SCALING DATA SHARING AMONG VEHICLES TO TACKLE LONG TAIL TRAFFIC SITUATIONS

by

HONGYU LI

A dissertation submitted to the

Graduate School–New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Computer Science

Written under the direction of

Marco Gruteser

and approved by

_____

_____

_____

_____

New Brunswick, New Jersey

May, 2021

ABSTRACT OF THE DISSERTATION

Scaling data sharing among vehicles to tackle long tail traffic situations

By Hongyu Li

Dissertation Director:

Marco Gruteser

Developing robust driver assistance and automated driving technologies requires an understanding of not just common highway and city traffic situations but also the long tail of unusual or rare traffic events (e.g., objects on the roadway and pedestrian crossing highway, etc.). Due to the rareness, variety, and complexity of the long tail traffic conditions, it is widely recognized that ensuring dependability under these situations remains a key challenge for automated vehicles. Specifically, the challenges of tackling the long tail traffic situations are threefold. (i) Due to the rareness of the long tail traffic events, understanding these events will require accumulating data collection in the order of billions of miles. Since most existing efforts to collect driving data build on a small fleet of highly instrumented vehicles that are continuously operated with test drivers, it is challenging to acquire the road data in such a large scale. (ii) Among the large amount of data needed, it is hard to automatically identify the unusual situations which are challenging to address and could lead to potential accidents. Although there exists a large body of work on abnormal driving event detection, they focus on detecting specific, known situations but cannot detect previously unknown unusual road events that are missing in the current set of test cases for automated vehicles. (iii) The complexity of diverse traffic situations makes a single vehicle hard to sense its surrounding comprehensively. Although vehicle-to-vehicle (V2V) communications provide a channel for point cloud data sharing, it is challenging to align

point clouds from two vehicles with state-of-the-art techniques due to localization errors, visual obstructions, and viewpoint differences.

To address such challenges, this thesis focuses on scaling data sharing among vehicles, which enables low-cost road data acquisition, unusual driving events identification and accurate vehicle to vehicle point cloud alignment. In particular, the proposed solution includes: (1) A light-weight sensing and driving data logging system that can derive internal driver inputs (i.e., steering wheel angles, driving speed and acceleration) and external perceptions of road environments (i.e., road conditions and front-view video) using a smartphone and an IMU mounted in a vehicle, which enables road data acquisition and sharing in heterogeneous driving scenarios crossing different vehicle models and drivers. (2) An automatic unusual driving events identification system, which can detect unusual situations through a two-pronged approach involving inertial monitoring of driver reactions and an autoencoder-based technique for detecting unusual video scenes. (3) A two-phase point cloud registration mechanism to fuse point clouds from two different vehicles, which focuses on key objects in the scene where the point clouds are most similar and infer the transformation from those. The systems are evaluated based on both experimental and simulation data, which shows accurate road data acquisition and unusual event identification, and tremendous point cloud fusion performance improvement when combining with state-of-art baselines.

Acknowledgements

This dissertation would not have been possible without the help and support of a large group of wonderful people to whom I owe a lot of gratitude.

First, I would like to express my deepest gratitude to my advisor Marco Gruteser. Marco is not only a tremendous advisor, but also a supportive mentor. He guided me through how to approach a research problem, how to shape a project from a unique perspective, how to think critically about potential solutions, etc. Even though there were many research and life difficulties during the past six years, he always generously shares his experiences and suggestions to mentor me through many crossroads during my PhD journey. I deeply feel fortunate to have Marco as my advisor.

I would also like to thank Prof. Richard Martin, Prof. Desheng Zhang and Dr. Fan Bai for serving on my qualifier and dissertation committee. Their insightful suggestions and valuable comments have greatly improve the quality and completeness of my work.

I am honored to have the opportunity to collaborate with and be mentored by a group of talent researchers, Dr. Hongsheng Lu and Dr. Bin Cheng from Toyota InfoTechnology Center, Dr.Mohammed Khider, Dr. Yu-Han Yang, Dr. Michael Fu, Dr.Frank Diggelen, and Dr. Sean Wan from Google, and Dr. Junghwan Rhee from NEC Labs. I really appreciate their time and advice during the collaboration and internships. Without their help, my work won't be so complete.

I am thankful for being part of WINLAB and for having all my dear friends and collaborators there including Luayang, Cagdas, Hansi, Mohamed, Ali, Viet, Shubham, Gorkem, Wuyang, Jakub, Michael, Jennifer, Ivan, Noreen, Elaine and Lisa.

Words cannot describe my gratitude to my family. I really appreciate the unconditional love and support from my wife, Xinran Hu. She is always like the sunshine in my world, brightening and encouraging me. From studying in east and west coast separately, to our

marriage and had our first baby girl, I'm so lucky to have you. Special thanks to my parents and Xinran's parents for the strong support. I won't be be afraid of any difficulties because I know my family has my back. Covid-19 makes the world too tough, but I'm so lucky to have you all.

Dedication

To my wife, Xinran Hu

my daughter, Elena Li

and my parents, Jianhui Li and Li Liu

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Advanced driver assistance systems and, in particular, automated driving offer an unprecedented opportunity to transform the safety, efficiency, comfort, and economics of road travel. This has led many organizations in the computing and transportation domains to focus on self-driving technology research. Although a number of prototypes illustrate impressive performance, it remains widely recognized that ensuring dependability under varied traffic conditions remains a key challenge. Self-driving vehicles have to operate safely even under the long tail of unusual or rare traffic events that are challenging to address and could lead to potential accidents. While each such event may be rare, in aggregate they represent a significant risk that technology must address to develop truly dependable automated driving and traffic safety technologies. The average human driver achieves on the order of almost 100 million vehicle miles traveled per fatality. Demonstrating driving performance at an above-average, advanced human driver level will therefore require successfully avoiding fatalities with unusual events that might be encountered within a billion miles of driving. This motivates the need for scaling the road dataset to billions of miles of driving so that they contain a representative set of such unusual and rare events.

Among the billions of miles of driving data that need to be collected, it is challenging to automatically identify the unusual events and corner cases among them. Manual inspection of collected data to flag unusual driving events is one possible solution but will require plenty of extra effort, amplify privacy concerns, and increase storage and networking overhead for collecting all data. In addition, simply detecting discrepancies between human drivers' input and self-driving system's choice is not reliable, due to the many degrees of freedom in navigating a road. Although there exists a large body of work on abnormal driving

event detection which can capture specific known situations, determine previously unknown unusual event patterns that are missing in the current set of test cases for automated vehicles remains a challenge.

Although the aggregated unusual events can improve the robustness of single-agent automated driving system, vehicles still have to contend with (i)physical occlusions, in which objects are blocked by others and are only partially observable or unobservable; (ii) sensing limitations, including field of view, resolving power, or lighting conditions that may limit the sensing range and quality. Fusing information among vehicles has the potential to overcome such limitations by sharing observations across a V2V network and merging them across different vehicles, since such physical occlusions and sensing limitations from one perspective can often be easily addressed when viewing the scene from a different perspective. Although there has been extensive research on point cloud alignment/registration, the state-of-the-art methods cannot be directly applied to align the pairwise point clouds from vehicles, as they require a large overlapping ratio between point cloud pairs. Due to occlusions from surrounding objects or vehicles perceiving the scene from different directions, the observations from different vehicles usually have little overlap ratio and fail to be aligned by the state-of-the-art point cloud registration algorithms.

1.2 Thesis Contribution

The long tail traffic situations could be defined as the rarely happened, complex scenarios, which are challenging to handle for human and autonomous drivers. This dissertation tries to tackle a subset of such events, which can be classified into 2 categories based on the root cause of their difficulty. One category is challenging due to its rareness, which is hard to capture and may lead to insufficient training samples. The other category's difficulty comes from the complexity of the scene, which will make some objects to be partial or completely invisible. The research presented in this thesis aims to tackle such long tail events by answering the following questions: *Can we sense and identify long tail traffic situations accurately by developing low-cost inertial and vision sensing system and fusing point clouds with surrounding vehicles?*

Specifically, this thesis answers the above questions by addressing the following three sub-questions:

(1) Due to rareness of long tail traffic situations, it is hard for existing testing fleets to discover and collect various unusual events since they only include limited number of vehicles. Scaling the data collection to take advantage of daily driving vehicles will help to accumulate the long tail traffic situations, so that the data collected from different drivers and vehicles can be shared to build more robust driving assistance systems. However, it is challenging to record the information accurately based on different vehicles types and manufactures with a minimal set of devices. In order to address this challenge, this thesis firstly tries to answer *how to develop low-cost and scalable system to collect driver inputs and road environments based on mobile devices?*

(2) Although the proposed data collection system can accelerate the traffic events gathering, it remains challenging to automatically identify the long tail events from a large number of collected trips. In addition, the uploading and storage of the information may require tremendous bandwidth and cause drivers' privacy concern. This raises the question of whether it is possible to identify the small fraction of useful data that represents corner cases and challenging situations through pre-processing of the inertial and visual information inside the vehicle, which would allow uploading only these critical events. Therefore, this thesis secondly addressed the question that *how can we automatically identify the long tail traffic situations through inertial and vision sensing based on on-board devices?*

(3) Scaling the collection and identification of unusual driving events can help to understand and analyze the scenarios when such events happened. However, it remains challenging for single vehicle to perform comprehensive perception under these complex scenarios. For example, some traffic participants cannot be detected by a vehicle, because they are partially or fully blocked by other surrounding objects. To compromise the limitation of single viewpoint perception, fusing information between vehicles can combine the sensing information from different perspective. Specifically, point clouds which are generated by stereo cameras and LiDARs can be fused from different vehicles to generate a more comprehensive perception of surrounding scene. However, due to the complexity of traffic scenes, state of art of point cloud fusion algorithms can not align point clouds accurately under

complex intersection scenarios. Therefore, this thesis thirdly tries to answer *can we improve point cloud registration accuracy between vehicles by determining the overlapping observation area?*

1.2.1 Contribution Summary

To address the above questions, the key systems and contributions of this thesis are:

- Proposed a low-cost yet reliable solution, BigRoad, that can derive internal driver inputs (i.e., steering wheel angles, driving speed and acceleration) and external perceptions of road environments (i.e., road conditions and front-view video) using a smartphone and an IMU mounted in a vehicle.
- Designed a scalable solution to detects unusual events through inertial sensing of sudden human driver reactions and rare visual events through a trained autoencoder deep neural network.
- Proposed a two-phase point cloud registration mechanism to fuse point clouds which focuses on key objects in the scene where the point clouds are most similar and infer the transformation from those.

1.2.2 Low-cost reliable driver inputs and road environment data collection

To our knowledge, we are the first to analyze whether driving data useful for self-driving research can be collect with a minimal set of inertial and video add-on devices, in contrast to the existing work with significant instrumentation for scaling data acquisition. Steering wheel angle estimation algorithms are developed by leveraging low-cost sensing devices (i.e., inertial measurement units (IMUs) and smartphones). The key novelty of this approach is that it extracts steering wheel rotation by eliminating vehicle movement from the steering-wheel-mounted sensor measurements. Besides, vehicle speed sensing algorithms is introduced to combine GPS with speed delay shifting, and an acceleration based complementary filter. We further devise an acceleration-based road condition estimator to enhance road condition awareness under poor lighting conditions and to illustrate further uses of the

sensed data. Collecting and analyzing 40 hours of driving data to determine the accuracy of the estimation techniques and to demonstrate that the fine-grained driving data provided by the proposed framework achieves comparable performance in an automated vehicle steering algorithm.

1.2.3 Scalable automatic unusual driving event identification system

We introduce, to our knowledge, the first scalable unusual events identification and collection approach for self driving research and development, which can be deployed to human driven vehicles, instead of highly-instrumented vehicles. Beyond that, the unusual events identification system we proposed can detect a variety of corner cases including both challenging situations for human drivers and unusual video for self driving models. Besides, the design of efficient unusual imagery detectors for low cost in-vehicle devices can limit the necessary video uploads, conserve bandwidth and reduce privacy concerns. We analyzed more than 120 hours of driving data to evaluate the accuracy of unusual events identification and demonstrate the potential of these detected events to improve the performance of self-driving algorithms.

1.2.4 Two-phase point cloud registration mechanism for connected vehicle point cloud fusion

We propose the first system which can accurately align point cloud pairs under complex traffic conditions, such as scenes with various occlusions and large view angle difference1 (e.g., 90°, 180°). Specifically, we propose a technique to identify co-visible objects by combining multiple similarity metrics obtained in 3D object detection results to distinguish co-visible objects from single-visible objects. The fusion accuracy is improved when point cloud registration is focused on the co-visible object with the overlapping area among the views. We evaluate our system based on both synthetic scenes and real-world experimental data across highway and intersection scenarios and show that it can improve point cloud registration algorithms with a significant margin.

Chapter 2

Background

This thesis discusses scaling road data to tackle long tail traffic situations. To better understand the high costs of long tail traffic events collection, we firstly survey the existing autonomous vehicle prototypes. Since our data collection and unusual events identification system are designed to mostly benefit end to end automated steering system, we introduce the high level design of automated steering system. Besides, the existing V2V communication will also be discussed since it enables various data sharing between vehicles. More detailed related work sections are also provided in the remaining chapters.

2.1 Autonomous Vehicle Prototypes

With the recent development of automated driving, more and more autonomous vehicle prototypes are operating on road. These automated testing fleets usually instrumented various sensors, including camera, LiDAR, radar, ultrasonic sensor, etc.

