, i.e. *which/where/when*, *which/where*, *which/when*, *which*. The accuracy of these measurements is defined by equations (1)$\sim$(4).

$$Accuracy_{Which/Where/When} = \frac{\sum L_P = L_R \wedge BS_P = BS_R \wedge SS_P = SS_R \wedge DS_P = DS_R}{\#MatchingPairs} \quad (1)$$

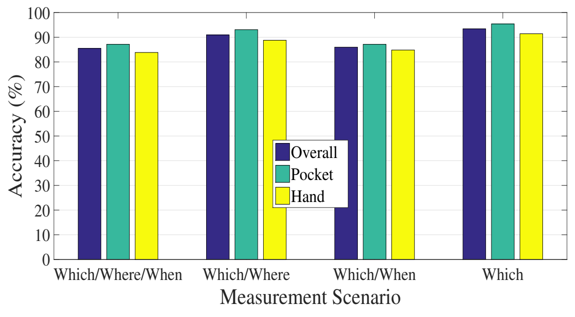
$$Accuracy_{Which/Where} = \frac{\sum L_P = L_R \wedge SS_P = SS_R \wedge DS_P = DS_R}{\#MatchingPairs} \quad (2)$$

$$Accuracy_{Which/When} = \frac{\sum L_P = L_R \wedge BS_P = BS_R}{\#MatchingPairs} \quad (3)$$

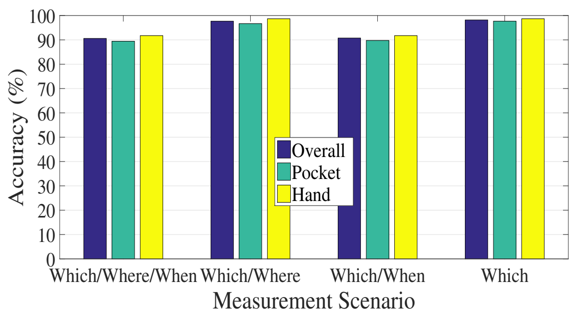
$$Accuracy_{Which} = \frac{\sum L_P = L_R}{\#MatchingPairs} \quad (4)$$

where L_P (L_R) is the id of bus line of passenger P (reference R); BS_P (BS_R) is the id of bus shift; SS_P (SS_R) and DS_P (DS_R) stand for source and destination segments.

The area we collected data is not completely flat, which exhibits barometer-friendly characteristics. Buses go through several slopes, which have rakes ranging from 5 to 30 degrees. Barometer is capturing the characteristics of the terrain of certain area, hence its performance may be less generalizable than motion sensors. Therefore, we evaluated RideSense based on micro features from motion sensors and barometer separately. The overall accuracy is shown in Fig. 4.7.



(a) Motion Sensors



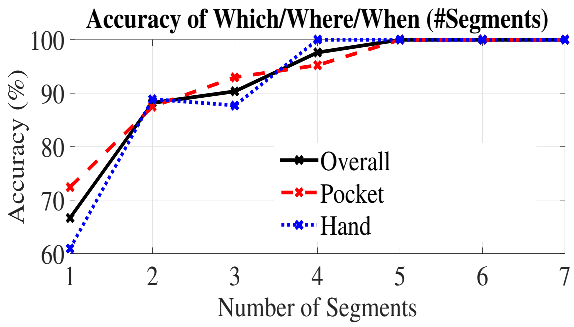
(b) Barometer

Figure 4.7: Accuracy of (a) motion-sensor and (b) barometer based RideSense in identifying *which/where/when*, *which/where*, *which/when*, and *which*, with phones in hand or pocket.

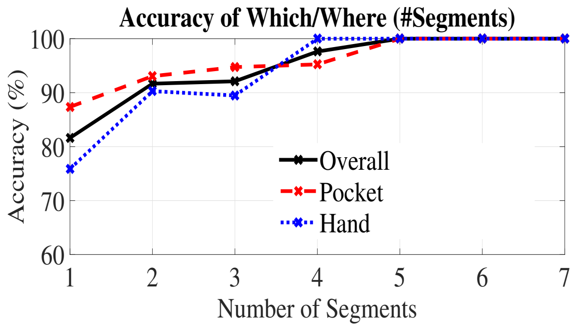
According to these results, when matching is based on motion sensors, the system achieves overall accuracy of 85% for *which/where/when*, 91% for *which/where*, 86% for *which/when* and *which* is identified with an overall accuracy to 93%. Barometer related results show that, in our area, if the RideSense uses barometer-based information, it could achieve much better performance than the situation with motion sensors. Its overall accuracy goes to 91% in the fine-grained measurement (i.e. *which/where/when*). In coarse-grained measurement, the accuracy is even higher, up to 98%.

When the evaluation is broken down into different positions of passenger phones, it is evident that with motion-sensor based RideSense, the position of the passenger phone affects the performance. In all the measurement, the passenger trace from the pocket phone could achieve higher correlation with the corresponding reference trace than the correlation between the trace from the hand phone and the reference phone. The reason behind this is that the motion sensor data collected by the phone in hand is polluted by the compensatory motion of hand, which weakens its correlation with the reference trace. On the contrary, if it is barometer-based, the positions of the passenger phone do not play a role and the accuracy between different phone positions does not show significant difference.

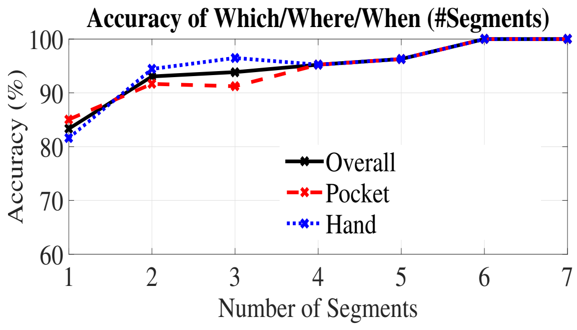
A passenger trip might cover different numbers of segments of each bus line. In the bus lines we collected, the numbers of segments range from 4 to 7. We derive sensor traces with all possible numbers of segments a passenger might travel, i.e. 1∼7 segments each trip. Then, we study the matching accuracy between passenger and reference traces based on different numbers of segments. The result is shown in Fig. 4.8. Due to the limitation of space, we only show the result for *which/where/when* and *which/where*. The accuracy of *which/when* and *which* also follow a similar trend.



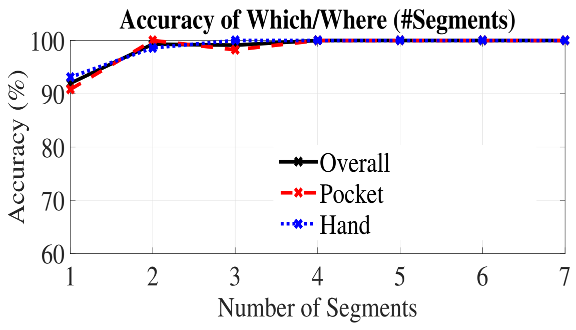
(a) Motion Sensors



(b) Motion Sensors



(c) Barometer



(d) Barometer

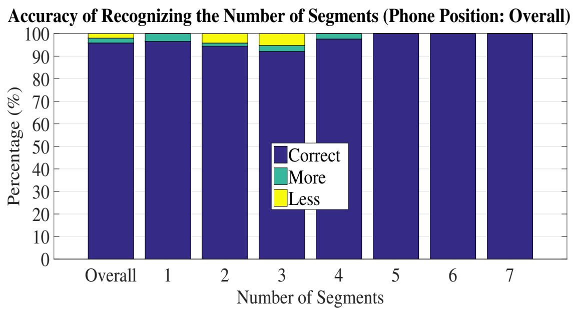
Figure 4.8: The accuracy of motion-sensor-based RideSense in identifying which/where/when (a) and which/where (b) when a passenger travels different numbers of segments. (c)(d) The accuracy of RideSense when barometer is used.

Overall, with motion sensors or barometer, the accuracy increases along with the increasing numbers of segments. In other words, when a passenger travels more segments on a bus line, the system can identify her ride more reliably.

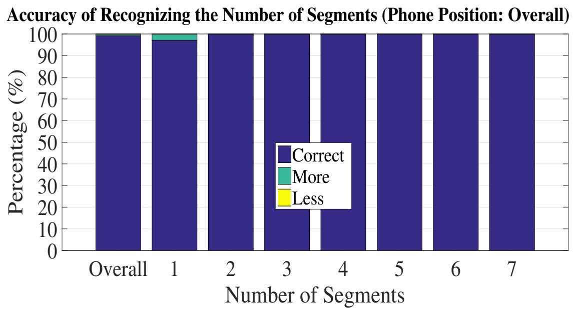
Consistent with the result shown in Fig. 4.7, in our area, barometer-based RideSense achieves higher accuracy than motion-sensor-based system. When motion sensors are used, the position of the passenger phone matters. The trace from the pocket phone could be used to recognize the numbers of segments a passenger traveled more accurately. When barometer is used, RideSense performance is not affected by the position of the phone or the motion of the passenger.

In many countries/cities, transportation fare is proportionate to the number of segments a passenger travels. After matching passenger trace with reference trace, if RideSense concludes a longer bus-riding than a passenger's actual trip, the passenger might be charged more than what she should pay. Conversely, if a shorter bus-riding is concluded for a passenger trace, the passenger might pay less than she should. Fig. 4.9 shows how accurately RideSense identifies the numbers of segments a passenger has traveled. When motion sensors are used, overall, a passenger will be charged correctly in 96% instances. In 2% cases, RideSense might charge a passenger more than she should pay, and in another 2% cases, the system charges passenger less. When a passenger takes a longer trip, more likely she will be charged correctly. Also, phone in pocket is more friendly to the billing system than the phone in hand. When barometer is used, the overall accuracy climbs to 99% and the performance is agnostic to the position of phone.

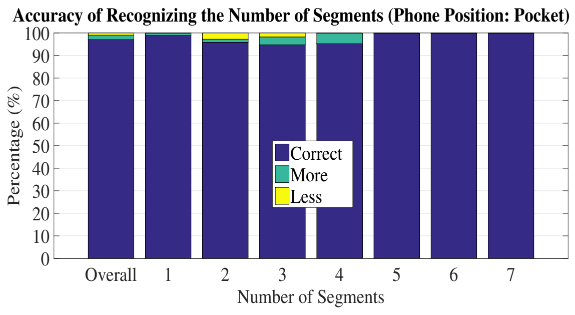
To summarize, our preliminary study shows that RideSense, while not yet accurate enough to be an alternative ticketless system, holds promise. By comparing the performance of motion-sensor-based matching and barometer-based matching in RideSense, we find that, in our area, by using barometer-based feature, the system could outperform motion-sensor-based method. We admit that the performance



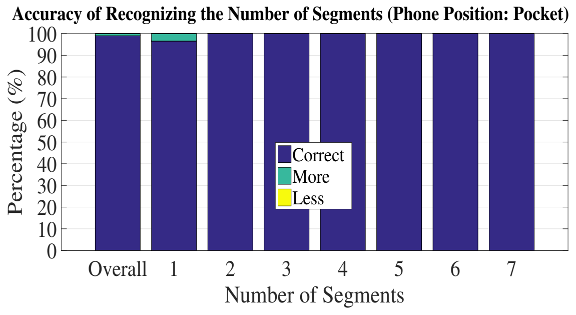
(a) Motion Sensors (Overall)



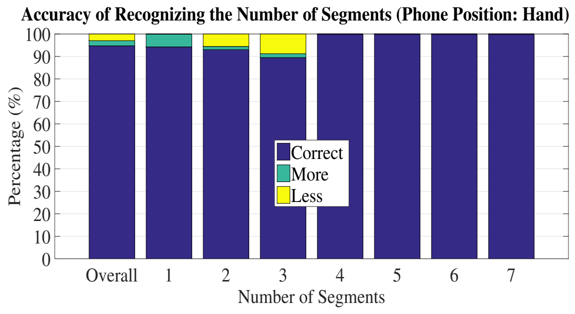
(d) Barometer (Overall)



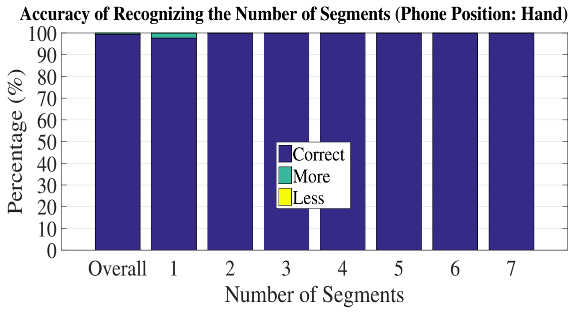
(b) Motion Sensors (Phone in Pocket)



(e) Barometer (Phone in Pocket)



(c) Motion Sensors (Phone in Hand)



(f) Barometer (Phone in Hand)

Figure 4.9: (a∼c) The accuracy of motion-sensor-based RideSense in recognizing the numbers of segments a passenger has traveled, which decides whether the passenger will be charged correctly, or charged more, charged less. (a) shows the overall accuracy; (b) shows the accuracy when the passenger phone is in pocket; (c) shows the accuracy when the passenger phone is in hand. (d∼f) Performance of barometer-based RideSense in charging passengers.

of barometer-based RideSense is correlated with the terrain in our area. For real deployment, we could combine motion sensor with barometer to build a hybrid RideSense, which utilizes the characteristics exhibited both in motion and terrain to achieve best performance.