Camera is one of the most critical components for the visual perception of automated testing fleets, which can generate the video stream at 10–30 fps to capture important objects such as traffic light, traffic sign, obstacles, etc., in real time. In addition to camera, LiDAR is another important sensor which measures the distance between vehicle and obstacles by actively illuminating the obstacles with laser beams. Typically, a LiDAR system scans the surrounding environment periodically and generates multiple measurement points. This "cloud" of points can be further processed to compute a 3D map of the surrounding environment. Alternatively, a stereo camera can be used to interpret the 3D environment. It is composed of two or more individual cameras. Knowing the relative spatial locations of all individual cameras, a depth map can be computed by comparing the difference between multiple images from different cameras. Therefore, the distance of an object in the scene

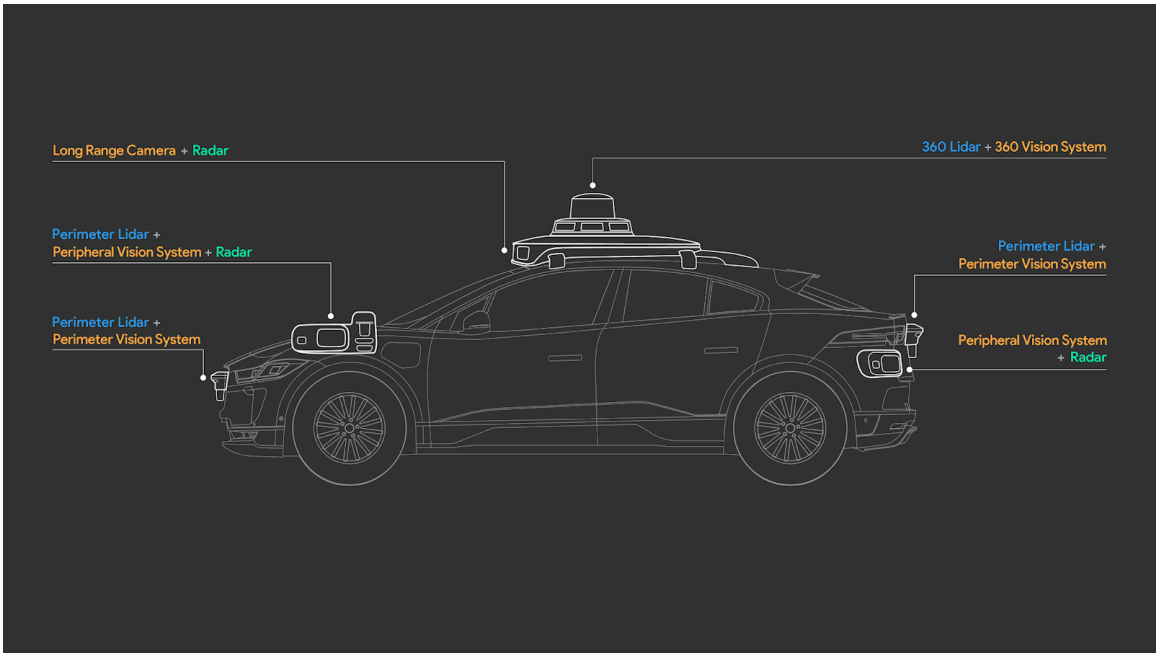


Figure 2.1: Waymo 5th generation hardware stack. [1]

can be estimated. Besides camera and LIDAR, radar and ultrasonic sensors are also widely used to detect obstacles. Their detection areas can be short-range and wide-angle, mid-range and wide-angle, and long-range and narrow-angle. For applications such as crashing detection and blind spot detection, a short detection range of 20–30 m is commonly used. For other applications such as cruise control, a long detection range of 200m is required. Ultrasonic sensors are similar to radars, but they use high-frequency sound waves, instead of radio waves, to detect objects. Both radars and ultrasonic sensors do not capture the detailed information of an obstacle (e.g., color, texture, etc.) and cannot classify the obstacle into different categories (e.g., vehicle, pedestrian, etc.).

One of the largest testing fleets on road is Waymo, which refers to its hardware suite for autonomous driving as Waymo Driver [1]. As shown in figure 2.1, it includes a 360 degree LiDAR on the roof and another 4 perimeter LiDARs (one in the front, one in the back, and one each left and right over the front tires). The last two LiDARs are supplemented by a radar. Also on the roof there are two radars facing forward and sideways, as well as on the rear left and right). Thus the radars also provide a 360 degree view. The system is supplemented by 29 cameras, 16 of which are located on the roof structure under LiDAR and several more facing forward in the roof box, some of which can see more than 500 meters

in high resolution. Perimeter cameras mounted around the car cover blind spots and can also see in the near infrared range. The 16 cameras under LiDAR are also all equipped with windscreen wipers that can wipe off raindrops and dirt with windscreen wiper water. It is estimated that the current sensor stack, as used by most off-the-shelf self-propelled technology developers, costs between $150,000 and $250,000 US dollars [3]. The high costs of the instrumented testing fleets brings challenges to further scale the data collection.

2.2 End to End Automated Steering System

End to end automated steering system is proposed by Nvidia [2] which is a convolutional neural network (CNN) to map raw pixels from a single front-facing camera directly to steering commands. This end-to-end approach proved powerful lane keeping capability in real world experiments. With minimum training data from humans the system learns to drive in traffic on local roads with or without lane markings and on highways. It also operates in areas with unclear visual guidance such as in parking lots and on unpaved roads.

Specifically, the system collects inputs from three cameras mounted behind the windshield of the vehicle. Time-stamped video from the cameras is captured simultaneously with the steering angle applied by the human driver. The collected single images sampled from the video, paired with the corresponding steering command are used to train the deep neural network. In order to enable the network to recover from mistakes and not drifting off the road, augmented data with additional images that show the car in different shifts from the center of the lane are also used for training. Figure 2.2 shows the high level view of the end to end automated steering training. Images for two specific off-center shifts can be obtained from the left and the right camera. Additional shifts between the cameras and all rotations are simulated by viewpoint transformation of the image from the nearest camera. Then, images are fed into a CNN which then computes a proposed steering command. The proposed command is compared to the desired command for that image and the weights of the CNN are adjusted to bring the CNN output closer to the desired output. The weight adjustment is accomplished using back propagation.

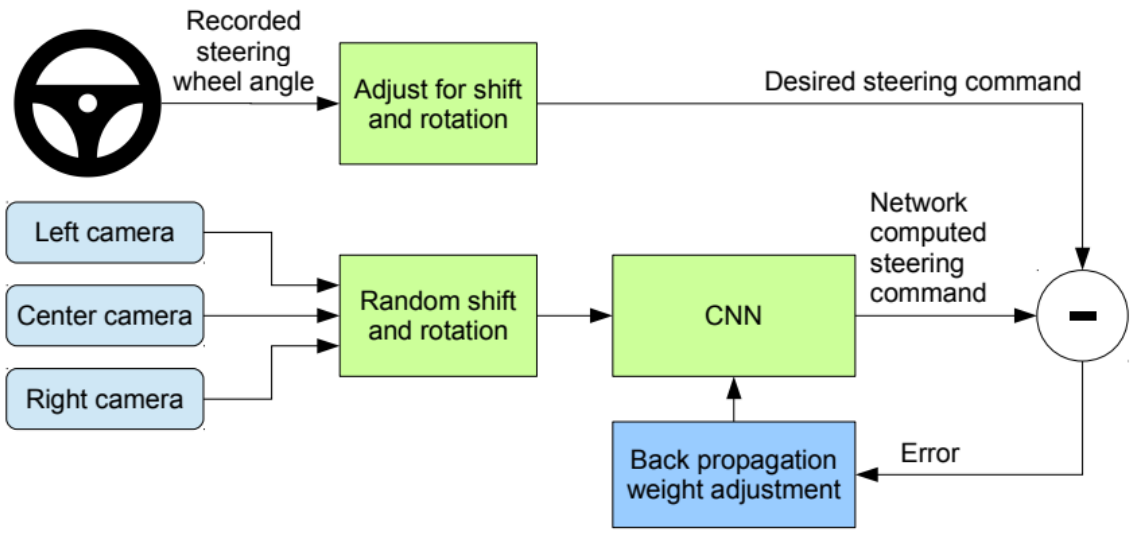


Figure 2.2: High level view of end to end automated steering training. [2]

The system automatically learns internal representations of the necessary processing steps such as detecting useful road features with only the human steering angle as the training signal. Compared to explicit decomposition of the problem, such as lane marking detection, path planning, and control, the end-to-end system optimizes all processing steps simultaneously. The system arguably leads to better performance and smaller systems for autonomous driving tasks. Better performance will result because the internal components self-optimize to maximize overall system performance, instead of optimizing human-selected intermediate criteria, e.g., lane detection.

2.3 V2V Communication

V2V communication technology enables the sharing of sensing information between vehicles, which can be leveraged for collaborative perception to improve the vehicle autonomy under complex traffic situations. With the shared sensing information from neighboring vehicles, a vehicle can extend its perception range to reveal hidden objects which were visually blocked or beyond the sensing range [4]. With different information availability, vehicles are able to share abstract information and raw sensor information by utilizing different V2V communication technology.

Dedicated short-range communications (DSRC) is a state-of-the-art wireless technology

that enables V2V, vehicle-to-infrastructure (V2I), or even vehicle-to-everything (V2X) communication systems [5]. DSRC's primary use cases are driving safety-related applications. In these applications, each vehicle periodically exchanges its status information (e.g., driving speed, GPS location, heading, etc.) with other road users (e.g., pedestrians, bicyclists, even smart traffic lights). With shared information, road users can achieve better situational awareness and thus improve driving safety [6–8]. However, although DSRC enables basic message exchange among road users with a range of up to 1000m (in near-ideal conditions), the maximum data rates of DSRC are normally as low as 2-6 Mbps.

Although DSRC enables basic message exchange between road users with a range of up to 1000 m (in near-ideal conditions), its data rates are normally as low as 2 - 6 Mbps, which are not sufficient for sharing large volumes of sensor data. Thanks to the large bandwidth channels in mmWave bands, mmWave is a promising candidate to realize the high data rates [9]. MmWave refers to the spectrum between 30 to 300 GHz. In these frequencies, channels can exploit a larger bandwidth. By combining the large bandwidth with advanced modulation techniques, mmWave significantly increases the data rates. As stated in the IEEE 802.11ad standard [10], mmWave can achieve up to 7 Gbps data rates with 2.16 GHz of bandwidth in the unlicensed 60 GHz band. Compared with DSRC, mmWave can be used to share the raw sensor data (e.g., a LIDAR sensor can generate $10 - 100$ Mb data per second and a camera generates $100 - 700$ Mb raw data per second). Therefore, V2V communication enables vehicles to share different volume of information depending on the application requirements.

Chapter 3

Big Road: Scaling Road Data Acquisition for Dependable Self-Driving

3.1 Introduction

Advanced driver assistance systems and, in particular, automated driving offer an unprecedented opportunity to transform the safety, efficiency, comfort, and economics of road travel. This has led many organizations in the computing and transportation domains to focusing on self-driving technology research. While this has resulted in a number of prototypes with impressive performance, it remains widely recognized that ensuring dependability under varied traffic conditions remains a key challenge [11–13].

Self-driving vehicles have to operate safely even under unusual or rare traffic events that are challenging to address and could lead to potential accidents. Developing such technology therefore requires an understanding of not just common highway and city traffic situations but also a plethora of widely different unusual events (e.g., objects on the roadway, pedestrian crossing highway, deer standing next to the road, etc.). While each such event may be rare, in aggregate they represent a significant risk that technology must address to develop truly dependable automated driving and traffic safety technologies. The average human driver achieves on the order of almost 100 million vehicle miles traveled per fatality [14]. Demonstrating driving performance at an above-average, advanced human driver level will therefore require successfully avoiding fatalities with unusual events that might be encountered within a billion miles of driving. This motivates the need for scaling road dataset to billions of miles of driving so that they contain a representative set of such unusual and rare events.

Most existing efforts to collect driving data build on a small fleet of tens of highly instrumented vehicles that are continuously operated with test drivers [15–17]. In terms

of miles recorded, it is challenging to accumulate a sufficiently large dataset with this approach. From the tidbits of published information, we know, for example, that Google's fleet has completed about 2 million testing miles [15]—impressive but still far off from a billion miles. On the other hand, since many existing efforts to develop automated driving technology are proprietary, the data obtained is usually closely guarded. It is not easy for individuals outside vehicle industry to collect driving inputs such as steering wheel angle and pedal operations. The OpenPilot [18] project reverse engineers the OBD-II data from two vehicle models of Hondas and Acuras, and uses the collected data to train the CNN based Adaptive Cruise Control (ACC) and Lane Keeping Assist System (LKAS). However, this solution can only be deployed in specific vehicle models, which makes it harder to recruit very large numbers of vehicles. Government-sponsored naturalistic driving studies have produced similar datasets, such as the UMTRI Naturalistic Driving Data [19] or the 100-vehicle VTTI Naturalistic Driving Study [20] that reached 2 million miles. These studies are more focused on human driving behaviors and still far below the target scope for supporting robust self-driving.

To address this challenge, this chapter asks whether data useful for self-driving can be gathered with only minimal instrumentation of vehicles. Such a minimal vehicle instrumentation approach would enable scaling by capturing events from tens of thousands of vehicles rather than only attempting to collect data with a few highly instrumented vehicles or certain vehicle models, as is common practice. A key challenge in creating such minimal instrumentation is the heterogeneity of vehicle designs and the proprietary nature of in-vehicle systems. To be useful for driver assistance and automated driving applications, the dataset must capture the surrounding traffic situation of the vehicle and how the vehicle was driven through this traffic scenario (i.e., it's precise trajectory and the necessary driver input). The latter is especially important for approaches relying on machine learning, which is increasingly used in such systems [2]. Here, the driver input data provides important positive and negative training examples that allow the system to learn how to react to traffic situations. It is also useful for system validation, since it allows automated comparisons of the response of automated driving algorithms with those of a human driver. Any significant deviations can then be more closely examined.