4.3 Limitations and Discussion

Needless to say, this paper is a small step towards the broader vision; substantial work remains as discussed here.

User Behavior: In our experiment, only minor user motion is introduced (i.e. the user is tapping on the phone in hand for marking the ground truth) in the sensing process of the passenger phones. In reality, situation might be much more complex. For instance, a passenger might be playing game with her smartphone during her bus trip. A malicious user might deliberately shake and move the phone to continuously and intensively distort the motion trace. These user behavior could pollute the sensor trace of the passenger's bus-riding, which thereby hinders the correlation with the reference trace. To cope with these situations, we are exploring factorization algorithms to separate vehicle motion from human motion, given that they have some statistically distinct properties. We leave this to future work.

Bandwidth and Energy Considerations: We have also assumed that the uploaded data is only from the segments during which the passenger is in the bus – in reality, this is non-trivial. A classifier that recognizes that a person has boarded a bus will need to run continuously, imposing an energy burden. Even if continuous sensing and classification can be solved efficiently, not all the data during a vehicular trip may need to be uploaded. Identifying the most discriminating data segments, and uploading them at opportune times, will reduce the upload bandwidth from millions of users. A more careful treatment is needed for a complete system.

Additional Opportunities: While challenges are many, opportunities exist too. Data from multiple passengers in the same bus should exhibit similarities that could be useful in improving the system's accuracy, and in thwarting misbehavior. The post-paid model possible with RideSense could also enable various pricing schemes and discounts. For instance, those that traveled in a crowded bus, or were delayed by accidents, could be compensated through a discount applied to their fare later, ultimately incentivizing public transportation. Moreover, in Uber-like business, by correlating the sensor traces from driver's phone and passenger's phone, service could be improved in case GPS signal is lost or corrupted.

In current implementation and evaluation, we fixed the reference phone on a seat close to the driver and tried passenger phones in pocket and hand. As a next step, we plan to try both the reference phone and passenger phone in different positions, and collect data with more bus lines. We will also carry out the study in different areas to see how the system will perform in different traffic situations and terrain.

4.4 Related Work

Smartphone sensors have been used in traffic/transportation related research to detect transportation mode, study driver's behavior, localize vehicles, and even identify travel information. Here we list several works which are close to our solution.

Trellis [59] is a WiFi-based solution for transit analytics. It utilizes the WiFi infrastructure in buses to detect the WiFi-enabled mobile device carried by passengers. This system could be used to identify popular routes, occupancy of buses, etc. Compared with our solution, with Trellis, WiFi in passengers' devices must be enabled. Due to the characteristics of wireless signal, the passengers in other parallel-moving vehicles could be erroneously counted as the passengers in current bus. While with RideSense, even two vehicles are moving in parallel, they still exhibit difference in motion (for example, the acceleration/deceleration in speeding up/down, time duration of stops, etc), which make the two vehicles potentially distinguishable.

SmartLoc [60] and eNav [61] are two works utilizing smartphone sensors to localize vehicles. They use low-power sensors to provide approximate location information through detecting landmarks and driving patterns, or by knowing the fine-grained road information from map. Both systems rely on GPS to obtain accurate location when the error goes large.

VTrack [62] and CTrack [63] use smartphone to collect GPS, WiFi and GSM information along a user's trip to figure out the road segment and trajectory a user travels. By aggregating the travel information, these two systems learn about the traffic situation and estimate travel time to help on route planning.

Authors in [64] utilize bluetooth to scan passenger's phone to identify passenger's trip on a bus. It treats first detection of a phone as the source station and if unreachable for a certain time, it concludes the passenger has got off. This solution cannot reliably isolate its detection within a bus, people in a neighboring bus might also be detected. It also incurs a privacy problem, as anyone can detect the passengers.

The works [65], [66] and [67] also perform sensor-based trace matching, similar to our system. [65] uses GPS and barometer to collect altitude related information and matches a passenger's partial GPS/barometer trace with pre-constructed reference trace to find out the bus line of travel. [66] uses GPS, WiFi and accelerometer data to determine bus-riding and matches routes based on the shape of the road in horizontal plane. [67] recognizes bus line and estimates arrival time based on matching cell tower ID sequence. This work relies on audio processing to detect the audio signal of IC card reader in order to tell whether a user is in public transit system. Compared with these works, our approach uses only energy-efficient sensors and provides *which/when/where* information about a passenger's travel, enables detailed traffic analysis.

4.5 Summary

We envision smart transportation of the future wherein purchasing tickets is not necessary. By exporting sensor data from passengers' smart devices and public vehicles, it should be possible to detect how a passenger utilizes public transportation, and bill her appropriately at the end of a day or a month. From an operational point of view, such a system could eliminate the entire ticketing and maintenance infrastructure (ticket dispensers/sellers, gate checks, card readers, etc.). From a user's perspective, she could experience a seamless hop-on/hop-off experience, without worrying about correct pricing, cash amount, losing the ticket, etc. While realizing this vision indeed poses logistical hurdles, RideSense is a first step towards initiating the process and asking the right questions. We believe there is adequate promise to engage into a serious research effort.

Part III

Improving Experience of Shopping

In shopping activities, the demand for location based services keeps increasing. Customers require location services to find stores and merchandise; business owners utilize those services to locate their customers and advertise their products. Most location based services require semantic place names such as Staples, rather than physical coordinates. Past work has mostly focussed on achieving localization accuracy, while assuming that the translation of physical coordinates to semantic names will be done manually. This part makes an effort to automate this step, by leveraging the presence of a website corresponding to each store and the availability of a repository of WiFi-tagged pictures from different stores. By correlating the text inside the pictures, against the text extracted from store websites, the proposed system, called AutoLabel, can automatically label clusters of pictures, and the corresponding WiFi APs, with store names. Later, when a user enters a store, her mobile device scans the WiFi APs and consults a lookup table to recognize the store she is in. This system could serve as a bridge for the information flow between customers and business owners, which benefits both sides.

CHAPTER 5

AUTOLABEL: LABELING PLACES FROM PICTURES AND WEBSITES

Given the prime spot location occupies in determining the context of a user, and the popularity and profit potential of context-aware services to mobile users, it is not surprising that localization has attracted a lot of research attention from both industry and academia. Most of the research has been directed at improving the localization accuracy; localizing a user with $< 5m$ accuracy is possible today. However, to roll out location based services in the wild, a semantic understanding of a place (e.g., Staples, CVS) is an equally crucial piece of the big localization puzzle and far less research has been focussed on that.

The core challenge here is in data labeling, i.e., annotating localization output with store names. Suppose WiFi AP vectors heard inside a store are unique. Then, a trivial solution to the above problem is to incentivize crowd-sourced users to walk into different stores, record WiFi APs inside them, and label the stores manually. However, such a nation-wide manual effort is unattractive and unlikely to take off quickly. As an alternative, we investigated if WiFi APs in stores are actually named after the store. In our area, we have observed that only less than 30% stores have meaningful AP names (e.g., Expresso Royale, Panera, Bestbuy-guest), from which the store names could be guessed easily. An overwhelming majority are unrelated to the stores. Besides, in a large shopping mall, public WiFi service is provided by the shopping mall authority, therefore most of the stores there do not have their own APs. As a result, the reality of SSID naming and WiFi deployment has rendered the idea of mining store names from WiFi SSIDs inadequate.

We find an opportunity to address this problem in the increasing frequency with which people are taking pictures everywhere. Given the limited storage available on mobile devices, many users are likely to upload their pictures to cloud, using services such as Google Photos [68] or Apple iCloud Photo Library [69]. We expect that some of these pictures will be from inside stores, since it is not unusual for people to take such pictures for soliciting the opinion of their family members or friends before buying an item. Considering that 2.5 trillion pictures will be shared or stored online in 2016, and 90% of them will be taken on smartphones [70], even if a tiny fraction of these pictures are from inside stores, that would form a large repository of in-store pictures.

It is expected that a pay off for cloud service providers is in mining the pictures and the associated metadata for geolocation. In recent years, many camera and mobile device manufacturers have rolled out cameras and/or camera apps that allow users to take geo-tagged pictures [71, 72]. When a user is taking a picture, the device/app records the GPS/cellular network/WiFi information [73] simultaneously, which helps in determining the location of the picture. Compared with the size of pictures themselves, the size of these geo-tags is negligible, but they could greatly benefit the users in categorizing and sharing pictures based on their location [74, 75].

Given this repository of in-store pictures tagged with WiFi APs, we wondered whether it is possible to automatically label WiFi APs with semantic store-names. We immediately considered the possibility of finding logos and names of the stores in the pictures. If a picture from Starbucks has the word “Starbucks” written in it, or contains the distinct green logo of the mermaid, it would be immediately possible to label the corresponding WiFi APs. Unfortunately, this approach presented

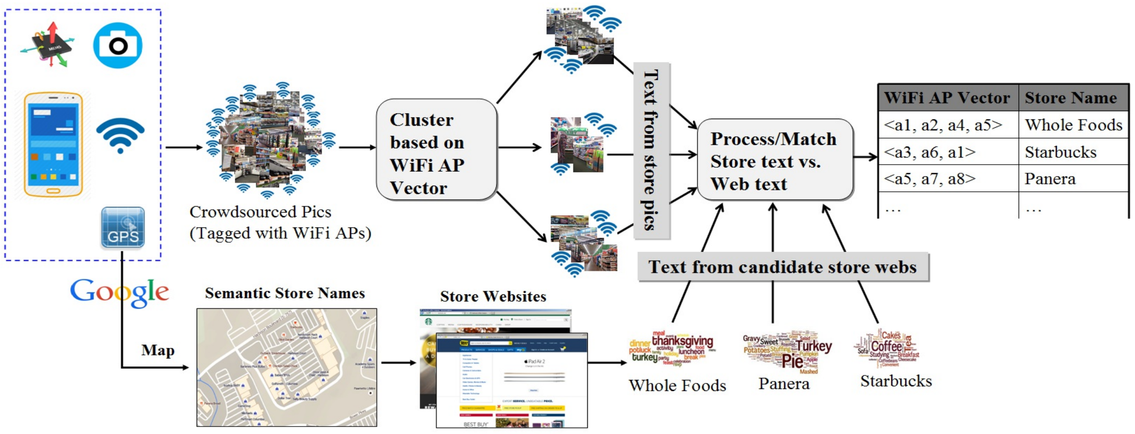


Figure 5.1: An operational overview of the AutoLabel system.

hurdles. For instance, (1) pictures from a "Payless" shoe store often had numerous logos of Nike and Adidas, similar to an actual Nike or Adidas store. (2) When looking into the actual data, we realized that with many stores, not a single picture from the store had the logo or the store-name in it. (3) Finally, even if a picture fortunately had a logo or store-name, the WiFi APs corresponding to that picture may not represent all sets of AP vectors audible in that store. A Walmart may have APs {a1, a2} audible in its grocery section, but {a2, a3} in the meat section – pictures from both these sections need to contain logos and store-names. Such occurrences are far less likely – the appropriate solution simply cannot rely on the store's name or logo being present in all pictures.

We propose an approach that exploits the presence of two "avatars" – online and offline versions – of the same store. Our core intuition emerges from a hypothesis that different words visible in pictures from a store X, are also likely to occur on the website of the store X. Put differently, let $H_{store}(Starbucks)$ denote the histogram of words in the pictures of Starbucks, and $H_{web}(i)$ denote the same from store i's website. We hypothesize that when $H_{store}(Starbucks)$ is matched against $H_{web}(i)$, the strongest match will correspond to $i = Starbucks$. Of course, the matching need not be performed for all stores in the world – given that a picture's rough location is also known, only stores in the vicinity of that location can be candidates. Figure 5.1 illustrates the idea.