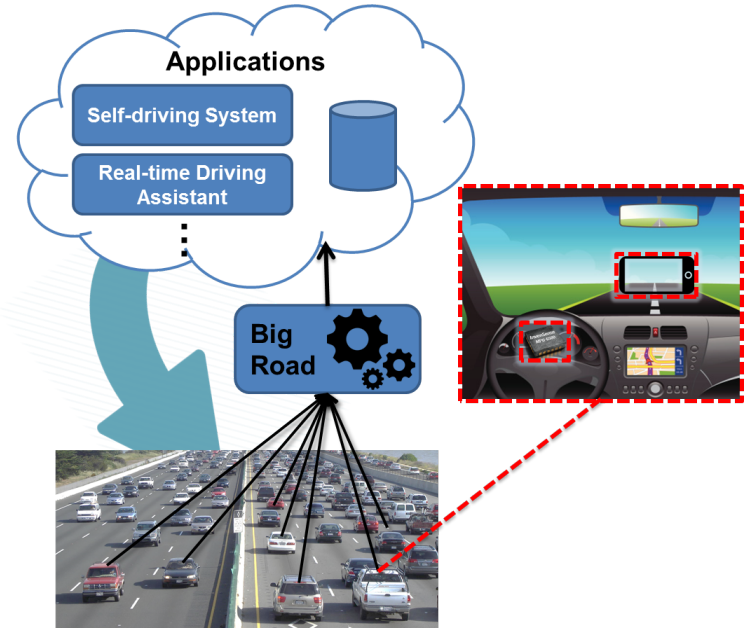


Figure 3.1: Illustration of BigRoad containing a smartphone and an IMU sensor: scaling road data acquisition.

The primary technical challenge is to obtain accurate vehicle movements without extensive instrumentation of the vehicle. While vehicles contain internal sensors to track steering and pedal inputs, the specific sensors vary among models and car makers use different proprietary data formats, if the information is exposed through the OBD-II port at all [21]. This heterogeneity renders scaling to many different vehicles difficult. Global Positioning System tracking of vehicles does not always capture fine-grained steering and speed changes, particularly in urban canyons. Gathering information about the surrounding traffic situation is more straightforward—in most situations, a front-facing camera can provide rich information. It is worthwhile complementing this information, however, in darkness and other situations where visual information may be insufficient.

BigRoad, as shown in Figure 3.1, aims to minimize instrumentation of vehicles by relying on low-cost inertial sensors that can be affixed to the vehicle. An inertial sensor in a dash-cam or windshield mounted smartphone enhances speed estimation, particularly in low-speed scenarios, therefore providing an indirect measurement of acceleration and braking inputs. A second steering wheel sensor gathers angle information, which allows a much more precise measurement of vehicle turning. To achieve this BigRoad incorporates algorithms

that isolate steering and vehicle movements from other forces acting on the vehicle. It also compensates for unknown orientations of the devices. Note that, even though smartwatch or other wrist-wear devices can also be used to infer driver inputs with less instrumentation, the inconsistency mapping between the device position and driver's steering wheel angle will bring additional error. Our initial work shows such method will has much lower accuracy compared with our proposed setup.

To understand whether such data can be useful for self-driving research, we study the performance of a key self-driving component, a self-steering algorithm. When trained with our data, this deep neural network-based algorithm takes road video as input and outputs the desired steering angle.

The contributions of this work can be summarized as follows:

- To our knowledge, this study is the first to analyze whether driving data useful for self-driving research can be collect with a minimal set of inertial and video add-on devices, in contrast to the existing work with significant instrumentation for scaling data acquisition.

- Developing steering wheel angle estimation algorithms by leveraging low-cost sensing devices (i.e., inertial measurement units (IMUs) and smartphones). The key novelty of this approach is that it extracts steering wheel rotation by eliminating vehicle movement from the steering-wheel-mounted sensor measurements.

- Designing robust vehicle speed sensing algorithms that combine GPS with speed delay shifting, and an acceleration based complementary filter.

- Devising an acceleration-based road condition estimator to enhance road condition awareness under poor lighting conditions and to illustrate further uses of the sensed data.

- Collecting and analyzing 40 hours of driving data to determine the accuracy of the estimation techniques and to demonstrate that the fine-grained driving data provided by the proposed framework achieves comparable performance in an automated vehicle steering algorithm.

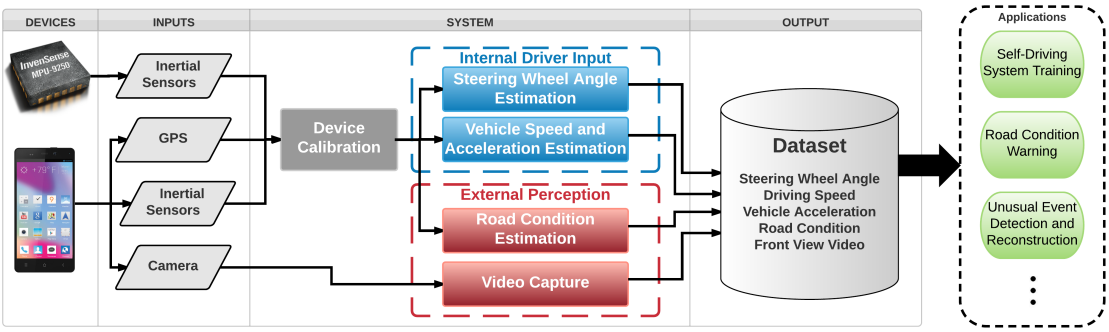


Figure 3.2: BigRoad system overview.

3.2 System Design

The main goal of BigRoad is to provide a light-weight automated data logging system dedicated to crowdsourcing fine-grained driving data, which can facilitate extensive research in self-driving vehicles and driving safety monitoring. By installing very few off-the-shelf sensing devices (i.e., a smartphone and an IMU) in a vehicle, the system can track vehicles' dynamics, drivers' driving behaviors, and road environments in real time. In addition to logging raw sensing data (i.e., GPS locations, motion sensor readings, and vehicles' front-view video), BigRoad devises novel approaches to derive various fine-grained driving data (i.e., steering wheel angles, vehicle speeds, and vehicle accelerations) by fusing the measurements from various sensors.

The major advantage of BigRoad is that it provides a minimum-effort solution for self-driving companies and researchers, who want to collect large datasets of ready-to-use driving data in real world without concerns of different vehicle types and driving behaviors. Traditional data logging systems only provide coarse-grained sensing data without recording the ground truth. Our system leverages various sensing technology in smartphones and IMUs to provide fine-grained measurements and experimental ground truth in the context of real driving. The information generated by BigRoad is two-fold: 1) The estimated steering wheel angles (directly from drivers), vehicle speeds and accelerations (indirectly from drivers) are considered to be *Internal Driver Input* to reconstruct vehicles' motions and driving behaviors in a fine-grained manner. 2) The estimated road conditions and videos captured by the cameras of smartphones are considered to be *External Perception* that can

capture driving environments and provide experimental ground truth.

BigRoad is realized with five main sub-tasks: *Device Calibration*, *Steering Wheel Angle Reconstruction*, *Vehicle Speed and Acceleration Estimation*, *Road Condition Estimation*, and *Time-stamped Video Capture*. Figure 3.2 shows the system overview of BigRoad. To participate, the user needs to install a smartphone on the dashboard and an IMU on the center of the steering wheel as illustrated in Figure 3.1. The system takes measurements of inertial sensors (i.e., acceleration and rotation rate in both the IMU and smartphone) as inputs to *Device Calibration*, which derives the rotation matrices that can calibrate the sensing measurements from both devices to the vehicle's coordinate system, disregarding vehicle models, steering wheel positions and IMU placements. The components of Internal Driver Input including *Steering Wheel Angle Estimation* and *Driving Speed and Acceleration Estimation* fuse GPS location and calibrated sensing measurements from inertial sensors to derive steering wheel angles, vehicle speeds, and accelerations. The components of External Perception including *Road Condition Estimation* and *Time-stamped Video Capture* utilize video camera and inertial sensors to capture critical information in the driving environment (i.e., traffics, road conditions, etc.). Road conditions in inclement weather can severely affect traffic demands, roadway capacity and increase risks of having traffic crashes [22].

The insight of our steering wheel angle estimation is that after applying the *Device Calibration*, which is discussed in Section 3.5, the rotation angles of the IMU aligned to the steering wheel plane could be used to estimate steering wheel angles. However, accurately estimating the steering wheel angles involves several challenges. First, steering motions from human usually happen in a sudden and change all the time, which makes them extremely hard to be captured by the IMU's inertial sensors. Second, the measurements from the inertial sensors of the IMU are contaminated by motions of the vehicle, such as turning and braking. Third, although the IMU's position on the steering wheel is stable, it is unknown and could be different from time to time. To harness these challenges, We obtain the rotation of the steering wheel based on IMU's sensor readings and uses the smartphone's acceleration to remove the error caused by vehicle motion during driving.

Besides, BigRoad fuses GPS and acceleration from smartphone to predict accurate vehicle driving speed. Since road conditions in inclement weather can severely affect traffic

demands, roadway capacity and increase risks of having traffic crashes [22], we propose to use real-time normalized traction force derived from inertial sensor readings during brakes activities to identify binary road conditions (i.e., dry or wet). Moreover, the *Time-stamped Video Capture* utilizes cameras of smartphones to provide the front-view video of the vehicle together with millisecond-granularity time stamps, which serves as the ready-to-use training datasets for various applications such as self-driving system training, road condition warning, and dangerous event detection and recommendation.

3.3 Internal Driver Input

3.3.1 Steering Wheel Angle Estimation

As one of the most important driver inputs, steering wheel operations play a critical role in self-driving systems. While we expect a much comprehensive system trained with a huge amount data in the future, collecting such a big dataset of steering wheel operations is urgent yet not easy. Different from other driver inputs, such as vehicle speed that is commonly available via a standard OBD-II interface, steering wheel operations (i.e., rotation angles) are usually hidden and only accessible to vehicle manufacturers. In order to obtain drivers' steering inputs regardless of vehicle models, we propose two sensor-based approaches to estimate steering wheel angles: 1) steering-wheel IMU based estimation and 2) phone based estimation. The difference between these two approaches is that the former requires an IMU attached to the steering wheel, while the latter only needs one smartphone at any in-vehicle position.

In particular, the steering-wheel IMU based estimation includes three steps: 1) the system fuses steering wheel angles estimated based on different inertial sensors in the IMU by using a complementary filter; 2) it removes the angle errors caused by vehicle motions from the sensor measurements; 3) the system calculates the angle biases from the coordinate alignment, and further calibrates the estimated steering wheel angle by removing the biases. Next, we first introduce the three steps for the steering wheel associated estimation, then we discuss the phone based estimation.

IMU Sensor Fusion

After coordinate alignment in the Device Calibration, measurements from the inertial sensors in the IMU are aligned to the steering wheel's two-dimensional plane. Intuitively, we could estimate the steering wheel angle by accumulating the angular velocities from the IMU's gyroscope. While the gyroscope measure rotation changes precisely, the integration process suffers from accumulative errors and results in large drifts over time. We also find that it is possible to estimate steering wheel angles based on the angular changes derived from the gravity projected onto the different axis from the IMU's accelerometer, but the accelerometer has lots of vibration noises, which cause large dynamic errors. [1]

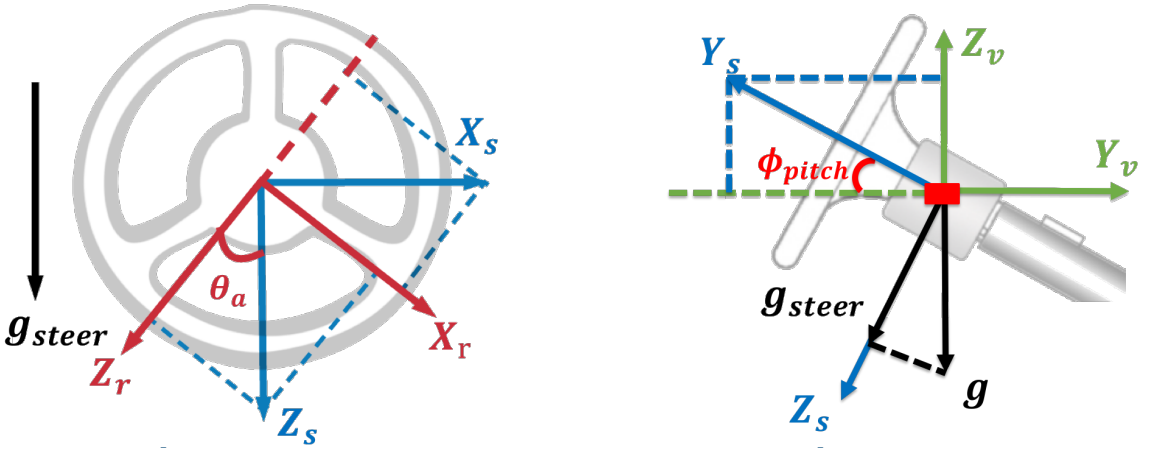
To address these limitations in both approaches, we devise a complementary filter [23] to fuse the steering wheel angles estimated based on the measurements from the IMU's accelerometer and gyroscope. The design of the complementary filter is shown as the following equation by applying both high-pass and low-pass filters:

$$\theta^{t|t-1} = cc * (\theta^{t-1} + \Delta\theta_{gyro}^{t|t-1}) + (1 - cc) * \theta_a, \tag{1}$$

where $\theta^{t|t-1}$ is the estimated steering wheel angle at time t based on the angle estimated at time $t-1$ denoted as θ^{t-1}, $\Delta\theta_{gyro}^{t|t-1}$ is the difference of estimated steering wheel angle derived by accumulating the IMU's gyroscope measurements around the axis of rotation over time t and $t-1$, θ_a is the steering wheel angle derived from the accelerometer's measurements, and cc is the variable that determines the time scale of the high and low-pass filters, which is set to 0.9 empirically.

Steering Wheel Angle Estimation based on Accelerations. We first consider the scenario when the vehicle is static, e.g., the vehicle is parked, in which gravity is the only force applied to the steering wheel. We examine the IMU's accelerations on the two dimensional steering wheel plane as illustrated in Figure 3.3(a). In the plane, we define two sub-coordinate systems: 1) X_s and Z_s denote the x and z-axis of the *steering wheel coordinate system*, and 2) X_r and Z_r denote the same in *rotated steering wheel coordinate system*. We assume Z_s points to the the same direction with the gravity projection(g_{steer})

[1] We have also considered exploiting magnetometer and rotation vector based approaches, but they are significantly affected by the surrounding magnetic field, which is usually unstable in urban environments.



(a) Steering wheel and rotated steering wheel coordinate system.

(b) Steering wheel and vehicle coordinate system.

Figure 3.3: Several coordinate systems and angle estimation.

on this plane and y-axis of both sub-coordinate are perpendicular to this surface as shown in Figure 3.3(b). Note that the *steering wheel coordinate system* is fixed but the *rotated steering wheel coordinate system* changes with the rotation of the steering wheel and IMU. Therefore, the problem becomes how to determine the angle between two coordinate systems.

Intuitively, when there is only gravity acceleration in the static scenario, the steering wheel angle θ_a can be estimated by first calculating the arc-tangent over the projection of the gravity acceleration on X_r and Z_r as shown in $\theta_{static} = atan2(a_{X_r}, a_{Z_r})$,

where a_{X_r} and a_{Z_r} can be derived from the IMU's acceleration readings aligned from its own coordinate system to the *rotated steering wheel coordinate system*, which is discussed in in the Device Calibration section.

$atan2(x, y)$ is the arc-tangent function that can calculate the angle in all four quadrants. The output of $atan2(x, y)$ ranges from $180 to 180$. Because the steering wheel angle may overflow this range, we find all possible angles by adding multiples of 360o the θ_{static}, and use the one closest to the last estimation as the estimated steering wheel angle θ_a.

Vehicle Motion Removal

The above steering wheel angle estimation is obtained when the vehicle is static. To have a better understanding of how the motions of vehicles affect θ_a, we revisit Figure 3.3(a)

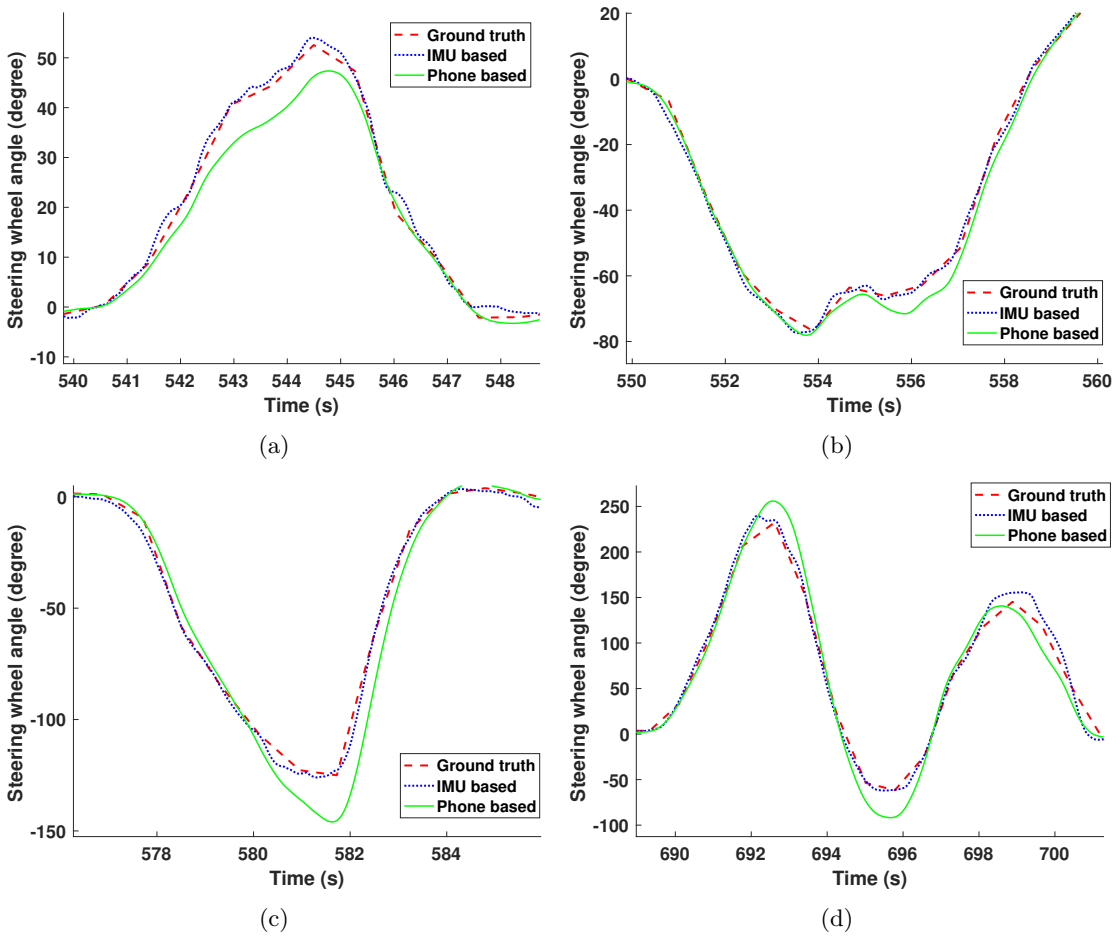


Figure 3.4: Compare the steering wheel angle estimation results of steering-wheel based and phone based approach.

for driving scenarios. When the vehicle is moving, the inertial sensors in the IMU and smartphone capture other accelerations (e.g., accelerations caused by turning, accelerating, and braking) in addition to the gravity force. Thus the accelerations projected to the X_r and Z_r axes can be derived as:

$$\begin{aligned} a_{X_r} &= sin(\theta_a) * a_{Z_s} + cos(\theta_a) * a_{X_s}, \\ a_{Z_r} &= cos(\theta_a) * a_{Z_s} - sin(\theta_a) * a_{X_s}, \end{aligned} \tag{2}$$

where a_{X_s} and a_{Z_s} are the accelerations aligned to the x- and z-axes of the *steering wheel coordinate system*. By combining two equations, we further derive the steering wheel angle in moving scenarios as below:

$$
\begin{aligned}
tan(\theta_a) &= \frac{tan(\theta_{static}) - a_{X_s}/a_{Z_s}}{1 + a_{X_s}/a_{Z_s} * tan(\theta_{static})}, \\
&= tan(\theta_{static} + atan(a_{X_s}/a_{Z_s})), \\
\Rightarrow \theta_a &= \theta_{static} - atan(a_{X_s}/a_{Z_s}), \\
&= \theta_{static} - \theta_{error}.
\end{aligned} \tag{3}
$$

We note that the estimated steering wheel angle in Equation 3 can be considered as the angle estimated in the static scenario (i.e., θ_{static}) calibrated by removing an angular error (i.e., θ_{error}), which is determined by the accelerations in the *steering wheel coordinate system* and independent of the poses of the steering wheel and IMU.

In order to obtain a_{X_s} and a_{Z_s} in Equation 3, we examine the relationship between the *steering wheel coordinate system* and the *vehicle coordinate system* as illustrated in Figure 3.3(b). We find that a_{X_s} is the same as the vehicle's acceleration on the x-axis of its own coordinate system (a_{X_v}), which can be easily obtained by aligning the measurements of the smartphone's accelerometer to the *vehicle coordinate system* in the Device Calibration. In addition, we find that the steering wheel always has a pitch angle (ϕ_{pitch}) to the y-axis of the vehicle coordinate system. Therefore, a_{Z_s} is the combination of the gravity acceleration and the vehicle's acceleration projecting to the z-axis of the *steering wheel coordinate system* as shown below:

$$
a_{Z_s} = g * sin(\phi_{pitch}) - a_{Y_v} * sin(\phi_{pitch}). \tag{4}
$$

Finally, the complementary filter is still used to estimate the steering wheel angle by fusing the estimated steering wheel angel based on accelerations and angular velocity derived from IMU's y-axis gyroscope reading after aligned to *rotated steering wheel coordinate system.*

Angle Bias Removal

The constant rotation of steering wheel makes error existing in the coordinate alignment during Device Calibration, resulting in a bias in the estimated steering wheel angle. To remove this angle bias, our system calibrates itself by examining the average estimated steering wheel angle when the vehicle is driving straight. The intuition is that the ideally

estimated steering wheel angle should be zero degrees when driving straight, therefore any non-zero average estimated steering wheel angle found when driving straight is the angle bias that we should remove.

In particular, our system keeps collecting the angular velocity on the z-axis of the *vehicle coordinate system* from the gyroscope of the smartphone ($gyro_{Z_v}$) and that on the y-axis of the *steering wheel coordinate system* from the gyroscope of the IMU ($gyro_{Y_s}$) since the start of each trip. If the angular velocities from both sources are less than a threshold (e.g., $0.01 rad/s$ and $0.1 rad/s$ for $gyro_{Z_v}$ and $gyro_{Y_s}$, respectively), the system considers that the vehicle is driving straight and pushes the estimated steering wheel angle into a sample pool. The average of the samples in the pool is considered as the angle bias to be removed from all the steering wheel angles estimated in the rest of the trip. The size of the sample pool depends on the time needed to collect enough samples that can confidently determine the angle bias in every trip. In our experiments, we find that our system can successfully find and remove this angle bias within the first one minute of typical daily commute trips.

Phone Based Steering Wheel Angle Estimation

An alternative approach in BigRoad is not to fix the sensor on the steering wheel. Instead, we can just employ the available smartphone inside the vehicle to accomplish the steering wheel angle estimation task. This method can be adopted to any inertial sensors, but we use the smartphone sensor here to minimize the infrastructure needs. Instead of explaining bunches of complex mathematical equations, we simplify the model as shown in Figure 3.5, in which we assume the turning angle of the two front tires are the same. For common vehicles, the steering wheel angle($\theta_{steering}$) is equal to the front tire's turning angle(θ_{tire}) multiply the vehicle's steering ratio(k). And if the turning radius(r) and wheelbase(l) of the testing vehicle are known, θ_{tire} can be calculated in real time:

$$\begin{aligned} \theta_{steering} &= k * \theta_{tire} = k * asin\frac{l}{r}, \\ r &= \frac{v}{\omega} = \frac{v}{gyro_{Z_v}}. \end{aligned} \tag{5}$$

The vehicle turning radius(r) can be obtained as shown in Equation 5 based on the physics phenomenon inherited from inertial sensor readings. Here v stands for the vehicle

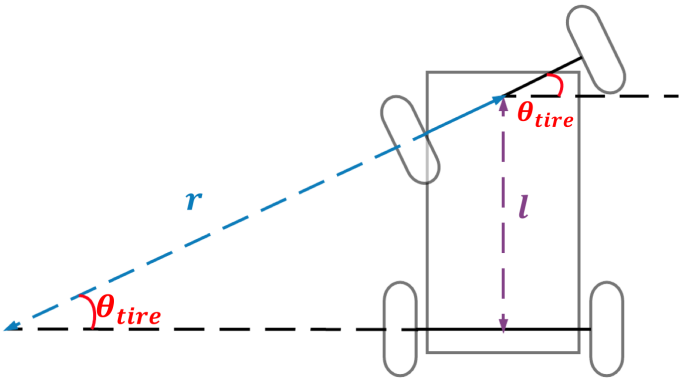


Figure 3.5: Simplified Ackerman mechanism based steering angle calculation.

speed, which we can get from the vehicle speed estimation module presented in section 3.3.2. ω represents the angular velocity of the vehicle, which can be replaced by the gyroscope reading on the z-axis of the *vehicle's coordinate system*($gyroZ_v$). Thus, we can calculate the vehicle turning radius, and then get θ_{tire} leveraging only a smartphone. However, the steering ratios(k) of most commercial vehicle are proprietary. To obtain the steering ratio of our testing vehicle, we apply linear regression on the steering wheel angle ground truth and estimated tire angle from part of our collected dataset, and use the ratio to evaluate the remaining part of the dataset.

To compare these two steering wheel angle estimation approaches, the phone based approach utilize less infrastructures, but it needs more vehicle information such as steering ratio and wheelbase. The steering-wheel IMU based approach can estimate steering wheel angle of any vehicle model without extra information by exploiting the additional IMU sensor on the steering wheel. Figure 3.4 demonstrates the estimation result of two approaches and compare them with ground truth. We will show that the performance of steering-wheel IMU sensor based approach is better than the phone based approach in section 5.7.

3.3.2 Driving Acceleration and Speed Estimation

In addition to steering wheel angles, driving accelerations and speed are also critical inputs for self-driving research and thus part of the system's outputs. The driving acceleration is the smartphone's acceleration projected to the driving direction when the smartphone is fixed in the vehicle for video recording. Therefore the acceleration can be obtained after

the smartphone's measurements are aligned to the vehicle's coordinate system. An intuitive way to obtain the driving speed is exploiting the location service in smartphones' operating systems. However, such speed information has average errors of over 1 km/h, not accurate enough for speed-sensitive applications, such as self-driving training. To improve the accuracy, we propose to shift the speed measurements from the system with an average delay and apply an acceleration-based complementary filter to compensate large speed changes.

Speed Delay Shifting

The speed information provided by smartphones' operating systems are derived by fusing the GPS, WiFi and cellular network information. Because GPS signals have large round trip time (RTT), such speed information always has some delays comparing to the ground truth (i.e., speed measurements from the OBD interface), although their trends are very similar. Figure 3.14 illustrates the time delay between OBD readings and smartphone readings and their similarity in a single trip. Based on this observation, we determine the time delay τ by minimizing the average absolute errors between the speed measurements from the OBD and smartphone's GPS through a segment of data of m samples as shown in the following equation:

$$\underset{\tau}{\operatorname{argmin}} \sum_{i=1}^{m} \frac{1}{m} |V_i^{obd} - V_{i+\tau}^{gps}|, \tag{6}$$

where V_i^{obd} and $V_{i+\tau}^{gps}$ are i^{th} speed measurement from OBD and GPS, respectively. The time delay τ will be applied to shift all speed measurements from the smartphone's GPS. Note that the delays are dominated by the RTT of GPS and they are reasonably similar among different trips, therefore the system just needs to calculate the τ once in a while.