Of course, the above may seem reasonable only if the pictures in the repository are already clustered per-store. Unfortunately, the picture repository may be flat, i.e., we may not have information about which pictures are from the same store. However, we show that even without this information, the end goal can be achieved through a technique we develop, called *simultaneous clustering and labeling*. Thus, we implement a processing pipeline that accepts crowd-sourced pictures as input and outputs a look-up table of WiFi AP vectors and store-names. We evaluate our system, *AutoLabel*, across 40 different stores and show performance variations with increasing numbers of crowd-sourced pictures from each of them. These results show labeling accuracy ranging from 87% to 94%, and even 100% in some cases. Our main contributions in this paper are summarized as follows.

- *Crafting the hypothesis that the text visible in the ambience of a store may match well with the text on that store's website.* We systematically validate this hypothesis across 40 different stores and believe that this finding can be useful in other contexts as well.

- *Building an end to end solution that labels APs based on in-store pictures, using techniques in text-matching and cross correlation.* We evaluate this system using real-world pictures from stores and demonstrate that a store name can be recognized based on the WiFi APs, with around 10 valid pictures only from each store.

The following sections expand on these contributions, beginning with a broader formulation of the research problem, followed by the system design and evaluation.

5.1 Formulation and Groundwork

The fundamental problem we face is essentially the need to affix semantic labels to electronic data. Electronic data here refers to WiFi AP vectors generated by devices, that have no connection to semantic labels – Starbucks, airport, library – that are known to humans. They seem to be two orthogonal dimensions of information. What we need is a bridge, or a common dimension, that would connect electronic data to semantic labels. If electronic data and semantic labels can both be projected to this common dimension, then the projected data can be matched for labeling. This work observes that words could form that common dimension. Fig. 5.2 illustrates this abstraction. On one hand, WiFi data can be projected to this dimension through pictures taken inside stores, while semantic labels (read web domain names) can be translated to the same dimension through webpages. If the words from the stores and webpages match well, electronic data (WiFi APs) can be given semantic labels (store names).

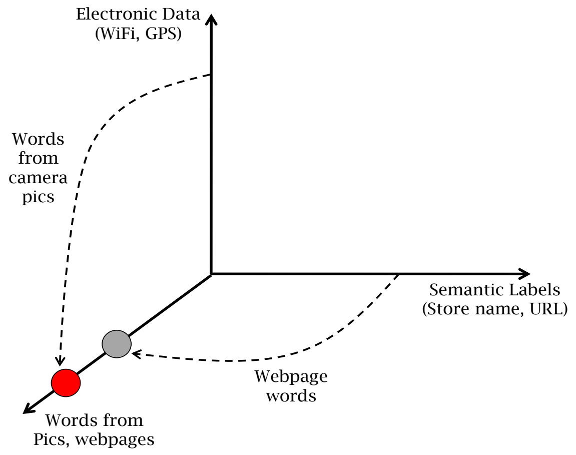


Figure 5.2: Unsupervised labeling by projecting electronic data and semantic labels to a common dimension.

The success of this idea hinges on the hypothesis that words extracted from pictures of a store are likely to "correlate" well with words extracted from the webpage of the corresponding store. More specifically, words from Best Buy should match most with words from Best Buy's website (compared to other websites of nearby stores). In seeking evidence for this hypothesis, we performed some experimental measurements.

We visited 6 stores near the university campus and walked the entire store with a (smartphone) video camera – the walk was performed like a "raster scan". From the videos of these stores, we removed some of the blurry frames, and extracted the text from them. The output was a frequency distribution of words from each store, say $H_{store}(i)$. Then, from each of the websites of these stores, we parsed out all words, extracted all nouns and proper names, and computed a frequency distribution, $H_{web}(i)$. Thus, $H_{store}(i)$ and $H_{web}(i)$ are both histograms of words – we need to compute a similarity measure between them. Treating these as words from documents, we applied a basic document matching algorithm [76]. Briefly, when a word from $H_{store}(i)$ and $H_{web}(i)$ matches, the algorithm essentially increases the matching score in proportion to the frequency of occurrence of the word.

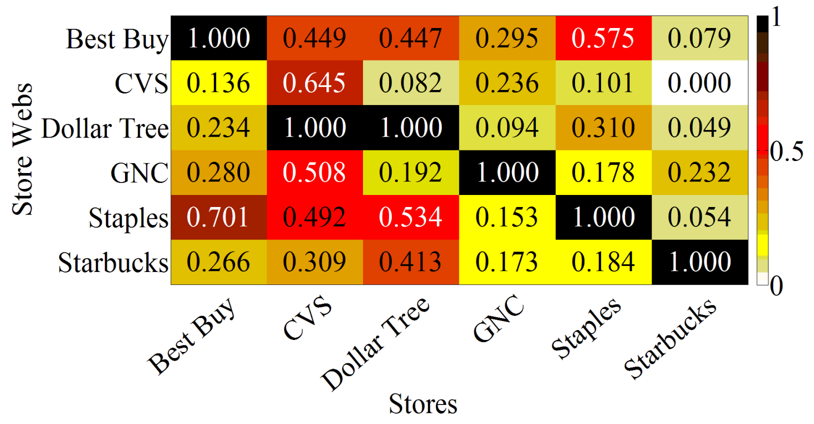


Figure 5.3: Matching between words in stores and webpages.

Fig. 5.3 plots the confusion matrix of the matching scores between words in stores and webpages. The values are normalized appropriately so they range between $[0, 1]$. An element ij in the matrix denotes the score between words from store i and words from webpage j. The elements are color coded based on their matching scores – darker shades imply higher values. Evidently, most of the diagonal elements are dark while the others are lighter. Some false positives and false negatives do occur. Although this is a toy experiment and results are not perfect, this certainly extends confidence that the core idea may work in real life. Hence, we proceed with complete system design and validation, details of which are discussed next.

5.2 System Design

The core intuition behind AutoLabel is to match text from in-store pictures against text from websites of stores. The flow of operations in AutoLabel can be briefly described as follows.

(1) In-store Text Extraction: Given a cluster of pictures from a store, we first extract words from the pictures using optical character recognition (OCR) tools available publicly. However, not all words in a store are equally effective in the matching process. We observe that words that are *at or above the eye-level*, i.e., towards the ceiling or higher side of the walls, often refer to product categories, menus, banners, etc. These words tend to be stable over time and often reflect the core attributes of a store. Given that store webpages are also likely to include these words, AutoLabel selects at-and-above eye-level words to represent a store. When a crowdsourcer's smartphone takes a picture, the camera app records the pose of the smartphone camera through motion sensors (e.g. gyroscope, accelerometer) and attaches the pose information to the picture[1]. From the pose, AutoLabel could identify which region in a picture is at or above eye-level, and hence extract words from that region.

[1]Note that the sensor snapshot is recorded when taking a picture and is included as its meta data along with WiFi AP information. Furthermore, while WiFi data is essential to AutoLabel, its performance does not hinge critically on sensor snapshot.

(2) Candidate Websites and Web-Text Extraction: Based on the rough location of a picture (derived from last known GPS or cell tower ID), we query Google Map to obtain a list of stores in that broad area. Parsing the URLs returned by a subsequent Google search on each of these stores, offers the websites of candidate stores. Then, these websites are parsed to extract words from them, specifically meta data that define the webpage, menu items, product categories, etc.

(3) Matching Text and Labelling: The matching process treats the in-store and website text as documents and applies established document matching algorithms, such as Cosine Similarity with *term-frequency inverse document frequency* (TF-IDF) [76] based weights for words. The best matching website is announced as the store inside which the pictures are taken – all WiFi AP vectors from these pictures are labeled with this store name, resulting in an APs$-$StoreName lookup table.

(4) Simultaneous Clustering and Labeling: Given this processing pipeline, we now return to the problem of clustering. When we do not have an a priori knowledge about the correct clustering of pictures, we first use the available WiFi information to gain a crude understanding of pictures that do *not* belong to the same store. As a simple case, two pictures with completely non-overlapping WiFi AP vectors can be assigned to separate clusters. We then form clusters within each of these WiFi-based clusters. Now, in any given iteration of the algorithm, i, we have a clustering, say C_i. We pick each cluster c_{ij} in clustering C_i and compute its matching score against the candidate websites, and then sum ($c_{ij}\ \forall j$) to obtain a score for C_i, say S_i. We repeat this operation for different clusterings of the same pictures. Our intuition is that the correct clustering, say C_r, would achieve the maximum matching score, i.e., $r = \text{argmax}(S_i)$. If this holds, we can label all the stores and their WiFi vectors in one shot, and populate the APs$-$StoreName look-up table.

(5) Localizing Users: When a user visits a store, her smartphone overhears a WiFi AP vector and performs a lookup on this vector in the APs−StoreName table. Observe that no store may match exactly with this vector, and many stores may match partially with this vector. We design a vector matching algorithm that considers the number of APs matched as well as the RSSI-based order of APs in each of these vectors. The output of this match is a store name that is supplied to apps that provide semantic location based services to the users. We elaborate on each of these components below.

A. Extracting Web and Store Text

AutoLabel expects the metadata associated with a picture to include the vector of WiFi APs heard by the device taking the picture, and also a rough location (could be using GPS). AutoLabel uses Google Map to search the businesses around this rough location and gets a list of candidate place names (could be retail store, restaurant, etc.). Then, it performs web search on these place names to get their homepages and extracts the web text.

Given that most business web sites have a certain structure, we leverage it to extract the higher level discriminative words from the web sites. For instance, typical business homepages contain category/menu section, which is used to categorize the products and navigate to second-layer pages. Therefore, we extract the words from first and second level menus on a web site. Besides the text shown on the webpage, many stores also define meta keywords in the html files of their homepages. While the meta keywords are meant for search engines, they usually contain the words which describe some of the key characteristics of that store. So, in AutoLabel, we also extract meta keywords from the stores' webpages.

The text from web pages may contain lots of unmeaningful information, such as symbol, punctuation, preposition and adverb, which do not represent the semantics of the store. Instead, nouns and proper names are more descriptive. Therefore, AutoLabel carries out noun/proper name extraction on the web text. In this work, Stanford NER [77] is used to identify proper names and WordNet [78] is utilized to check whether a word is a noun.

To extract text from pictures, several powerful OCR tools such as Google Goggles [79] and Amazon Firefly [80] exist. Similar to the web text, we sanitize the store text by filtering out symbols, prepositions, adverbs, etc., while keeping nouns and proper names. To avoid filtering out store name, which can be quite discriminative if it appears in a picture, we supply the list of candidate store names derived through Google Map to the sanitization procedure. Next, we present how the store text is matched against text from candidate web sites to find the store name.

B. Matching Store and Web Text

Given the store text ST of an unknown store x, AutoLabel compares it against the web text WT_s of each candidate store s (see Table 5.1 for notation) and computes the similarity score. The store s^* whose web text has the highest similarity score is deemed the matching store, i.e., $x = s^*$. The pseudocode of this procedure is given in Algorithm 1.

Algorithm 1 extracts the text from the given set of pictures. It then sanitizes the text to extract nouns and proper names, while keeping the potential store names. Now it needs to compare the resulting bag of words ST[2] against the bag of words WT_s from each candidate store s. If any words that match store names appear in ST, then the candidate set CS is restricted to those stores.

While matching text, for capturing the importance of a word, we adopt the `TF-IDF` method used in information retrieval and text mining. Within a store, words

[2]For convenience, we abuse the notation and use symbols like ST to refer to two different things like bag of words and unique set of words. But the intent will be clear from the context.