Acceleration based Complementary Filter

The speed delay shifting does be able to largely reduce the error due to asynchronous sampling, but the speed from smartphones are still not accurate when changing frequently, because the update rate of GPS is as low as one update per second. To capture the frequent changes in speed, we propose to fuse the speed derived by the vehicle's acceleration and the speed from the smartphone's GPS. In particular, we estimate the driving speed by using a

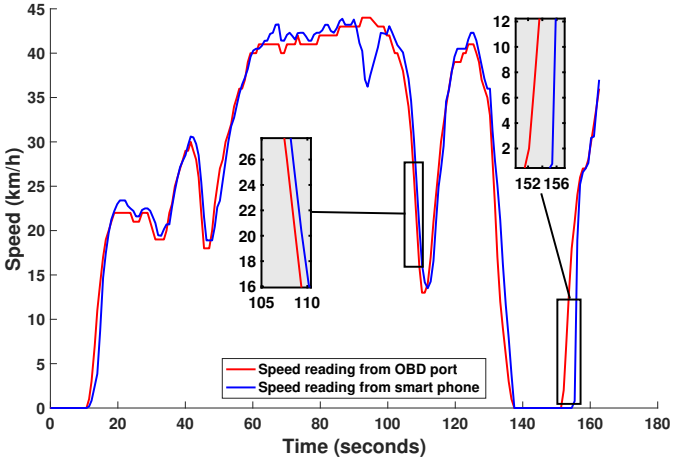


Figure 3.6: Speed readings from OBD-II and smartphone.

complementary filter as shown in the following equation:

$$V_t^E = \alpha * (V_{t-1}^E + \Delta V_t) + (1 - \alpha) * V_t^P, \tag{7}$$

where V_t^E and V_{t-1}^E denote the speed of the vehicle at time t and $t-1$, ΔV_t is the speed change between t to $t-1$ derived from the accumulated accelerations of the vehicle, V_t^P is the speed from the smartphone's GPS, and α is a variable controlling the balance between two speed sources. The acceleration of the vehicle is available from the smartphone's accelerometer measurements after the coordinate alignment in Device Calibration.Although advanced GPS chipsets [24] have already integrated accelerometers for tracking speed, our method is still able to improve the accuracy of speed logged by commodity smartphones, which are only equipped with standalone GPS module.

3.4 External Perception Recording

In addition to driver inputs, the external environment should be acquired in order to provide a perception for self-driving vehicle studies. We provide a synchronization mechanism in the next subsection since many studies utilize camera input to have a grasp of the external environment. Beyond the visual perception, the road conditions can be also useful. In section 5.2, we introduce methods to estimate road conditions, namely wet and dry.

3.4.1 Time-stamped Video Capture

The most intuitive external perception provided by BigRoad is the real-time front-view video feed from Smartphones' rear cameras that capture everything happening on the road in front of the vehicle. The desire of such directly visual information is tremendous, as it has been exploited in many vehicle applications, including self-driving, traffic crowdsourcing, etc. Most of these applications need to synchronize the video frames to sensor data at millisecond granularity for training or evaluation. However, the video recorded by most mobile systems in the market only provides the time information at second-level granularity, which cannot support such fine-grained synchronization. To address the low-granularity issue, BigRoad devises a video capture module that records video frames with a list of time stamps for each frame using the smartphone's $system_current_time$ at millisecond granularity. The fine-grained time information provided by BigRoad can facilitate many vehicle application. For instance, the steering wheel angle output can be interpolated to match the time stamp of each frame, and used to train autonomous steering system.

3.4.2 Road Condition Estimation

Next, we introduce how BigRoad provides the estimation of road conditions (i.e., dry or wet) based on sensors integrated in smartphones.

Intuition

In advanced driving assistance and automated driving systems, the awareness of road conditions is one of the critical parameters for adjusting speed and turning angles to ensure the safety of passengers. Most existing studies identify factors that affect the road's friction coefficient by using vehicle-mounted specialized sensors (e.g., optical fibers [25], stereo camera [26], 24-GHz automotive radar [27] and voice recorder [28,29]), which require either extra cost and installation effort or visibility/lightening environments. Our approach is thus challenging as we do not use any additional sensors (e.g., camera, automotive radar) to directly sense the road's condition. In contrast, we analyze the statistics of braking events captured by phones' inertial sensors through crowdsourcing and estimate road conditions,

which is low-cost and easy to deploy.

In particular, we develop a proof-of-concept estimation framework to determine binary road conditions (i.e., dry or wet) by examining inertial sensors measurements in vehicles' brake activities from BigRoad. The intuition is two-fold: First, different road conditions have different maximum friction coefficients, which are usually defined as the following equation [30]:

$$\mu_{max} = max(|\frac{F_x}{F_z}|), \tag{8}$$

where F_x is the longitudinal traction force and F_z is the normal force acting on the tire. By defining the normalized traction force (NTF) of the vehicle as $\rho = \frac{F_x}{F_z}$, the μ_{max} can be represented by the $max(\rho)$, which can be obtained by a smartphone's inertial sensors when the vehicle is experiencing a hard brake. Because dry roads always have much larger μ_{max} than wet roads have, the $max(\rho)$ of a vehicle in dry road conditions should be larger than those in wet road conditions.

Second, in order to avoid skids on wet roads, most drivers would brake earlier and more gently [31] than they do on dry roads, resulting in relatively smaller ρ in normal braking activities. Since drivers usually perform multiple braking activities in each trip due to the stop signs, traffic lights, traffics, and etc, it is possible to crowdsource NTFs in braking activities from a large number of vehicles and drivers to accurately estimate road conditions. We note that video camera in BigRoad is another sensor that we can use to estimate road conditions by using image processing techniques, but it highly depends on the visibility and lightening environments. Therefore, we focus on using the inertial sensors based approach in this chapter.

Per-brake NTF Derivation

In order to derive the NTF ρ during each brake, we first detect and segment the braking events using the acceleration readings collected by the smartphone in BigRoad. Specifically, we examine the standard deviation of accelerations on three axes to determine the vehicle's stop period using a threshold-based approach. We then extract six-second measurements before the starting point of the stop period as the segment data in a braking event.

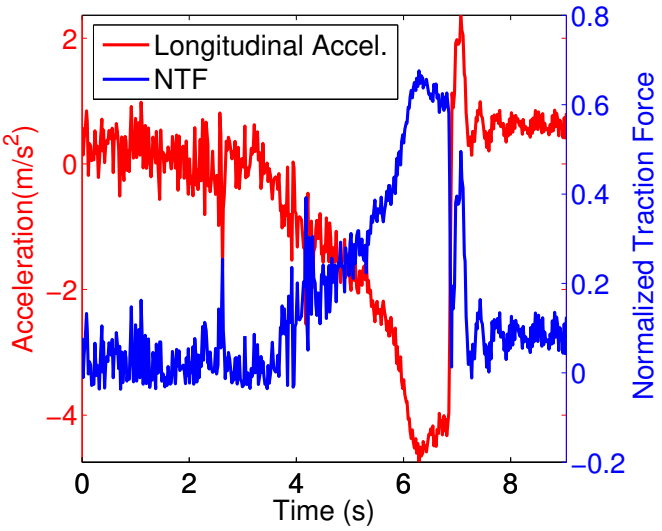


Figure 3.7: Example of longitudinal acceleration and normalized traction force during a brake.

Next, we calculate the NTF in each brake event using the longitudinal acceleration, which can be obtained by the Device Calibration (Section 3.5). To simply the problem, we consider a bicycle-type vehicle model [30], where the difference between left and right tires is ignored. If we ignore the effects caused by winds and road gradient, the longitudinal force during a brake can be obtained by:

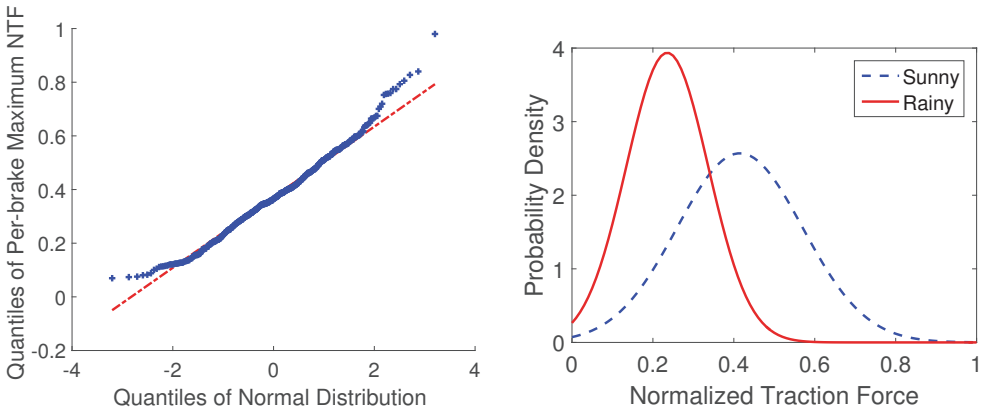
$$F_x = m|a_x| - |F_r|, \tag{9}$$

where m is the total mass of the vehicle, a_x is the longitudinal acceleration and F_r is rolling resistance force which is usually between $0.015mg$ and $0.02mg$.

In addition, the normal forces on the front and rear tires can be calculated as:

$$F_{zf} = \frac{mgL_r - ma_x h}{L}; F_{zr} = \frac{mgL_f + ma_x h}{L}, \tag{10}$$

where L_f and L_r are the distances from the center of gravity to the front and real axles respectively. h is the distance from the center of gravity to the road surface and L is the wheel base (i.e., $L = L_f + L_r$). Note that we can use either F_{zf} or F_{rf} to calculate the vehicle's NTF ρ depending on the form of the vehicle's drivetrain (i.e., front-wheel drive or rear-wheel drive). Finally, the NTF can be derived using the vehicle's longitudinal acceleration during each brake and a few of the vehicle's basic information. Figure 3.7 shows an example of the longitudinal acceleration and its corresponding NTF during a brake.



(a) QQ-plot (vs. normal distribution). (b) Probability density functions.

Figure 3.8: QQ-plot and probability density functions of maximum per-brake NTFs.

Crowd-sourcing-based Road Condition Estimation

The instant NTF derived from each brake can be exploited to estimate road-conditions by being compared to different road-condition models. Such models are nothing but the distributions of NTFs abstracted from a large number of NTFs crowd-sourced from vehicles driving in different road conditions. The crowdsourced data used for building general models should cover different braking types, driving behaviors, etc., so that the models can be resilient to these different situations. Additionally, we empirically observe that the maximum NTF in each brake (e.g., the peak value of the NTF in Figure 3.7) fits normal distributions. The Quantile-Quantile plot in Figure 3.8(a) compares over 700 maximum NTFs from 40 daily trips of 5 drivers with normal distribution and verifies this observation. We thus use the least-squares based approach to fit the maximum NTFs collected from brakes in sunny days (i.e., dry condition) and rainy days (i.e., wet condition) to two road-condition models, which are two probability density function following normal distributions, namely pdf_d and pdf_w. Figure 3.8(b) shows an example of pdf_d and pdf_w generated by a training dataset (i.e., 350 brakes in sunny days and 100 brakes in rainy days) collected by BigRoad from 5 users at different locations. The figure also verifies that drivers brake more gently in rainy days as we expected.

It is important to note that determining the road condition based on the NTF information of only a single braking action is difficult, because the two distributions shown in

Figure 3.8(b) are largely overlapping with each other. BigRoad takes the advantage of crowdsourcing technique and collects a large number of braking events with different braking types and driving behaviors from various on-road vehicles in a specific driving area. The system eventually can respectively calculate the joint probability of all these brakes performed on dry or wet roads, and determine the road conditions accordingly.

Specifically, we can estimate the road condition by comparing the instant maximum NTF with the abstracted road condition models as shown in the following equations:

$$\begin{cases} \text{Road is dry, if} \prod_{i=1}^{N} pdf_d(\hat{\rho}_i) \geqslant \prod_{i=1}^{N} pdf_w(\hat{\rho}_i) \\ \text{Road is wet, if} \prod_{i=1}^{N} pdf_d(\hat{\rho}_i) < \prod_{i=1}^{N} pdf_w(\hat{\rho}_i), \end{cases} \tag{11}$$

where $\hat{\rho}_i$ is the maximum NTF of the i^{th} collected brake, N is the total number of brakes being used.

3.5 Device Calibration

In order to ensure BigRoad work with most vehicle models, the system should be able to calibrate itself to different steering wheel positions, sensor placements, and vehicle models. The main challenge is that both the smartphone and IMU have their own coordinate systems, and the sensor measurements from both devices need to be aligned to the same coordinate system (e.g., the vehicle coordinate system) before they are useful.

We develop two modules to automatically align measurements from smartphone and IMU to the *vehicle coordinate system*(X_p, Y_p, Z_p) and *rotated steering wheel coordinate system*(X_r, Y_r, Z_r) when driving, namely *Smartphone Calibration* and *IMU Calibration*. The relationship between the two coordinate systems is shown in Figure 3.9.

Smartphone Calibration. We implement the coordinate alignment algorithm to align sensor measurements from the smartphone to the *vehicle coordinate system* based on an existing work [32]. Specifically, it utilizes the smartphone's accelerometer and gyroscope measurements when the vehicle brakes while driving straight to derive the corresponding rotation matrix.