Table 5.1: Notation

$\widehat{IM}$	Set of images taken inside the stores in an area
$\widehat{AP}$	Set of AP vectors that are labelled
$\widehat{CS}$	Set of candidate stores
ST_i	Text extracted from an in-store image i
AP_i	Vector of APs associated with an image i
CS_i	Set of candidate stores for an image i
WT_s	Bag of words from the web site of store s
$IM_{c,s}$	Images associated with store s in clustering c
$ST_{c,s}$	Text extracted from images $ST_{c,s}$
SN_{AP}	Histogram of store names associated with the AP vector AP, i.e., $\{(s, n) : n$ is the number of instances AP is labelled with $s\}$
$f(t, T)$	Frequency of word t within the text T
$maxf(T)$	Maximum frequency of any word in T
$w(t, T)$	Weight assigned to a word t within the text T
$\texttt{Sanitize}(T, S)$	Filter T to exclude words that are not nouns or proper names, while keeping the store names S
$\texttt{HistMerge}(H)$	Merge the set of histograms H into one
$\texttt{Rank}(a, A, B)$	Rank of a in vector A w.r.t. elements in set B

which occur more frequently are likely to be more important than other words in characterizing it. The word "shoe" appears often in Shoe Carnival and helps recognize it among nearby stores that also sell shoes but many other items as well. The *term frequency* (TF) method gives more frequent words proportionally higher weight than infrequent ones within a store. On the other hand, if a word appears in only one store, even if infrequently, it helps discriminate that store from others. If a word occurs in many stores, its contribution in discriminating the store will naturally diminish. The *inverse document frequency* (IDF) method captures that intuition by giving higher weight to less common words among a set of stores.

Specifically, AutoLabel employs TF $\times$ IDF as the weight for each word. It uses augmented TF, which is computed as $\texttt{TF}(t, T) = 0.5 + \frac{0.5 \times f(t,T)}{\texttt{maxf}(T)}$, where $f(t, T)$ is the frequency of word t in text T; $\texttt{maxf}(T)$ is the maximal frequency of any word in T. IDF for a word t depends on its occurrence in the web text of all candidate stores. Suppose n is the total number of candidate web sites and k is the number of sites in which the word t occurs. Then $\texttt{IDF}(t) = 1 + log(\frac{n}{k})$. The resulting weight assigned

Algorithm 1 : Given the set of images IM associated with an unlabelled store and the candidate stores CS, label AP vectors with a store name: `SetLabel`(IM, CS)

```
1: /* sanitize extracted text but keep store names */
2: $ST \leftarrow \texttt{Sanitize}(\cup_{i \in IM} \texttt{OCR}(i), CS)$
3:
4: /* if store names appear in the in-store text */
5: /* then restrict the candidate set to those stores */
6: **if** $(ST \cap CS) \neq \emptyset$ **then** $CS = ST \cap CS$
7:
8: /* assign weight (TF x IDF) for all words */
9: **for each** word $t$ in $\cup_{s \in CS} WT_s$ **do**
10:    $\texttt{IDF}(t) \leftarrow log(1 + \frac{|CS|}{|\{s \in CS: f(t, WT_s) > 0\}|})$
11: **for each** word $t$ in $ST$ **do**
12:    $w(t, ST) \leftarrow (0.5 + \frac{0.5 \times f(t, ST)}{maxf(ST)}) \times \texttt{IDF}(t)$
13: **for each** store $s$ in $CS$ **do**
14:    **for each** word $t$ in $WT_s$ **do**
15:       $w(t, WT_s) \leftarrow (0.5 + \frac{0.5 \times f(t, WT_s)}{maxf(WT_s)}) \times \texttt{IDF}(t)$
16:
17: /* find the web site most similar to the store */
18: **for each** store $s$ in $CS$ **do**
19:    $SIM_s \leftarrow \frac{\sum_{t \in ST \bigcap WT_s} w(t, ST) \times w(t, WT_s)}{\sqrt{\sum_{t \in ST} w(t, ST)^2 \times \sum_{t \in WT_s} w(t, WT_S)^2}}$
20: $s^* \leftarrow \text{argmax}_s(SIM_s)$
21:
22: /* add s* as a label to each AP vector */
23: **for each** $i \in IM$ **do**
24:    $SN_{AP_i} \leftarrow \texttt{HistMerge}(\{SN_{AP_i}, \{(s^*, 1)\}\})$
```

to each word t in T is $w(t, T) = \texttt{TF}(t, T) \times \texttt{IDF}(t)$ (lines $9-15$). It then computes the similarity SIM_s between the store text ST and the web text WT_s of each candidate store s (lines $18-19$). This is essentially the Cosine Similarity, with 1 being most similar[3]. If the store s^* is the one with the highest similarity measure SIM_{s^*}, then s^* is deemed the name of the store from which the text ST is gathered.

[3]While Cosine Similarity measures between -1 and 1, with 1 being most similar and -1 being diametrically opposite, in our context, it is bounded in positive space between $[0, 1]$.

Algorithm 2 : Given the collection of images $\widehat{IM}$ and the set of candidate stores $\widehat{CS}$, perform simultaneous clustering and labeling: `SCAL`($\widehat{IM}, \widehat{CS}$)

```
1: /* find candidate stores for each image */
2: for each $i \in \widehat{IM}$ do
3:     $CS_i \leftarrow \{s \in \widehat{CS} : (ST_i \cap WT_s) \neq \emptyset\}$
4:
5: /* generate all potential clusterings of images */
6: for each $c \in \times_{i \in \widehat{IM}} CS_i$ do
7:     /* skip if AP vectors corresponding to any two images in a cluster have no
       common AP */
8:     if $\exists i, j \in IM_{c,s} \wedge ((AP_i \cap AP_j) = \emptyset)$ continue
9:
10:    /* compute similarity for this clustering */
11:    for each $s \in \widehat{CS}$ do
12:        $SIM_{c,s} \leftarrow \frac{\sum_{t \in ST_{c,s} \bigcap WT_s} w(t, ST_{c,s}) \times w(t, WT_s)}{\sqrt{\sum_{t \in ST} w(t, ST_{c,s})^2 \times \sum_{t \in WT_s} w(t, WT_S)^2}}$
13:    $SIM_c \leftarrow \sum_{s \in \widehat{CS}} SIM_{c,s}$
14:
15: /* get one that maximizes sum of similarities */
16: $c^* \leftarrow \mathrm{argmax}_c(SIM_c)$
17:
18: /* label AP vectors of images in $s$'s cluster with $s$ */
19: for each $s \in \widehat{CS}$ do
20:     for each $i \in IM_{c^*,s}$ do
21:         $SN_{AP_i} \leftarrow$ HistMerge($\{SN_{AP_i}, \{(s, 1)\}\}$)
```

C. Labeling APs

Labeling of APs automatically by correlating the text in store images and the web text is the essence of AutoLabel. With dense deployment of APs, it is likely that an AP vector is associated with images from a single store and thus gets a unique label. But it is possible that a device may hear the same set of APs in two neighboring stores, Starbucks and Radio Shack. Hence, two images, one in Starbucks and the other in Radio Shack, may record the same set of APs and get labelled with those two store names. However, we can disambiguate such scenarios and increase the likelihood of unique labels by considering ordered vector of APs based on their RSSI values. In our implementation and evaluation, we adopted ordered AP vectors, as it is observed that such an ordering is fairly stable [81]. Nevertheless, the same ordered AP vector may be heard in multiple stores, and we should prepare for that while assigning and retrieving labels.

To account for multiple labels for the same AP vector, we store the mapping from AP vector to store name, SN_{AP_i}, as a histogram, i.e., a set of tuples (s_j, n_j), where n_j is the number of instances in which AP_i is labelled with store name s_j. Whenever AP_i gets another label s_k, SN_{AP_i} is updated to increment the number of instances n_k, if s_k is an existing label, else add a new tuple $(s_k, 1)$. Suppose an AP vector $\{B, C, D\}$ is labelled with Starbucks in 17 instances and RadioShack in 2 instances. Then, the corresponding histogram for $\{B, C, D\}$ is $\{$(Starbucks, 17), (RadioShack, 2)$\}$. Now, if the same vector gets mapped to Starbucks again, the histogram gets updated to $\{$(Starbucks, 18), (RadioShack, 2)$\}$. We refer to this update by using a generic function `HistMerge` that merges a set of histograms in to one. In Algorithm 1, once the store text ST and the corresponding images IM are determined to be from a store s^*, the AP vector AP_i associated with each image i is labelled with store name s^* (lines $23-24$). Here, SN_{AP_i} is updated by merging it with a new histogram of one tuple, $\{(s^*, 1)\}$ (line 24). This SN_{AP} table is referred later for semantically localizing users.

D. Simultaneous Clustering And Labeling

The discussion thus far has been about identifying the store name given a set of images from that single store. In practice, the AutoLabel system may not know in advance which of the images in its collection are taken from the same store. Therefore, it has to perform simultaneous clustering and labeling (SCAL). Here, we present a version of SCAL in Algorithm 2.

The first step is to identify, for each image, a subset of candidate websites with matching text (lines $2-3$). Based on this reduced set of candidate stores per image, we generate all possible clusterings. These clusterings are further filtered by using AP vector information associated with images. We observe that two AP vectors inside one store have at least a common AP. Therefore, it is expected that two images that fall within a cluster have overlapping AP vectors. Otherwise, that clustering is not considered further (line 8). For each of the remaining clusterings, a similarity score is computed. This is the sum of the similarity scores between the text in images of each cluster and the corresponding store website (lines $11-13$). Once we find the clustering with the highest similarity score, the AP vectors of images in each cluster can all be labelled with the corresponding store name (lines $19-21$). The rationale is that when the clustering is right, the similarity scores for each cluster would be high and so too is their sum. Naturally, more sophisticated SCAL algorithms can be devised and exploration of which is the focus of our future work.

E. Localizing Users

The purpose of labeling of AP vectors with store names by AutoLabel is to enable semantic localization of users based on the APs heard by their mobile devices. Given an AP vector heard by a device, it is straightforward to retrieve the associated store names if that exact AP vector is labelled by Algorithm 1 earlier. In practice, APs heard at a location vary over time, particularly with smartphones also acting as mobile hotspots. However, the heard AP vector is likely to include the APs of the stores around. Therefore, even a partial match may be sufficient to localize a device to the correct store.

Algorithm 3 looks for the labelled AP vectors with the highest cardinality match with the given AP vector AP. If there is no matching AP, it returns empty. Among the matching labelled AP vectors, we find the more specific match by considering the relative ordering of APs. The labelled AP vector that has APs in the most similar

Algorithm 3 : Given the vector of APs heard at a place, return the histogram of store names : `GetLabel`(AP)

```
1:  /* Find the maximum length of an AP vector in our collection that matches with
    the given AP vector */
2:  $m \leftarrow \max_{A \in \widehat{AP}} |AP \cap A|$
3:
4:  /* If no matching APs at all , return null */
5:  **if** $m = 0$ **return** $\emptyset$
6:
7:  /* Find vectors with max cardinality match */
8:  $M \leftarrow \{A \in \widehat{AP} : |AP \cap A| = m\}$
9:
10: /* Find most similar vectors as per RSSI-order */
11: **for each** $A \in M$ **do**
12:     $A' = A \cap AP$
13:     $SIM_A = \sum_{a \in A'} \frac{1}{\texttt{Rank}(a, A', AP)} \times \frac{1}{\texttt{Rank}(a, A', A)}$
14:
15: /* Merge histograms of most similar vectors */
16: **return** HistMerge($\{SN_A : A \in M\}$)
```

order as that of the given AP vector is chosen as the best match. This process is expected to yield a unique match with an AP vector and then we return the corresponding histogram of store names. It is possible that a given AP vector matches with two labelled vectors with identical similarity score. Then, we merge the store name histograms of all the matching labelled AP vectors.

Suppose an AP vector $\{B, C, D\}$ is mapped to store names {(Starbucks, 8), (RadioShack, 2)}, i.e., a device hearing APs $\{B, C, D\}$ is near Starbucks and RadioShack, with probability of 0.8 that it is inside Starbucks and 0.2 inside RadioShack. Similarly, suppose another AP vector $\{E, C, D\}$ is labelled with {(Staples, 4), (RadioShack, 6)}, implying that the device is inside RadioShack and Staples with probabilities 0.6 and 0.4 respectively. Now, when a device hears APs $\{F, C, D\}$, Algorithm 3 finds that the labelled AP vectors $\{B, C, D\}$ and $\{E, C, D\}$ match the most and returns the resulting histogram {(Starbucks, 8), (RadioShack, 8), (Staples, 4)}. In other words, a user with the device hearing APs $\{F, C, D\}$ is near

Starbucks, RadioShack, and Staples, and the probability that she is inside those stores is $0.4, 0.4,$ and 0.2 respectively.

An end-user app may use the information returned by AutoLabel system in different ways. It may use the list of nearby store names to offer coupons to the user. Or, it could pick the most probable store, if the probability is beyond a certain high threshold, and act accordingly. For instance, in the above example, an app on a device hearing APs $\{B, C, D\}$ may conclude that it is inside Starbucks, given that the corresponding probability of 0.8 is high enough. Overall, we expect that with the increasing density of APs and larger collection of in-store pictures over time, AutoLabel will be able to automatically label APs and accurately localize users to stores.