IMU Calibration. The coordinate alignment for IMU is challenging, because the

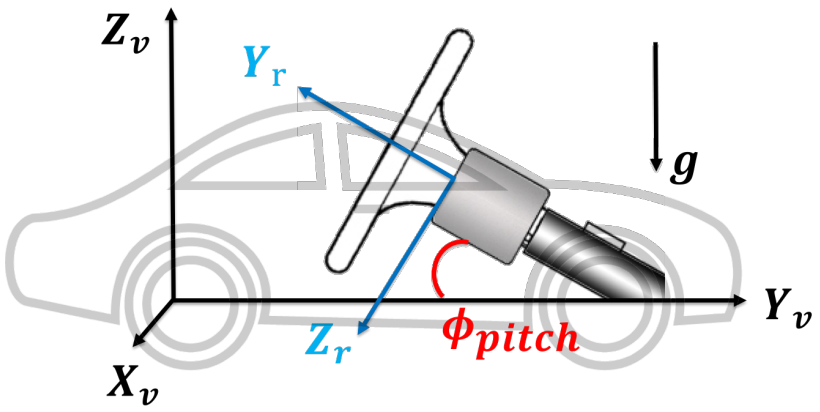


Figure 3.9: Vehicle coordinate system and rotated steering wheel coordinate system.

IMU's orientation is subject to the rotation of the steering wheel, IMU's position, and vehicle models. The IMU Configuration adopts two steps to align the sensor measurements from the IMU to the *steering wheel coordinate system*: First, the system aligns the sensor measurements to the *vehicle coordinate system* when vehicle is driving straight using the same approach introduced in the Smartphone Configuration. Since we perform this alignment while driving straight, aligned IMU's coordinate system will depart the *vehicle coordinate system* when steering wheel turns, but its x-axis is always identical to the *rotated steering wheel coordinate system*. Second, the system calculates the steering wheel pitch angle (ϕ_{pitch}) and further rotate other two axis to the *rotated steering wheel coordinate system*.

As shown in Figure 3.9, ϕ_{pitch} is the angle between Y_v's negative direction and Y_r's positive direction, which is usually adjustable by the driver. And this angle also exists between negative direction y-axis of the aligned IMU and Y_r's positive direction. Thus, ϕ_{pitch} can be derived from the angular velocity projection on aligned IMU's coordinate system's y- and z-axis, as shown in equation12:

$$\phi_{pitch} = atan(\frac{-gyro_{Y_i}}{gyro_{Z_i}}), \tag{12}$$

where $gyro_{Y_i}$ and $gyro_{Z_i}$ are the gyroscope reading's on IMU's y- and z- axes after aligned to *vehicle coordinate system*. The system automatically picks twenty samples from a steering motion and calculate ϕ_{pitch} based on Equation 12. Then, IMU measurements can be rotated to the *rotated steering wheel coordinate system* by rotating around x-axis for $(180 - \phi_{pitch})$ degree. Besides, BigRoad also includes a data synchronization module to remove the delay

of IMU data caused by Bluetooth transmission. This module uses the same brake segment used in smartphone coordinate alignment and runs cross correlation on smartphone and IMU acceleration trace to find the delay.

3.6 Performance Evaluation

In this section, we evaluate our system with respect to (i) the accuracy of internal driver input information it provides, (ii) the accuracy of external perception monitoring, and (iii) the usefulness of our collected data, namely whether our collected data could allow training of self-driving system components.

3.6.1 Experiment Setup

We conduct driving experiments with BigRoad to evaluate its performance. Our experiments include six vehicle models and 7 drivers driving on various types of roads for their daily commute in two states. In total, we collect 143 trips in a 3-month period. During these trips we used different experiment configurations to allow studying different questions as follows:

Internal Driver Inputs Accuracy. We collect 84 trips from three vehicles (i.e., 2015 Honda Civic, 2016 Chevrolet Impala, 2016 Chevrolet Equinox) and five drivers to evaluate the accuracy of the collected driver inputs. We used these vehicles because we were able to gain access to ground truth steering wheel angle data reported by an internal sensor (for two of the vehicles a carmaker provided us with a specialized device and for one vehicle we were able to reverse engineer the data format of messages captured with a standard OBD-II/CAN interface). We also recorded driving speed from the internal vehicle bus, which is openly available as part of the OBD-II standard. We were able to sample the data from the Chevrolet Impala and Equinox at 100Hz and 10Hz, respectively. While there is no commercial OBD reader can directly read steering wheel angles from the Honda Civic, we reverse engineer the bus data of the Honda Civic to record the steering wheel angle from the OBD interface with 100Hz. However, limited by the reverse engineering, we can only record the ground truth at 1.43Hz if monitoring speed and steering wheel angle simultaneously.

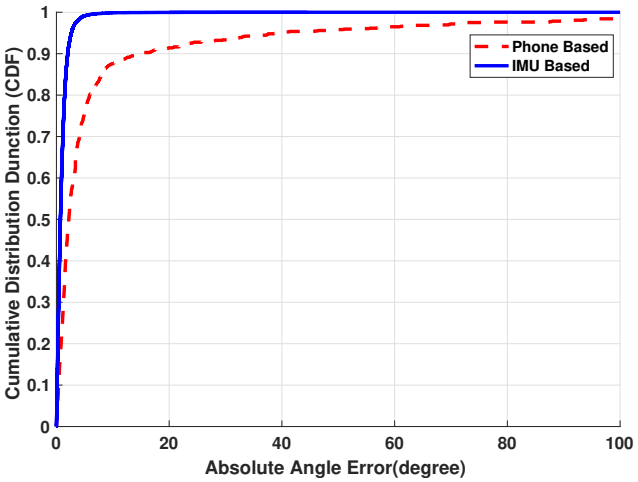


Figure 3.10: CDF of steering wheel angle absolute error.

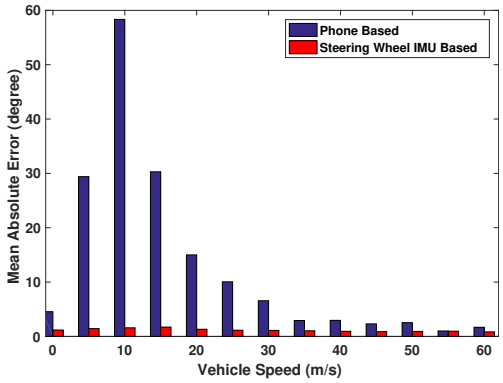
External Perception Accuracy. We evaluate the performance of road condition estimation based on 59 trips from four vehicles and five drivers. The ground truth of the road condition and weather for each trip is manually labeled by the drivers.

Data Usefulness. We utilize 5 trips from three different drivers driving the 2015 Honda Civic on highways to show the usability of BigRoad's video output. The dataset includes about 1.5hours video with 30fps and steering wheel angle ground truth from the OBD interface at 100Hz.

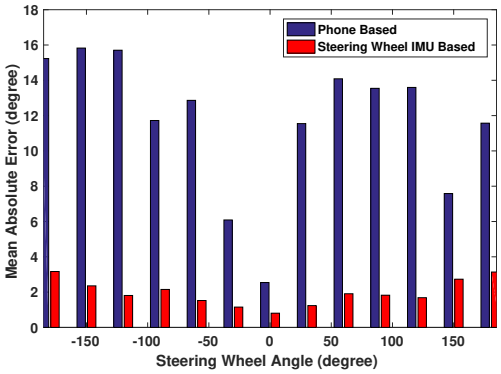
3.6.2 Accuracy of Driver Input

Steering Wheel Angle Estimation. We first evaluate the performance of our steering-wheel IMU based and phone based approaches. In particular, we calculate the absolute errors[2] of the estimated steering wheel angle from both approaches by respectively comparing them with the ground truth from the OBD readers. Figure 3.10 shows the CDFs of the steering wheel angle estimation errors of our two approaches. The rationale behind comparing the performance of the two methods is to quantify the accuracy-complexity trade-off (how does accuracy degrade when using only phone sensors?) and to understand whether the accuracy with phone sensors is still sufficient for self-driving data collection. We observe that the overall errors of the steering-wheel IMU based approach are much smaller than

[2]The errors used in the rest of the paper are all absolute error, unless stated otherwise.



(a) Error distribution with respect to vehicle speed.



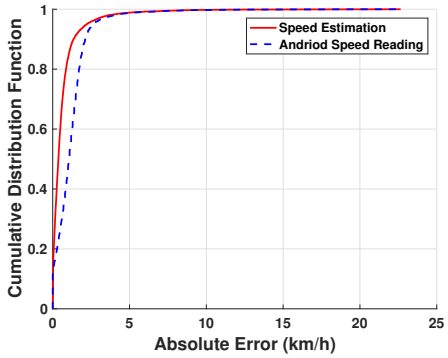
(b) Error distribution with respect to steering angle.

Figure 3.11: Error distribution with respect to speed and steering angle.

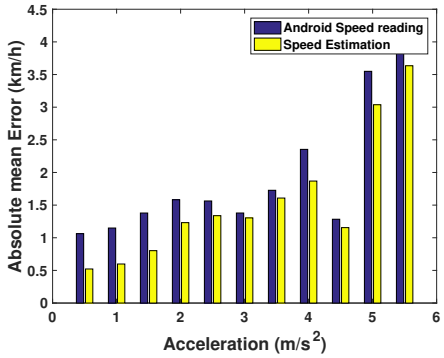
those of the phone based approach. In particular, the mean error of the steering-wheel IMU based approach is 0.96ith the median of 0.69nd 90-percentile of 1.99 while the mean error of the phone based approach is 7.53ith the median of 2.19nd the 90-percentile of 15.05 Given the 1.5ccuracy of the commercial steering angle sensor [33], our steering-wheel IMU based approach could archive better or equivalent performance.

We further study the impact of the driving speed and steering wheel angle to the estimation accuracy. Figure 3.11(a) presents the error distribution of the steering-wheel IMU based and phone based approaches with respect to 13 bins of driving speed between 0 to 60km/h. From the figure, we can tell that the steering-wheel IMU based approach has constantly low errors with all driving speed, but the phone based approach has larger errors in lower speed range (i.e., between 5-30km/h). This is because the phone based approach cannot get accurate vehicle turning radius when the driving speed and angular velocity in the turn are low. Figure 3.11(b) presents the error distribution of the two approaches with respect to 13 bins of steering wheel angle ranging from $-180to180$. We find that estimation error of phone based approach increases dramatically when vehicle is turning, while the steering-wheel IMU based approach works well for most steering wheel angles.

Driving Speed Estimation. Next, we evaluate the performance of the driving speed estimation of BigRoad by respectively comparing the errors between the estimated driving speed and the driving speed directly obtained from the Android with the ground truth in Figure 3.12(a). From the figure we can observe that BigRoad can achieve much lower error



(a) CDF of speed estimation absolute error.



(b) Error distribution with respect to acceleration.

Figure 3.12: CDF of speed estimation absolute error.

than directly using the speed from Android. In particular, the driving speeds from the Android location service have an average error of 1.17km/h, while that of BigRoad is only 0.65km/h. The 90-percentile error of the estimated speeds from Android and BigRoad are 2.11km/h and 1.37km/h, respectively, which indicate that the delay shifting and complementary filtering techniques in BigRoad can effectively remove the large errors introduced by GPS.

To further explore the limitation of our speed estimation method, we plot the mean errors of estimated speed from Android and BigRoad with respect to the y-axis acceleration of the vehicle in Figure 3.12(b). We find that the speed estimation errors of both approaches generally increase with the accelerations of the vehicle although BigRoad can achieve lower errors. This is because both approaches mainly rely on the location updates from GPS, which have a low refresh rate and cannot capture speed changes between two samples.

3.6.3 Accuracy of External Perception

As the main part of external perception, we mainly evaluate the performance of road condition estimation in this work. As discussed in Section 3.6.1, we distribute BigRoad systems to 5 users to collect driving data in their daily commutes. In total, we collected 737 brakes from 43 driving trips in sunny days and 193 brakes from 16 driving trips on rainy days. We use 50% braking data for training the road condition models (i.e., pdf_d and pdf_w) and use the rest of data for the testing purpose.

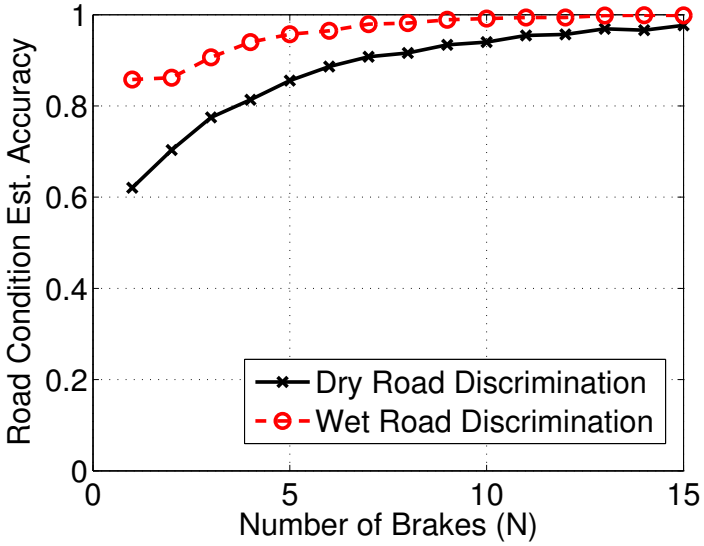


Figure 3.13: Discrimination accuracy of the road condition estimation.

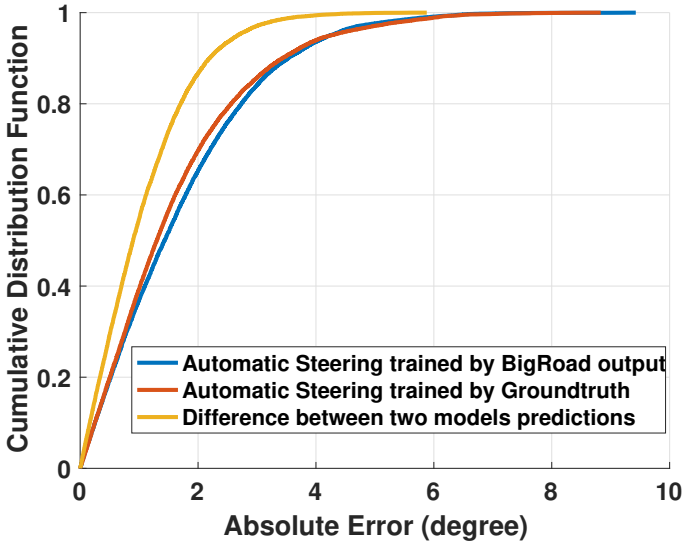


Figure 3.14: CDF of two trained Automatic Steering Algorithms and their comparison.