5.3 Evaluation

In this section, we evaluate the AutoLabel system. Below, first we summarize our findings and later describe the data collection and present the detailed results.

- The similarity between the text from a store and that from any other store, and likewise between the text from a store's website and that from other store's websites, is low, less than 0.2 in 95% cases.
- Overall, in 87% instances, the text from a store matched the most with the text from that store's website than any other store's website; when compared the text from a store against that from nearby stores within an area, the accuracy climbs above 94%.
- Even when a random set of stores are chosen as neighbors, to mimic a large-scale study, the average labelling accuracy of AutoLabel is above 90%.
- In general, around 10 random pictures with text from a store suffice for AutoLabel to distinguish that store from nearby stores with high accuracy.

- With simultaneous image clustering and labelling, the average labelling accuracy is around 63%.

- An app using AutoLabel localizes a user based on the heard APs to inside-store and outside-store with an accuracy of 77% and 91% respectively.

- Further refinements needed for unsupervised clustering of images and accurate localization of users.

A. Data Collection

We collect data from 40 stores: 18 of them inside a shopping mall, 10 are neighboring stores located along a street, and 6 are in other places, all in Champaign, IL; the remaining 6 stores are from a mall in San Jose, CA. Among them 35 are retail stores and 5 are restaurants. To collect in-store pictures efficiently, our crowdsourcers are asked to take videos inside these stores, and randomly extract image frames from these videos. This in a way mimics the in-store picture collection by many crowdsourcers over time. The number of pictures (with text) extracted from videos ranges from 6 to 217 across stores.

While collecting data, the crowdsourcers' smartphones collect the WiFi AP data in front of and inside the stores they visit. For street stores, the GPS location in front of each store is recorded; for the stores in shopping mall, the GPS coordinates in front of the gate of the shopping mall are saved. The GPS information need not be accurate, as it is only used to filter the candidate set during matching to improve the efficiency of the system. The crowdsourcers also record the semantic names of the places they visit, which serve as the ground truth in evaluating AutoLabel.

B. Similarity between Stores

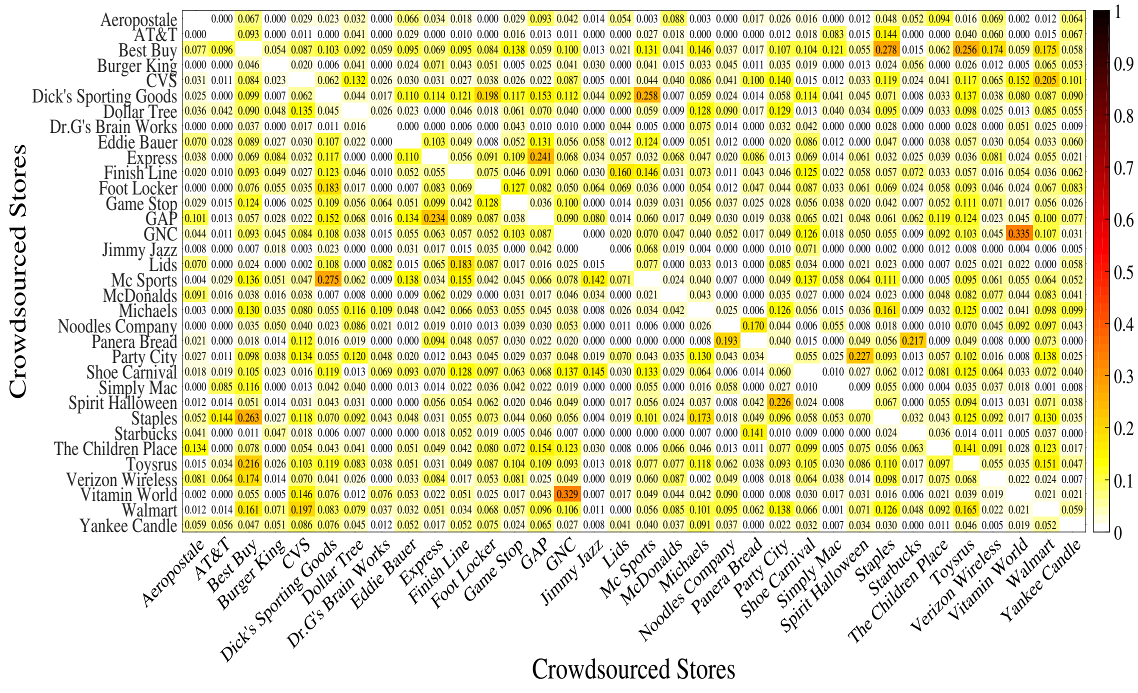
To distinguish a store from other stores using AutoLabel, it is necessary that: i) the text in those stores are dissimilar; and ii) the text on the websites of those stores are dissimilar. To check whether these necessary conditions for AutoLabel hold, we study the similarity between the text inside stores and also the similarity between the text of the websites of these stores.

Fig. 5.4(a) shows the similarity between the text in all stores in Champaign, IL. Although the stores with similar business have higher similarity, the overall similarity is low, less than 2% cases have similarity scores above 0.2. Fig. 5.4(b) illustrates the similarity of the text on the websites of these stores. It is evident that while the homepages of stores having similar business have more common text, the overall similarity of web text is still low. This gives us confidence that it is feasible to distinguish stores based on their store text and web text.

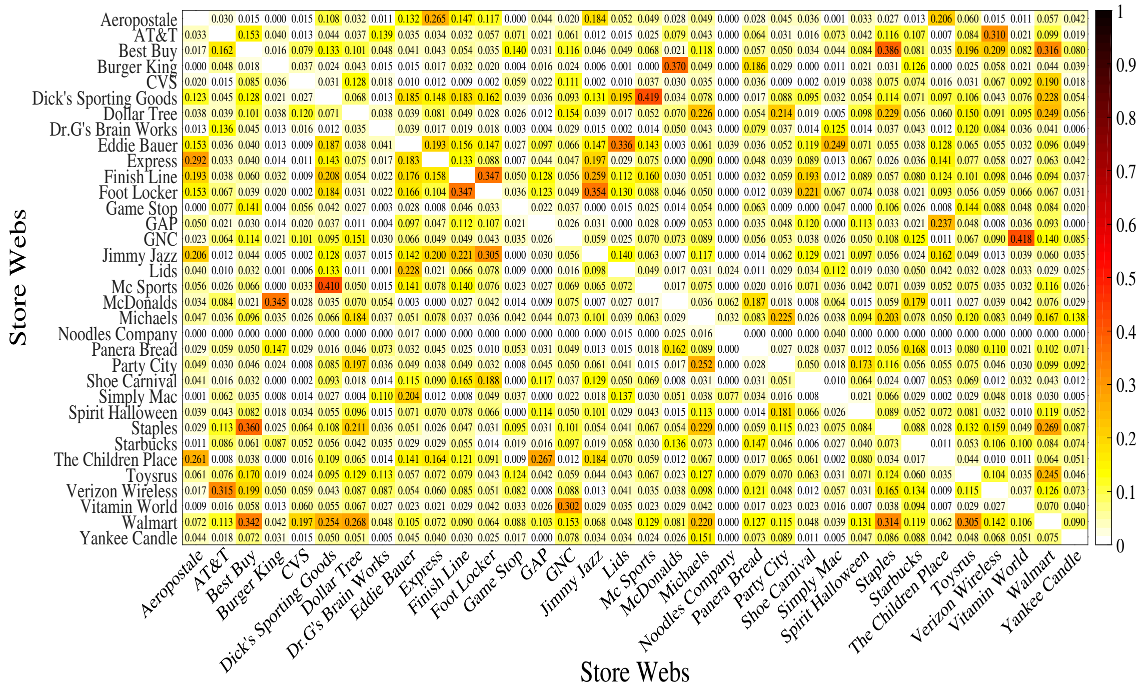
C. Matching Store Text with Web Text

We consider two cases: i) excluding store name and ii) including store name, in case it appears in the text of store images. The intention is to see how well a store can be distinguished based on the text alone without taking advantage of the presence of store name in the text.

Fig. 5.5 shows the matching result between the store text and web text from the 18 stores in the mall in Champaign. To make the best matching pairs evident, we plot the similarity scores normalized by the highest score, such that the darker the color is, the better the match is. Even when the store name is excluded from the text (Fig. 5.5(a)), the matching score between the store text and the web text of that store is the highest in 16 out of 18 cases, yielding 89% matching accuracy. With store name included, when it happens to appear in the store text (Fig. 5.5(b)), matching accuracy goes up to 94%. There are, however, some instances of mismatching, par-

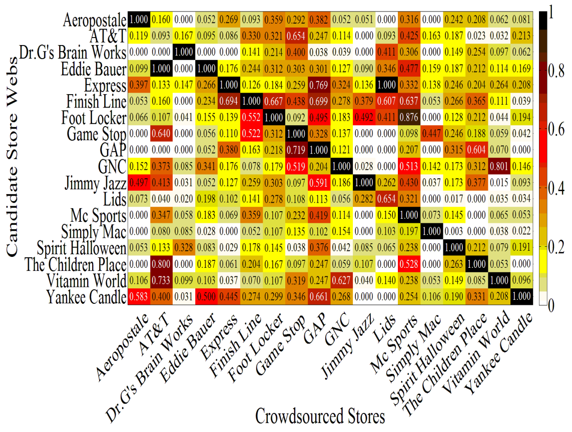


(a)

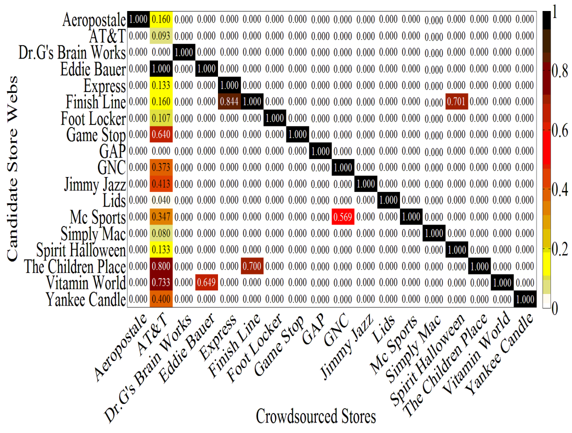


(b)

Figure 5.4: The similarity between: (a) the text in stores in Champaign, IL; (b) the web text of these stores.



(a)

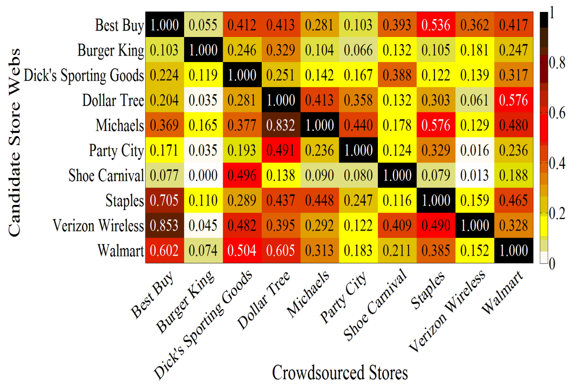


(b)

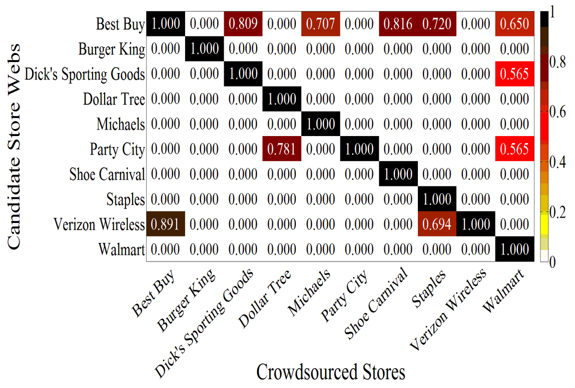
Figure 5.5: Matching between store text and web text for stores in the mall in Champaign, IL (similarity scores normalized by the highest similarity value): (a) excluding and (b) including the store name if it appears in the store text.

ticularly in case of the AT&T store. By reviewing the collected data, we find that the AT&T store in the shopping mall is quite small and we could only extract little text at/above eye-level from the pictures. Among the text, some words (e.g. "accessory") are common with other stores, which leads to the mismatches. This problem of text sparsity and potential remedies are discussed later in Section 5.5. We repeat the same matching procedure with the 10 stores along a street in Champaign and plot the results in Fig. 5.6. In this scenario, with or without the store name being part of the text, the matching accuracy is 100%.

In the above matching experiment, text from a store in the mall is compared against other stores in the mall, and likewise with the stores along a street. The reason is that, in practice, given some text from a store, AutoLabel does not need to compare with all the available web text in its database. Instead, it only needs to compare with the candidate stores from the area where the store text is collected. The rough GPS location helps to find the candidate stores in the expected area. Nevertheless, to study how well AutoLabel performs without GPS information and when all these stores are near each other, we perform the matching between all



(a)



(b)

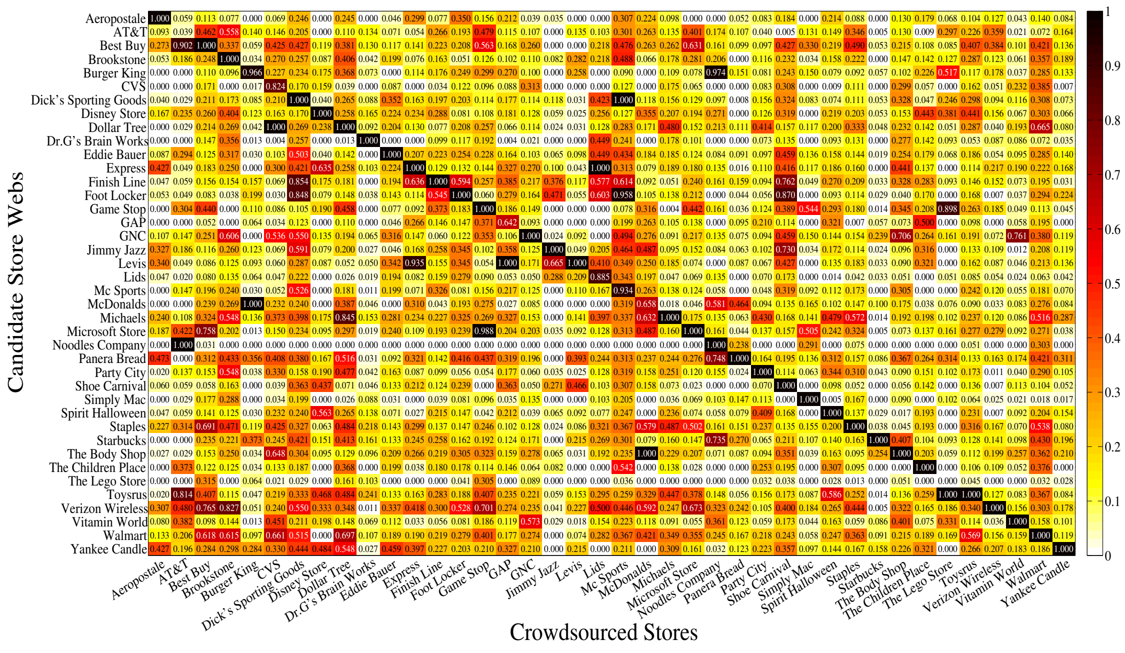
Figure 5.6: Matching between store text and web text for stores on the street (similarity scores normalized by the highest similarity value): (a) excluding and (b) including the store name if it appears in the store text.

40 stores, including those in San Jose. Fig. 5.7 shows that, even in this case, the matching accuracy is still high; 80% and 87% respectively without and with including store name in the text.

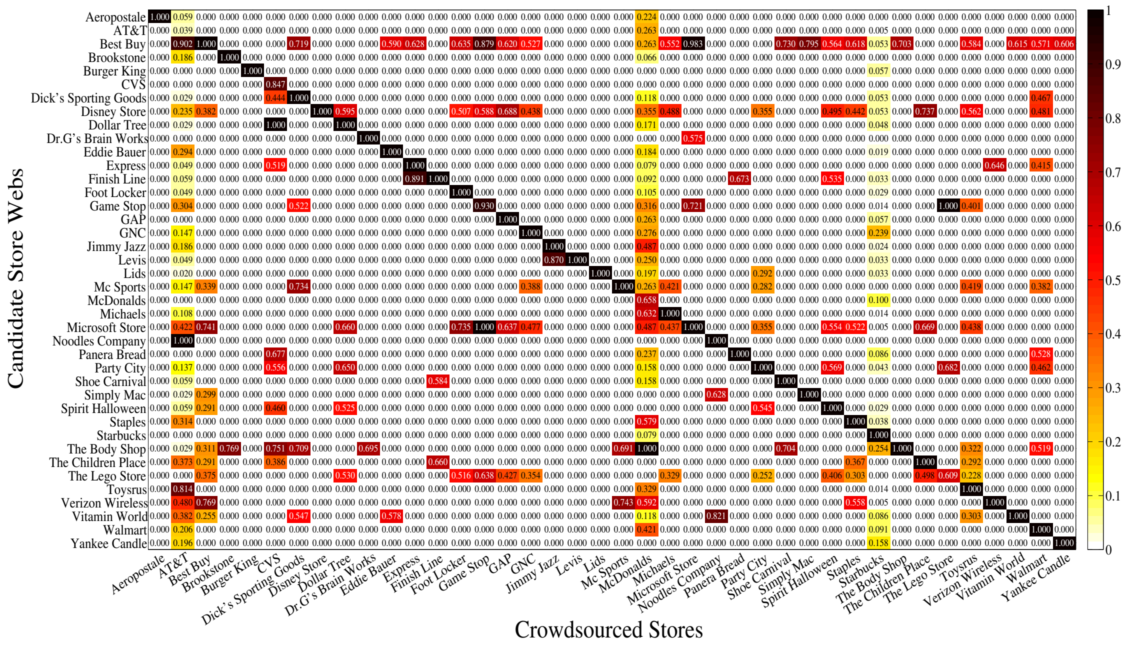
D. Matching with Limited Number of Pictures

To study the matching accuracy when different numbers of in-store pictures are available, we randomly select 5∼40 from the pictures taken at each store. For each number of in-store pictures, we randomly sample 20 sets. For each scenario, we calculate three accuracy metrics, whether the correct store's matching score is the top one, among the top two, or among the top three. Here and in the rest of the section, while matching, store name is included in the text if it appears in the selected images.

Fig. 5.8 shows the accuracy of matching with varying numbers of in-store pictures. In many stores, 10 random pictures with text allow AutoLabel to build reasonably confident mapping between in-store text and web text. When up to 20 useful in-store pictures are available, the labeling algorithm performs better, but it does not improve significantly from the 10-picture scenario.

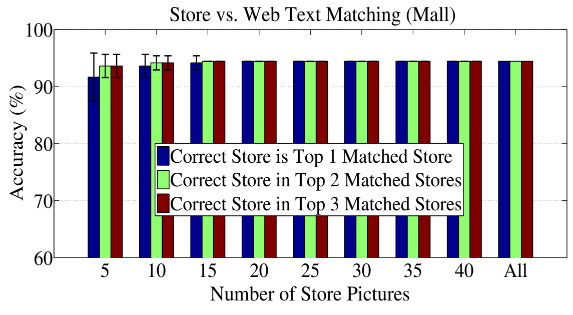


(a)

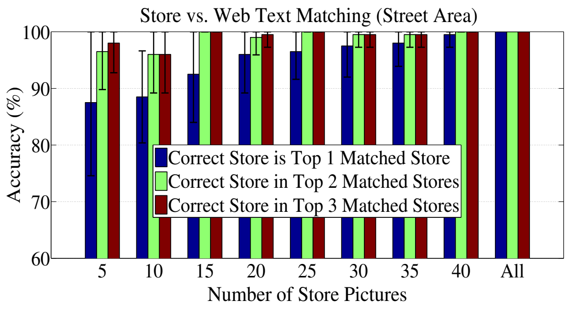


(b)

Figure 5.7: Matching between store text and web text for 40 stores (similarity scores normalized by the highest similarity value): (a) excluding and (b) including the store name if it appears in the store text.



(a)



(b)

Figure 5.8: Accuracy with varying number of pictures: (a) mall stores; (b) street stores.

E. Matching with Diverse Stores in an Area

Thus far, we have evaluated AutoLabel for the stores in two shopping malls in Champaign and San Jose, and also along a street in Champaign. But different cities/areas have different numbers and combinations of stores. To mimic those situations, we evaluate AutoLabel with randomly selected subsets of all the stores we collected. We considered areas of four sizes: 5, 10, 15 and 20 stores. For each area size, 100 non-duplicate subsets are randomly selected. Within each subset, 5 different numbers of pictures (i.e. 10, 20, 30, 40, all) are used for matching between these stores. Fig. 5.9 shows the distribution of accuracy for different area sizes. It indicates that when an area gets denser (i.e. more stores occur in an area), the matching accuracy goes down. This is because when more stores occur in the same area, the chance of similar stores occurring together increases, therefore, text from one store will have more chance to be matched with another store's web text. Nevertheless, the average matching performance is still high, above 90%.

F. Simultaneous Clustering And Labelling

Thus far, we have evaluated, given a set of images taken within a store, how accurately can AutoLabel recognize the name of that store. In the absence of such an a

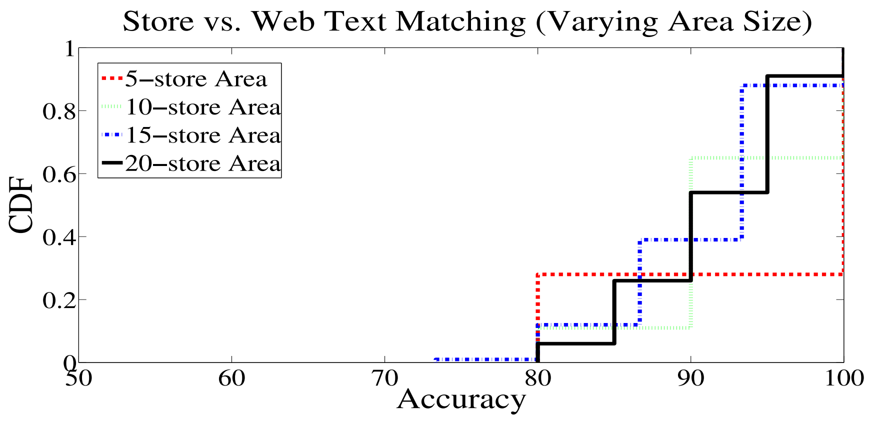


Figure 5.9: Accuracy with varying number of stores.

priori clustering of store pictures, AutoLabel performs simultaneous clustering and labelling following the Algorithm 2.

To evaluate the performance of this algorithm, we randomly select a subset of pictures from the crowdsourced data collected in the shopping mall in Champaign and run it to see whether the pictures could be clustered to their ground-truth store correctly. We conduct 200 such runs and plot the accuracy of resulting clustering in Fig. 5.10. We find that the accuracy on average is around 63% and it goes up to 80% in some instances. While these results may not be quite impressive, nevertheless they are promising. We believe there is much scope for refinement of this algorithm with more sophisticated analysis of the text, color/theme pattern seen in the pictures, and AP vector information. By leveraging such multi-dimensional information that is readily available, we can further discriminate pictures of different stores, and hence make the clustering and labelling algorithm more accurate.

G. Localization of Users with AutoLabel

After AutoLabel constructs the mapping from AP vector to store name, an end-user app could use it to automatically recognize stores by matching the detected APs with the labelled AP vectors. We developed a very simple app and installed the

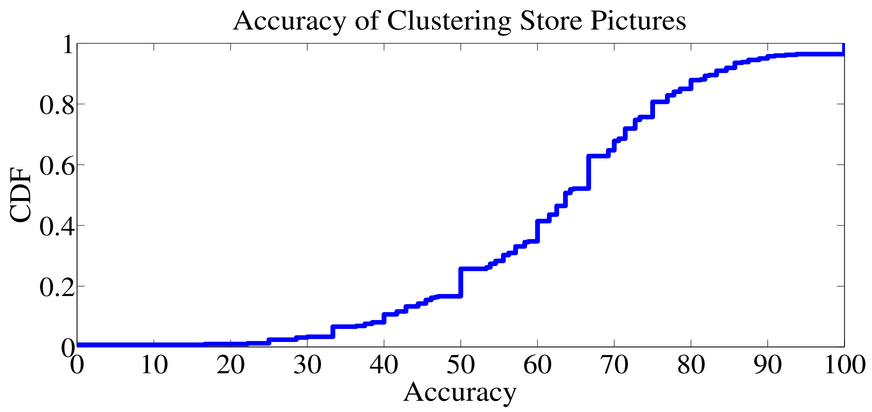


Figure 5.10: Clustering of store pictures.

mapping derived from AutoLabel based on the data we collected from the stores in two areas: mall and street in Champaign. From this, we assess the performance of Algorithm 3 for localizing users to stores.