Specifically, we can identify the road condition by using the NTF statistics from N collected brakes according to equation 11. Figure 3.13 shows the road condition discrimination accuracy with the different number of N. We observe that BigRoad can achieve high accuracy to determine road's condition (i.e., dry or wet) by only using the normal braking events from crowdsourced driving data. And the system only needs a small number of brakes (e.g., 15) from the vehicles to achieve over 95% discrimination accuracy.

3.6.4 Usability of BigRoad Output

We demonstrate the usability of BigRoad's outputs by comparing the results of our automatic steering application when using two different pairs of driving data (i.e., the front-view video together with either the estimated steering wheel angles from BigRoad or the ground truth from the OBD). We use about 1.1 hour real-road driving data to train the deep neural network in the application, and use the rest of data for testing. The comparison of the results are shown in Figure 3.6.4. As we can see that the mean errors of the application are 1.62nd 1.7or using the steering wheel angles provided by the OBD and BigRoad, respectively. This indicates our steering angle prediction application can effectively predict steering angle by using the data from BigRoad. Moreover, we plot the CDF of the sample-to-sample difference between the prediction results from the two scenarios. We observe that the mean difference between using these two inputs are as low as 1.07 which verifies that the estimated steering wheel angle from BigRoad is comparable effective to the steering wheel angle recorded by the OBD in self-driving training. Furthermore, we find that the state-of-the-art end to end autonomous vehicle training [2] does not directly use the driver's steering angle obtained from steering sensors as a training label because the driver's input may not be perfect. They apply a computer vision based calibration to slightly correct the driver's steering angles. We note that such calibration could also be applied to BigRoad outputs before using them to train the auto steering network, which could result in an even more similar performance as that of the network trained by steering sensor readings.

Chapter 4

Automatic Unusual Driving Event Identification for Dependable Self-Driving

4.1 Introduction

While automated driving technology has made great strides, ensuring dependability over a broad set of unusual traffic situations and corner cases remains a key challenge [11–13]. Current automated driving products largely require human supervision (NHTSA level 2) [34], with only few systems that allow taking eyes off the road under very limited conditions (level 3) [35]. Self-driving without supervision in select geographic areas environments (level 4) appears to be emerging but it is still under active development and testing (e.g., [15]).

Validating such technology requires understanding the unusual events and corner cases (e.g., objects on the roadway, pedestrian crossing highway, deer standing next to the road, etc.) that one could encounter in billions of miles of driving. Such a large number of miles is needed since the goal is to achieve safety levels far above average human drivers and human drivers in the United States achieve almost 100 million vehicle miles traveled in between fatalities [14]. This motivates collecting a catalog of unusual driving events that represent challenging situations expected in billions miles of driving to accelerate the development of truly dependable level 4 and level 5 systems.

Most existing efforts collect driving data with a small fleet of tens to hundreds of highly instrumented vehicles that are continuously operated with test drivers, but it is challenging to cover billions of miles with such a small fleet. Road testing is therefore augmented with stress testing with corner cases on proving grounds and in simulation. This helps, but it remains uncertain whether a comprehensive set of corner cases was tested. Such a comprehensive set of unusual events and corner cases can be more easily obtained by scaling data collection to large numbers of minimally instrumented (camera-equipped) human driven

vehicles, as previously advocated by BigRoad [36]. This, however, would still require identifying the unusual events in such a vast dataset to create test cases for proving grounds or simulators.

Manual inspection of collected data to flag unusual driving events is one possible solution, but will require plenty of extra effort, amplify privacy concerns, and increase storage and networking overhead for collecting all data. When vehicles are primarily human-driven, we cannot simply watch for human interventions as in current self-driving prototypes to identify corner cases that the system cannot handle well. Due to the many degrees of freedom in navigating a road simply detecting discrepancies between human drivers steering and speed input and self-driving system's choice is not reliable. Although there exists a large body of work on abnormal driving event detection [37–50], this work focus on detecting specific, known situations but cannot detect previously unknown unusual road events that are missing in the current set of test cases for automated vehicles. Therefore, automatically identifying unusual driving events remains a challenge.

To address this challenge, we propose an automatic unusual driving events identification system, which can detect unusual situations through in-vehicle algorithms and can easily be scaled for wide deployment. It identifies unusual situations through a two-pronged approach involving inertial monitoring of driver reactions and an autoencoder-based technique for detecting unusual video scenes. It detects sudden driver reactions (e.g.hard braking and swerving), since situations that challenge human drivers are also likely to be interesting test cases for automated vehicles. Since sudden driver reactions usually involve accelerations and angular speed, a three stage inertial sensing approach is proposed to detect unusual braking and swerving events. The rationale for the second video-based detection algorithm is that not all corner cases which may confuse self driving system will elicit a response from a human driver. Since previously unseen corner cases are more likely to differ from the training samples, we propose autoencoder-based approaches to identify these unfamiliar views, including a detector that can run on the vehicle side on low cost devices with 71.43% accuracy and a detector partially running in the cloud with 80.3% accuracy. The performance is evaluated based on 120 hours road driving data collected by about 10 drivers. To further illustrate the efficiency of our proposed system, the vehicle end unfamiliar view

detector is implemented on an android phone and can process video frames at 17Hz, which is sufficient for flagging unusual scenes. Moreover, the unusual driving events detected by our approaches have been useful for re-training and improving the performance of the self driving model. This performance improvement on more complex road situations demonstrates the potential to accelerate the development of robust self driving systems with this unusual event detection framework. The contributions of this work can be summarized as follows:

- Introducing, to our knowledge, the first scalable unusual events identification and collection approach for self driving research and development, which employs human driven vehicles, instead of highly-instrumented vehicles.
- Developing an unusual events identification system to detect a variety of corner cases including both challenging situations for human drivers and unusual video for self driving models.
- Designing efficient unusual imagery detectors for low cost in-vehicle devices to limit the necessary video uploads, to conserve bandwidth and reduce privacy concerns.
- Analyzing more than 120 hours of driving data to evaluate the accuracy of unusual events identification and demonstrate the potential of these detected events to improve the performance of self-driving algorithms.

4.2 Unusual Events and Design Objectives

This chapter focuses on unusual events and corner cases that can confound automated driving systems. For example, these include traffic scenarios that blind sensors, scenarios where distinct lighting condition is introduced, emerging objects that are difficult to identify, and unexpected movements by traffic participants. The ultimate goal is to understand the long tail of such traffic scenarios, meaning those that one would expect to encounter only after millions of miles of driving but that still need to be handled by the system to achieve a level of robustness that far exceeds human drivers. Specifically, this chapter targets a subcategory of unusual events which can be inferred from sudden human driver reactions

and unfamiliar views and are likely to affect the performance of self driving systems.

4.2.1 Current Approaches

Current approaches to collect data and test self driving systems on unusual events can be categorized as follows. **Public road testing** has been conducted by multiple companies through a small fleet of highly instrumented vehicles across different areas. Waymo has tested their vehicle on roads for 5 million miles [51] and Uber for 2 million miles [52]. **Closed course testing** is able to stage challenging driving cases (such as people jumping out of canvas bags or porta potties on the side of the road, skateboarders lying on their boards, thrown stacks of paper in front of sensors, etc. [51]) at the test facilities, like the Castle of Waymo and the Mcity of University of Michigan. Such testing allows recreate difficult situations more frequently than they occur during public road driving and can capture data with the sensor suite of an autonomous vehicle. **Simulation testing** allows simulated testing of corner cases, and further extended testing on the variations of such corner cases by tuning many different moving angles and different speeds of vehicles for example.

The robustness challenge: The accumulated public road miles still fall far short of the number of miles needed to demonstrate a lower fatality rate than above-average human drivers. Besides, the recent fatal accident during public road testing illustrates the challenge with correctly handling a broad set of road situations [53]. Current testing regimes seek to amplify this testing of 'known' corner cases through a combination of proving ground testing and simulations. While certainly useful, it remains unclear whether this extrapolation from known corner cases can lead to the desired level of robustness or whether new categories of unknown corner cases exist that will still need to be discovered in the next billions of miles of driving. The approach that this chapter proposes is meant to complement these current techniques and addresses precisely this question.

4.2.2 Scaling Data Collection with BigRoad

As previously argued [36], current testing efforts could be accelerated through large-scale road-data collection involving hundreds of thousands of minimally instrumented, human-driven vehicles. While such vehicles may not generate sensor data that is directly usable in automated vehicles, it allows identification of new corner cases that can then be recreated on proving grounds and in simulation, as outlined above. Although [36] accurately records internal driver inputs (i.e., steering wheel angles, driving speed and acceleration) and external perceptions of road environments (i.e., road conditions and front-view video), it remains challenging to upload and store rich video data from such a large number of videos. In addition, drivers may have privacy concerns when video data is collected on their complete trips. This raises the question of whether it is possible to identify the small fraction of useful data that represents corner cases and challenging situations through preprocessing of the data inside the vehicle, which would allow uploading only these critical events.

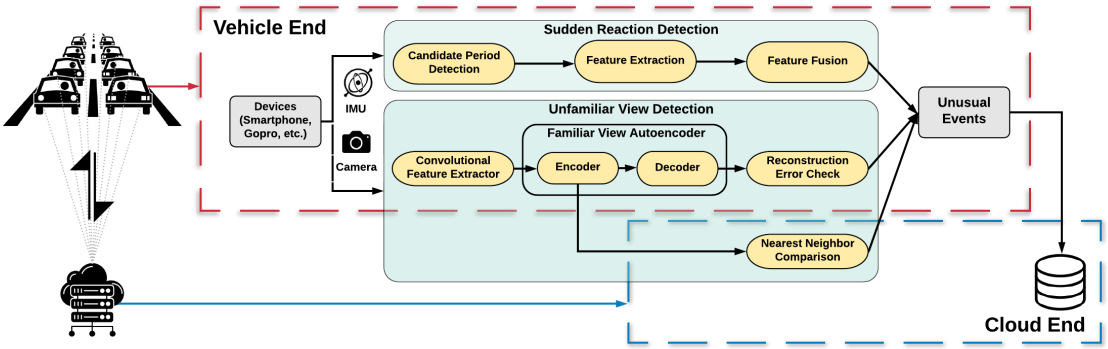


Figure 4.1: System overview of unusual events identification.

4.2.3 Design Goals

Based on the drawback of current approaches and the scaling requirement discussion above, we identify the following key design goals for a scalable automated unusual driving event collection system.

- **Sense from human driven vehicles.** Rapid scaling to hundreds of thousands of vehicles requires making use of human-driven vehicles to cover a wider range of driving areas and cumulatively gather the rare happen unusual situations. The driving event

identification system can therefore not rely on a full self-driving sensor suite but should expect minimal infrastructure and data source such as dashcams.

- **Minimize data uploads.** As the scale of data collection increases to hundreds of thousands of vehicles, tremendous wireless bandwidth usage would be required for uploading all video data into the cloud. Full uploads also increase privacy concerns. Thus, the system should be able to identify relevant events while most of the rich video data remains in the vehicle.

- **Build on off-the-shelf devices and stay within their computational limits.** Large scale data collection based on human driven vehicles will benefit from affordable off-the-shelf devices, which provide limited the computational power. Therefore, both time and space efficiency of the system should be optimized to guarantee a real time processing or near real-time computation with limited resources.

The design of our system will try to benefit automated driving systems as follows. For automatic driving components that only take front view videos and inertial readings as input, our system may be able to provide data that can be directly used as training or testing inputs. For systems which require other sensor inputs, such as radar or lidar, this data may not be available from human-driven vehicles but the detected unusual events can still help uncover traffic situations of interest and previously unknown corner cases. This information can then be used to define test cases and stage them on testing sites to collect the necessary data for other sensors to fully test the system. Besides, since our goal is to enable unusual driving events collection at very large scale, it is likely to help identify unusual driving events and develop test scenarios that will lead to more reliable automated driving systems.

4.3 System Overview

The system enables scalability through aggressive in-vehicle filtering of sensor data, which reduces the volume of data that needs to be transferred over a network, ameliorates potential privacy concerns, and saves backend (human) analysis resources. The filtering removes all data that are not classified as unusual road situations. It detects unusual events using a

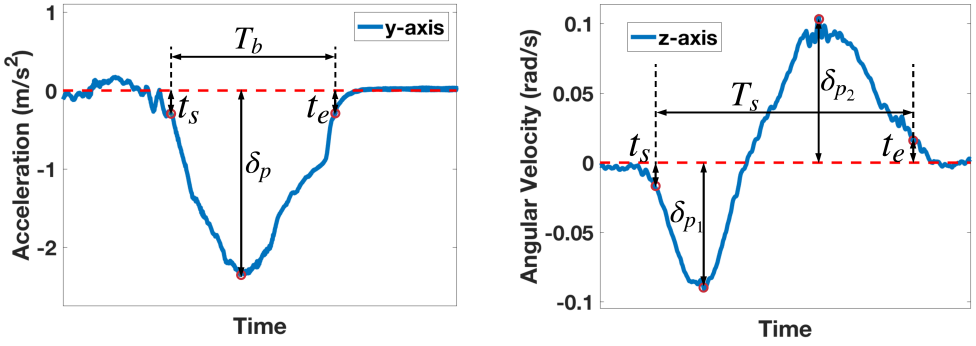
two-pronged strategy that monitors (i) how a human driver, if present, reacts in terms of steering and braking and (ii) how different the camera inputs are from previously observed inputs.