We asked a student to use the app in these two areas. The app periodically scans for WiFi APs. It then compares the detected AP vector with the labelled AP vectors to find a matching store and reports it to the user. Once the app reports a store, the user evaluates it with three options: correct, wrong and unrecognized.

We consider two scenarios for performance evaluation. 1) Out-store recognition: If the app reports a store and it is within the user's sight, then that is deemed a correct match, otherwise a wrong match if it is out of the user's sight. If there are stores (in AutoLabel's map) within the user's sight, but the app does not recognize any of them, we conclude it as an unrecognized case. 2) In-store recognition: When a user is inside a store (which is in AutoLabel's map), only when the reported store is the same as the store currently the user is in, we conclude it as correct; wrong otherwise. If the user is in a store, after a WiFi scan, if the app neither reports this nor any other store, the result is unrecognized.

Fig. 5.11 shows that in-store localization accuracy is around 77%, while that for out-store scenario is around 91%. While further refinement is necessary, particularly for improving in-store localization accuracy, even the current version of AutoLabel is still valuable considering the usage in practice. For example, when a customer is walking in the hallway of a shopping mall, if the app could show a coupon with an accuracy of 91% for the stores in the customer's line of sight, it could benefit both the businesses and customers. Furthermore, as more images and the corresponding AP data get gathered overtime, the localization accuracy of AutoLabel naturally improves.

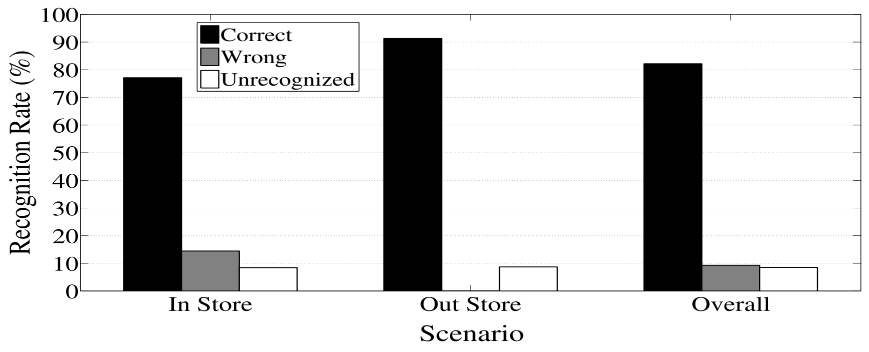


Figure 5.11: Performance of place recognition.

5.4 Related Work

With the rich body of work on localization, it may appear that labeling places automatically is a well-researched problem. Surprisingly, relatively few proposals focus on the "automatic labeling" aspect, which we discuss below.

ISIL [82] tries to infer place names by mining WiFi SSIDs. While indeed a creative idea, it does not generalize – far too many stores assign non-meaningful SSIDs. Our initial attempts at this idea showed that in Champaign less than 30% of stores could be identified; even in [82] the accuracy is understandably low at 60%. UnLocIn [83] also leverages WiFi data to infer user's location by querying WiFi databases. It studies from a security perspective, where an attacker attempts to determine the user's place.

Some researchers have indeed focused on the automatic labeling problem. Authors in [84] utilize GPS data to infer the street address of the place the user visited, and then reverse geo-codes this address to find the place (e.g., Starbucks). While this may not be accurate (due to GPS errors indoors), authors correlate with calendar, address book, and even credit card transactions to strengthen precision – a privacy concern. Another approach in [85] tries to mine geo-tagged tweets to infer the place the user is in. The same authors in [86] also use phone call and SMS records to automatically help users check-in into new places (i.e., automatic foursquare). Several other works attempt automatic place identification based on analysis of user trajectories, frequency and timing of visits, and other sensor data, converting them into a functional place (e.g., office, gym) [87–92].

The work in [93] localizes users based on text signs. It requires users to manually input the text signs they see, matches them against a GPS-tagged sign database to obtain the GPS value. [94] tries to identify the stores that appear in a photo by matching it against the images of the exteriors of the nearby stores extracted from the web. Both approaches require user intervention to localize self or identify a store from outside, whereas AutoLabel aims to automatically identify the store a user is in.

Perhaps closest to this paper is the work in [95], which tries to connect the text in crowdsourced pictures with the posts in social networks to infer business names. While similar to AutoLabel in spirit, [95] utilizes two disparate information sources: i) crowdsourced pictures, whose content reflects curated choices of the business owner; ii) social-network posts about the business, whose content reflects the unstructured and sometimes unrelated views of the netizens, making the correlation weak – perhaps the reason for its low accuracy of 38%. Given that offline version (i.e. the physical store) and online version (i.e. the store's website) of the same business are curated to have similar look and content, we believe AutoLabel holds better promise for semantic localization.

5.5 Limitations and Future Work

We now discuss scenarios that pose practical challenges for deploying AutoLabel. Considering that the core idea behind AutoLabel is the correlation of store and web text, lack of a website for a store and sparsity of text in stores or on websites adversely affect its performance. Furthermore, when the number of in-store pictures and the amount of text extracted from them is small, proper clustering of images becomes more critical. In the following, we discuss these concerns and potential remedies.

Web Presence: AutoLabel makes the general assumption that stores have their own webpages, from which text can be obtained. While most businesses indeed have online footprints, and the trend is certainly in that direction, we find some that do not, especially when a store is small. In such cases, one solution might be to utilize the "review page" for business in Google map or Yelp. Customer reviews, comments, basic descriptions could contain certain keywords, which are likely to have some correlation with store text. We leave the investigation and evaluation of such cases to future work.

Web Text: In our current implementation, AutoLabel only extracts text from "category/menu" items on webpages. We notice that some webpages do not have such an organization – typical for small businesses that sell few items not amenable to extensive categorization. For these webpages, web texts may not be rich and could derail AutoLabel. Moreover, webpages "express" the store in various other ways, including color themes, phrases, advertisements, and pictures. It is possible that physical stores and webpages also "correlate" in more subtle dimensions. A holistic understanding of the convergence and divergence (of the physical and online world) merits research attention, and is left to future work.

Image Clustering: With proper clustering, AutoLabel can accurately identify a store, even when only a few pictures from that store are available. While the current clustering and labeling performance of AutoLabel is inadequate for deployment, there is a lot of potential for improvement. Spatial relationship between stores can be exploited for clustering of images. Moreover, WiFi SSID mining, which alone is insufficient for labeling, could still provide some hints to improve the performance of AutoLabel. Overall, considering that our approach is complementary to existing approaches, they can work in conjunction towards automatic semantic localization.

5.6 Summary

Automatic user localization relies on electronic/digital information that can be sensed by devices (e.g., GPS, WiFi, ambient sound, etc.). However, for many applications, the notion of location is only useful in its semantic form (e.g., Starbucks, Pizza Hut, airport) – the sensor data and physical coordinates are pointless. In an attempt to bridge this gap, the technical question pertains to automatically labeling sensor data (or physical coordinates) with their semantic names. We observe that words – from a physical store and its corresponding online webpage – can serve as an effective bridge between the digital and semantic world. By extracting words from in-store pictures and correlating with web-words, it is possible to map pictures to their places, ultimately labeling WiFi APs with the name of the place. We believe this could be a scalable approach to plug an important hole in the broad localization puzzle.

Chapter 6

Conclusion

This dissertation shows how the information describing the characteristics of the same components and movement in transportation and shopping activities could be sensed by different entities from different perspectives, which thereby could be correlated to provide position information to benefit people and organizations involved in these activities. Following this principle, this work proposes two correlated systems, OmniView and DriverTalk, to help drivers learn about the relative positions of surrounding vehicles and also enable targeted communication between the drivers moving together on the road. Besides, an efficient and seamless billing mechanism, RideSense, is designed and implemented to improve existing public transit systems. This dissertation also presents AutoLabel for positioning customers with regard to businesses in shopping activities. It enables customers' smartphones automatically recognize the stores the customers are in or nearby, which provides a bridge for the information flow between customers and business owners. With these solutions, drivers could travel in a safer and more relaxing way; passengers of public transit systems could have better bus-riding experience; customers could gain better shopping experience. Overall, the works presented in this dissertation address several positioning problems in transportation and shopping areas, which shed some light on the future development of automobile, traffic systems and retailing.

Bibliography

[1] Rufeng Meng, Srihari Nelakuditi, Song Wang, and Romit Roy Choudhury. Omniview: A mobile collaborative system for assisting drivers with a map of surrounding traffic. In *2015 IEEE International Conference on Computing, Networking and Communications (ICNC)*, pages 760–765.

[2] Rufeng Meng, Romit Roy Choudhury, Song Wang, and Srihari Nelakuditi. Drivertalk: Enabling targeted communication between drivers. In *2016 IEEE International Conference on COMmunication Systems and NETworkS (COMSNETS)*.

[3] Rufeng Meng, David Grömling, Romit Roy Choudhury, and Srihari Nelakuditi. Ridesense: Towards ticketless transportation. In *Vehicular Networking Conference (VNC)*. IEEE, 2016.

[4] Rufeng Meng, Sheng Shen, Romit Roy Choudhury, and Srihari Nelakuditi. Matching physical sites with web sites for semantic localization. In *Proceedings of the 2nd workshop on Workshop on Physical Analytics*, pages 31–36. ACM, 2015.

[5] Rufeng Meng, Sheng Shen, Romit Roy Choudhury, and Srihari Nelakuditi. Autolabel: labeling places from pictures and websites. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp)*, pages 1159–1169.

[6] Smartphone market share. `http://goo.gl/J0xRbY`.

[7] Smart car wars: Ford, gm throw open their doors to app developers. `http://www.forbes.com/sites/joannmuller/2013/01/07/smart-car-wars-ford-gm-throw-open-their-doors-to-app-developers/`.

[8] Carplay. `https://www.apple.com/ios/carplay/`.

[9] Apple pay. `http://www.apple.com/apple-pay/`.

[10] Traffic safety facts. `http://www-nrd.nhtsa.dot.gov/Pubs/812196.pdf`.

[11] Surround view systems. http://www.mobileye.com/technology/applications/surround-view-systems/.

[12] Gps accuracy. http://www.gps.gov/technical/ps/2008-SPS-performance-standard.pdf.

[13] Differential global positioning system. http://en.wikipedia.org/wiki/Differential_GPS.

[14] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2001*, volume 1, pages I–511.

[15] Rainer Lienhart and Jochen Maydt. An extended set of haar-like features for rapid object detection. In *2002 IEEE International Conference on Image Processing*, volume 1, pages I–900.

[16] Mit cbcl streetscenes. http://cbcl.mit.edu/software-datasets/streetscenes/.

[17] Tugraz database. http://www.emt.tugraz.at/~pinz/data/GRAZ_02/.

[18] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88, 2010.

[19] Caltech cars 1999,2001. http://www.vision.caltech.edu/archive.html.

[20] John B Kenney. Dedicated short-range communications (dsrc) standards in the united states. *Proceedings of the IEEE*, 99, 2011.

[21] Daniel Jiang, Qi Chen, and Luca Delgrossi. Optimal data rate selection for vehicle safety communications. In *Proceedings of the fifth ACM international workshop on VehiculAr Inter-NETworking*, pages 30–38. ACM, 2008.

[22] Cooperative its for all: Enabling dsrc in mobile devices. http://docbox.etsi.org/workshop/2013/201302_ITSWORKSHOP/S05_COOPERATIVEITSandFUTUREASPECTS/QUALCOMM_SUBRAMANIAN.pdf.

[23] How snapdragon and honda are working to save lives with smartphones. https://www.qualcomm.com/news/snapdragon/2015/06/16/how-snapdragon-and-honda-are-working-save-lives-smartphones.

[24] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60, 2004.

[25] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Computer vision–ECCV 2006*, pages 404–417. Springer.

[26] Stefan Leutenegger, Margarita Chli, and Roland Y Siegwart. Brisk: Binary robust invariant scalable keypoints. In *2011 IEEE International Conference on Computer Vision (ICCV)*, pages 2548–2555.

[27] Alexandre Alahi, Raphael Ortiz, and Pierre Vandergheynst. Freak: Fast retina keypoint. In *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 510–517.

[28] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: an efficient alternative to sift or surf. In *2011 IEEE International Conference on Computer Vision (ICCV)*, pages 2564–2571.

[29] Sumo simulator. http://sumo-sim.org/.

[30] The network simulator - ns-2. http://www.isi.edu/nsnam/ns/.

[31] Night vision system. http://goo.gl/gGAx62.

[32] Tarak Gandhi and Mohan M Trivedi. Dynamic panoramic surround map: motivation and omni video based approach. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops, 2005. CVPR Workshops*, pages 61–61.

[33] Hong Cheng, Zicheng Liu, Nanning Zheng, and Jie Yang. Enhancing a driver's situation awareness using a global view map. In *2007 IEEE international conference on Multimedia and expo*, pages 1019–1022.

[34] Bo Xu, Ouri Wolfson, and Hyung Ju Cho. Monitoring neighboring vehicles for safety via v2v communication. In *2011 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, pages 280–285.

[35] Shahram Rezaei, Raja Sengupta, Hariharan Krishnan, Xu Guan, and Raman Bhatia. Tracking the position of neighboring vehicles using wireless communications. *Transportation Research Part C: Emerging Technologies*, 18(3):335–350, 2010.

[36] Andrea Corti, Vincenzo Manzoni, Sergio M Savaresi, Mario Santucci, and Onorino Di Tanna. A centralized real-time advanced driver assistance system based on smartphones.

[37] James A Misener, Raja Sengupta, and Hariharan Krishnan. Cooperative collision warning: Enabling crash avoidance with wireless technology. In *12th World Congress on ITS*, volume 3. Citeseer, 2005.

[38] Jie Yang, Jie Wang, and Benyuan Liu. An intersection collision warning system using wi-fi smartphones in vanet. In *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, pages 1–5.

[39] Lane Width. `http://safety.fhwa.dot.gov/geometric/pubs/mitigationstrategies/chapter3/3_lanewidth.cfm`.

[40] Driving and Tailgating FactSheet. `http://www.tdi.texas.gov/pubs/videoresource/fsriskstailgati.pdf`.

[41] 70% noise pollution due to honking. `http://timesofindia.indiatimes.com/city/gurgaon/70-noise-pollution-due-to-honking/articleshow/21473250.cms`.

[42] Shanghai Drivers Banned From Honking Their Horns! `http://www.carscoops.com/2007/07/shanghai-drivers-banned-from-honking.html`.

[43] Ford's 'talking cars' could reduce crashes, fuel use. `http://gigaom.com/2011/06/01/fords-talking-cars-could-reduce-crashes-fuel-use/`.

[44] Swarun Kumar, Lixin Shi, Nabeel Ahmed, Stephanie Gil, Dina Katabi, and Daniela Rus. Carspeak: a content-centric network for autonomous driving. *ACM SIGCOMM Computer Communication Review*, 42(4):259–270, 2012.

[45] Stephen Smaldone, Lu Han, Pravin Shankar, and Liviu Iftode. Roadspeak: enabling voice chat on roadways using vehicular social networks. In *Proceedings of the 1st Workshop on Social Network Systems*, pages 43–48. ACM, 2008.

[46] Wenjie Sha, Daehan Kwak, Badri Nath, and Liviu Iftode. Social vehicle navigation: integrating shared driving experience into vehicle navigation. In *Proceedings of the 14th Workshop on Mobile Computing Systems and Applications*, page 16. ACM, 2013.

[47] Anup Doshi and Mohan M Trivedi. Communicating driver intents: a layered architecture for cooperative active safety applications. In *2010 13th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 373–378.

[48] Uichin Lee, Jiyeon Lee, Joon-Sang Park, and Mario Gerla. Fleanet: A virtual market place on vehicular networks. *Vehicular Technology, IEEE Transactions on*, 59(1):344–355, 2010.

[49] Felica. `http://www.sony.net/Products/felica/business/data/FeliCa_E.pdf`.

[50] Apple pay - transport for london. `https://tfl.gov.uk/fares-and-payments/contactless/other-methods-of-contactless-payment/apple-pay`.

[51] Shuangquan Wang, Canfeng Chen, and Jian Ma. Accelerometer based transportation mode recognition on mobile phones. In *2010 Asia-Pacific Conference on IEEE Wearable Computing Systems (APWCS),*, pages 44–46.

[52] Luca Bedogni, Marco Di Felice, and Luciano Bononi. By train or by car? detecting the user's motion type through smartphone sensors data. In *Wireless Days (WD), 2012 IFIP*, pages 1–6. IEEE.

[53] Samuli Hemminki, Petteri Nurmi, and Sasu Tarkoma. Accelerometer-based transportation mode detection on smartphones. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, page 13. ACM, 2013.

[54] Kartik Sankaran, Minhui Zhu, Xiang Fa Guo, Akkihebbal L Ananda, Mun Choon Chan, and Li-Shiuan Peh. Using mobile phone barometer for low-power transportation context detection. In *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*, pages 191–205. ACM, 2014.

[55] Dongyoun Shin, Daniel Aliaga, Bige Tunçer, Stefan Müller Arisona, Sungah Kim, Dani Zünd, and Gerhard Schmitt. Urban sensing: Using smartphones for transportation mode classification. *Computers, Environment and Urban Systems*, 53, 2015.

[56] Hao Xia, Yanyou Qiao, Jun Jian, and Yuanfei Chang. Using smart phone sensors to detect transportation modes. *Sensors*, 14(11):20843–20865, 2014.

[57] Carl De Boor, Carl De Boor, Etats-Unis Mathématicien, Carl De Boor, and Carl De Boor. *A practical guide to splines*, volume 27. Springer-Verlag New York, 1978.

[58] Oliver J Woodman. An introduction to inertial navigation. *University of Cambridge, Computer Laboratory, Tech. Rep. UCAMCL-TR-696*, 14:15, 2007.

[59] Lei Kang, Bozhao Qi, and Suman Banerjee. A wireless-based approach for transit analytics. In *Proceedings of the 17th International Workshop on Mobile Computing Systems and Applications*, pages 75–80. ACM, 2016.

[60] Cheng Bo, Xiang-Yang Li, Taeho Jung, Xufei Mao, Yue Tao, and Lan Yao. Smartloc: Push the limit of the inertial sensor based metropolitan localization using smartphone. In *Proceedings of the 19th annual international conference on Mobile computing & networking*, pages 195–198. ACM, 2013.

[61] Shaohan Hu, Lu Su, Shen Li, Shiguang Wang, Chenji Pan, Siyu Gu, Md Tanvir Al Amin, Hengchang Liu, Suman Nath, Romit Roy Choudhury, et al. Experiences with enav: a low-power vehicular navigation system. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 433–444.

[62] Arvind Thiagarajan, Lenin Ravindranath, Katrina LaCurts, Samuel Madden, Hari Balakrishnan, Sivan Toledo, and Jakob Eriksson. Vtrack: accurate, energy-aware road traffic delay estimation using mobile phones. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, pages 85–98. ACM, 2009.

[63] Arvind Thiagarajan, Lenin Ravindranath, Hari Balakrishnan, Samuel Madden, Lewis Girod, et al. Accurate, low-energy trajectory mapping for mobile devices. In *NSDI*, 2011.

[64] Vassilis Kostakos, Tiago Camacho, and Claudio Mantero. Towards proximity-based passenger sensing on public transport buses. *Personal and ubiquitous computing*, 17(8):1807–1816, 2013.

[65] Sharmistha Chatterjee, Jukka K Nurminen, and Matti Siekkinen. Low cost positioning by matching altitude readings with crowd-sourced route data. In

Proceedings of the 10th International Conference on Advances in Mobile Computing & Multimedia, pages 169–178. ACM, 2012.

[66] Arvind Thiagarajan, James Biagioni, Tomas Gerlich, and Jakob Eriksson. Cooperative transit tracking using smart-phones. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, pages 85–98. ACM, 2010.

[67] Pengfei Zhou, Yuanqing Zheng, and Mo Li. How long to wait?: predicting bus arrival time with mobile phone based participatory sensing. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pages 379–392. ACM, 2012.

[68] Picture this: A fresh approach to photos. `https://googleblog.blogspot.com/2015/05/picture-this-fresh-approach-to-photos.html`.

[69] Apple icloud photo library. `http://www.apple.com/icloud/photos/`.

[70] Tmt predictions 2016 - photo sharing: trillions and rising. `http://goo.gl/zjjP6W`.

[71] Geotagging: Do more with your images and videos. `http://goo.gl/CkvGXR`.

[72] Geotag photos pro. `http://www.geotagphotos.net/`.

[73] Eye-fi adds geotagging via wi-fi. `http://goo.gl/q8pzQs`.

[74] Michael Grothaus. *My Photos for Mac*. Que Publishing, 2015.

[75] Subrata Goswami. *Indoor location technologies*. Springer Science & Business Media, 2012.

[76] E Garcia. Cosine similarity and term weight tutorial. *Information retrieval intelligence*, 2006.

[77] Stanford named entity recognition. `http://nlp.stanford.edu/software/CRF-NER.shtml`.

[78] Wordnet. `http://wordnet.princeton.edu/`.

[79] Google goggles. `www.google.com/mobile/goggles`.

[80] Amazon firefly. `https://developer.amazon.com/public/solutions/devices/fire-phone/docs/understanding-firefly`.

[81] Qiuxia Chen, Dik-Lun Lee, and Wang-Chien Lee. Rule-based wifi localization methods. In *IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, 2008. EUC'08*, volume 1, pages 252–258.

[82] Truc D Le, Thong M Doan, Han N Dinh, and Nam T Nguyen. Isil: Instant search-based indoor localization. In *Consumer Communications and Networking Conference (CCNC)*, pages 143–148. IEEE, 2013.

[83] Yuan Tian Le Nguyen, Sungho Cho, Wookjong Kwak, Sanjay Parab, Yuseung Kim, Patrick Tague, and Joy Zhang. Unlocin: Unauthorized location inference on smartphones without being caught.

[84] Juhong Liu, Ouri Wolfson, and Huabei Yin. Extracting semantic location from outdoor positioning systems. In *MDM*, page 73. Citeseer, 2006.

[85] Kiran Rachuri, Theus Hossmann, Cecilia Mascolo, and Sean Holden. What is this place? inferring place categories through user patterns identification in geo-tagged tweets. In *PerCom*, 2015.

[86] Deborah Falcone, Cecilia Mascolo, Carmela Comito, Domenico Talia, and Jon Crowcroft. Beyond location check-ins: Exploring physical and soft sensing to augment social check-in apps. In *MobiCase*, 2014.

[87] Jeffrey Hightower, Sunny Consolvo, Anthony LaMarca, Ian Smith, and Jeff Hughes. Learning and recognizing the places we go. In *UbiComp 2005: Ubiquitous Computing*, pages 159–176. Springer.

[88] John Krumm and Dany Rouhana. Placer: semantic place labels from diary data. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, pages 163–172.

[89] Zhenyu Chen, Yiqiang Chen, Shuangquan Wang, and Zhongtang Zhao. A supervised learning based semantic location extraction method using mobile phone data. In *2012 IEEE International Conference on Computer Science and Automation Engineering (CSAE)*, volume 3, pages 548–551.

[90] Moustafa Elhamshary, Moustafa Youssef, Akira Uchiyama, Hirozumi Yamaguchi, and Teruo Higashino. Transitlabel: A crowd-sensing system for automatic labeling of transit stations semantics. In *ACM Mobisys*, 2016.

[91] Mao Ye, Dong Shou, Wang-Chien Lee, Peifeng Yin, and Krzysztof Janowicz. On the semantic annotation of places in location-based social networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 520–528. ACM, 2011.

[92] Elena Ivannikova. Semantic place recognition for context aware services. 2012.

[93] Bo Han, Feng Qian, and Moo-Ryong Ra. Human assisted positioning using textual signs. In *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*, pages 87–92. ACM, 2015.

[94] Amir Roshan Zamir, Afshin Dehghan, and Mubarak Shah. Visual business recognition: a multimodal approach. In *ACM Multimedia*, pages 665–668. Citeseer, 2013.

[95] Yohan Chon, Yunjong Kim, and Hojung Cha. Autonomous place naming system using opportunistic crowdsensing and knowledge from crowdsourcing. In *ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2013.