In vehicles, the system assumes an on-board device with the sensing and computational capabilities of a high-end smartphone. Specifically, it requires a camera, accelerometer and gyroscope sensors, and processing capabilities to execute neural networks, and network connectivity to allow collection of data about unusual events and corner cases. Collected data can be stored and further analyzed in the backend infrastructure, as shown in figure 4.1.

The rationale for the first filtering approach, **Sudden Reaction Detection**, is that situations that surprise a human driver are more likely to also challenge an automated driving system than more standard driving situations. Since deployment on conventional human-driven vehicles would allow reaching the necessary scale much more quickly, the system can make use of detailed measurements of the human driver's sudden steering and braking reactions to road events.

However, one can also expect situation that does not elicit a reaction from an attentive human driver, but could confuse automated driving algorithms. This motivates the two-pronged approach where the system automatically seeks to identify unusual road imagery. This could be achieved through a set of classifiers that watch out for specific situations of interest such as a deer crossing or a stroller on the roadway. This would necessarily require an enumeration of expected unusual situations and not necessarily identify the unknown unusual road situations that are the motivation for this work. For this reason, the second filter, **Unfamiliar View Detection**, evaluates how different the image appears from previously observed road video. In order to efficiently compute this on an in-vehicle unit, the system employs an autoencoder for similarity detection. If computation is also available on cloud end, a fined-grained detection could be performed.

Note that these ideas could also extend to a more complete set of self-driving sensor inputs that include radar or lidar but that this design deliberately omits them to illustrate how such a system could be deployed on large numbers of vehicles without significant instrumentation cost.



(a) Braking Event (b) Swerving Event

Figure 4.2: The accelerometer and gyroscope reading on sample braking and swerving event.

4.4 Sudden Reaction Detection

Human drivers' reactions like hard braking or high speed swerving usually involve large accelerations and angular speed. Such features can be captured by the accelerometer and gyroscope of an Inertial Measurement Unit (IMU) available in many phones, cameras, and cars. The detector is triggered when the feature score exceeds a threshold, which can be chosen as a percentile of the feature value. Therefore, to identify such sudden reaction events, we propose a three-stage inertial sensing detection technique leveraging the IMU potentially available within vehicles. The three-stage detection mechanism includes *Candidate Period Detection*, *Feature Extraction* and *Feature Fusion*.

4.4.1 Candidate Period Detection

When unusual situations happen, human drivers may react with hard braking or swerving to avoid accidents. This motivates the first stage of our detection mechanism to identify candidate periods in terms of braking and swerving events, which show relatively high amplitudes on the accelerometer and the gyroscope.

Since the pose of the IMU within the vehicle is usually unknown, the system uses coordinate alignment algorithms to project the IMU's reading from its own coordinates system to the vehicle's coordinates system. As the vehicle will only sense gravity while stationary and will have a dominating acceleration component in the driving direction when accelerating, the vehicle's coordinate system can be determined from measurements during

these situations [32]. Besides, we utilize a low pass filter to remove the noise from the raw IMU's reading caused by vehicle vibrations and bad road conditions.

Braking Event Detection. Generally speaking, when a driver brakes, a large acceleration can be observed in the opposite direction of the driving direction. Figure 4.2(a) shows the acceleration trace of a braking event, in which a negative spike can be observed due to braking. In order to more accurately capture the bumps that are actually caused by braking events, a peak detection method is applied first to find all the negative peaks of the accelerometer readings on the driving direction, whose value is defined as δ_p to quantify the amplitude of braking event as shown in figure 4.2(a). A threshold is used to remove noisy peaks such as the ones caused by road bump vibrations. Then the system searches forward and backward to find the starting point t_s and ending point t_e of this bump.

Swerving Events Detection. During swerving events, a driver usually first turns the steering wheel to one direction quickly and then turns back to the other direction. This action will result in two consecutive bumps in opposite directions of the gyroscope readings, as shown in Figure 4.2(b). Therefore, we capture such characteristics by thresholding peaks on gyroscope reading for peak detection, and then identifying swerving based on a short time interval between peaks in opposite directions. Similar to the brake detection, δ_p, t_s, t_e is defined as the amplitude of the higher peak, starting point, and ending point of a swerving event, respectively.

4.4.2 Feature Extraction

To identify the patterns of interest out of the candidate period set, we carefully select three feature extraction methods based on preliminary experiments and analysis. Based on the detected candidate periods, we first describe a **Strawman solution** using the *amplitude* of sensor readings as a feature to detect unusual events:

Amplitudes. Due to the large accelerations or gyroscope readings during unusual situations, using amplitudes of such reading as features seems to be an intuitive solution. Specifically, the sudden braking events and swerving events could be detected based on a threshold δ_p value.

However, this solution does not work well in practice because the majority of braking

events with large acceleration and swerving events with large angular velocity are normal events like braking when facing red traffic lights and swerving-like readings when changing lanes.

We therefore seek to emphasize more sudden events and propose **Derivative-based** as well as **Duration-based** features to further characterize detected events.

Derivatives. Since unusual events do not always come with high-amplitude readings, estimating the urgency or suddeness of a braking or swerving event is also helpful to determine unusual events. To this end, we calculate the derivative of acceleration to represent the urgency of a braking event. Similarly, calculating the derivatives of gyroscope reading during swerving events is used to identify urgent swerving events.

Duration. As a sudden braking or swerving event often happens in a short moment, we can also use the event duration to detect the urgency of unusual events. As shown in Figure 4.2, the duration of braking or swerving events are represented by T_b or T_s correspondingly, which is equal to the interval between t_s and t_e.

4.4.3 Feature Fusion

To effectively take advantage of all three features, we propose a feature fusion mechanism to combine the three extracted features (amplitude-based, duration-based and derivative-based) in order to increase the accuracy of identifying unusual events. This is motivated by preliminary results that showed that there is little overlap among the detected unusual events with any one of these features. We design an *accuracy driven weight assignment* method to assign weights to different features based on their detection accuracy. The principle underlying this method is illustrated in Equation 1.

$$f_{fusion} = \sum_i w_i f_i \qquad w_i = \frac{n_{f_i}}{\sum_i n_{f_i}} \tag{1}$$

The fused feature value (f_{fusion}) of a candidate period is equal to the sum of each feature value (f_i) multiplied by its weight (w_i). The value of each feature is normalized to adjust values measured on different scales to a notionally common scale. Specifically, We calculate the mean value and standard deviation of each feature, and use them to shift

and scale each feature value. The weight of each feature (w_i) is calculated based on the detection accuracy. Specifically, for each feature, we extract the top 5% potential unusual events and calculate the detected unusual events for each feature (n_{f_i}). We divide n_{f_i} by the sum of detected unusual events for all features to calculate the corresponding weight (w_i). To design a general method that works for most real world scenarios, our fusion weights are generated from a large dataset which contains different scenarios including highways, local roads, night view roads, etc. With this method, the system assigns a higher weight for features with better detection accuracy, while assigning a lower weight to the one with worse detection accuracy.

4.5 Unfamiliar View Detection

Although sudden reaction detection is able to identify events that surprise human drivers, one can still expect situations that do not elicit an attentive driver's reaction but confuse automated driving algorithms. This motivates identifying unusual road imagery by evaluating how different the image appears from previously observed self driving training images. Very different, distinct views may not have been sufficiently represented in training or test cases and lead to an increased risk of erroneous self-driving decisions. This can be intuitively implemented by calculating the Euclidean distances between new image samples and all samples used to train the automated driving system, but would require tremendous computational resources. To enable system scalability with limited in-vehicle computation, we propose two autoencoder-based unfamiliar view detectors, (i) a lightweight **in-vehicle detector** based on autoencoder's reconstruction error, and (ii) a **joint in-vehicle and cloud detector** based on the autoencoder's embedded vectors. Although autoencoders were applied to anomaly detection before [54, 55], to our knowledge, we propose the first design for driving video data with a novel architecture and loss function of the auto-encoder. This design of the autoencoder shows better performance than a naive auto-encoder application.

We develop this technique in the context of automated steering, since this is a key function of automated driving that usually heavily relies on camera data—data which can also be easily recorded from human driven vehicles [36]. Our unfamiliar view detectors will specifically identify corner cases for an end-to-end self steering systems. We expect though

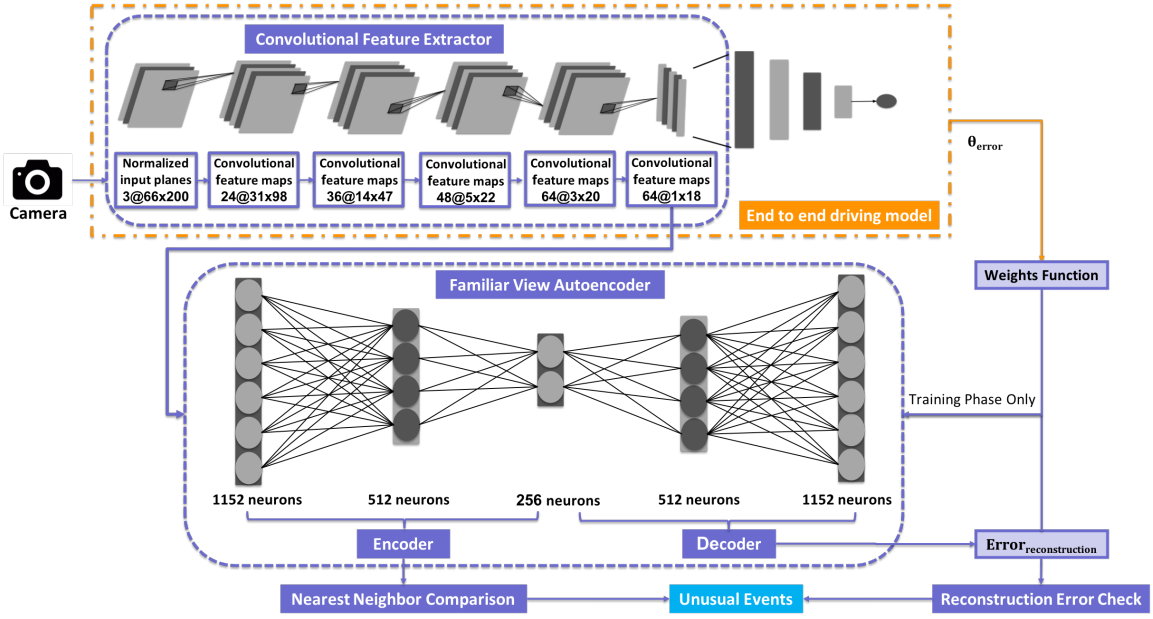


Figure 4.3: System overview of autoencoder based unusual event detection.

that the design of the detectors can also be extended to other self driving components that use camera data as input or where a comparable sensor readings can be collected from human-driven vehicles.

4.5.1 Autoencoder and Self-steering Background

Since our proposed unfamiliar view detectors are designed based on *autoencoder* for *end to end steering systems*, we will briefly introduce background on these two systems.

Autoencoder

An autoencoder is a neural network that is trained to encode the input in a set of low dimensional representations, which can be used to reconstruct an output that is nearly identical to its input. The groundtruth or labels of an autoencoder are just the input features themselves, therefore an autoencoder is also considered an unsupervised deep neural network. Internally, it has a hidden layer h that has a lower number of dimensions than the number of input dimensions, so that this hidden layer can be trained to describe an **embedded vector** used to represent the input. The network usually consists of two main parts, an encoder function $h = \phi(x)$ and a decoder that produce a reconstruction $r = \psi(h)$.

An autoencoder is designed to simply learn the set $\psi(\phi(x)) = x$, but normally will not copy perfectly because the model size is usually restricted to allow them to copy only approximately, and to copy only input that resembles the training data. Because the model prioritizes the aspects of the input which should be copied, it often learns useful properties of the data.

$$L(x, x') = \|x - x'\|^2 = \|x - \psi(\phi(x))\|^2 \tag{2}$$

Autoencoders are usually trained to minimize **reconstruction errors**, the loss ($L(x, x')$), which is defined in equation 2. The reconstruction error is a metric to quantify the distance between the input and reconstructed input, and therefore can be considered as a difference indicator between the test samples and the training samples. Since autoencoders are trained to minimize the reconstruction errors when reconstructing training samples, a test sample which is similar to training set tends to have lower reconstruction error, while a sample which is different from training samples will have higher reconstruction error.

End to end self steering system

An end to end self steering system maps raw pixels from a single front-facing camera directly to steering commands using a convolutional neural network. It is firstly proved to be powerful by Nvidia's demo [56] in lane keeping self driving tasks. Its neural network architecture [2] consists of 9 layers, including a normalization layer, 5 convolutional layers and 3 fully connected layers. The first layer of the network performs image normalization and the normalizer is hard-coded which is not adjusted in the learning process. Such normalization in the network will allow the normalization scheme to be altered with the network architecture and to be accelerated via GPU processing. The convolutional layers were designed based on different size of kernels and strides. Following five convolutional layers with three fully connected layers will lead to an output control value, which is the inverse turning radius. Based on this network architecture, a variety of end to end steering architectures are proposed. [57] [58] [59] utilize similar architecture with different layers and kernels to fit different input image dimensions. Other [60] [61] bring in recurrent neural networks

(RNN), and try to improve the performance by considering the sequential information from continuous frames. However, the majority of architectures share the common feature that starting with some convolutional layers as a feature extractor, and then use fully connected layers or RNNs to infer steering angles.

4.5.2 Design

To compare input images with previously observed training views, we propose a deep neural network architecture, which consists of a *convolutional feature extractor* to filter steering sensitive properties and a *familiar view autoencoder* to further learn the representation of well-trained samples in lower dimensionality. As shown in figure 4.3, both vehicle end and vehicle-cloud end unfamiliar view detectors are built based on this network architecture, but utilize different layer's output to achieve desired balance between accuracy and efficiency. **In-vehicle detector** leverages the outputs from the decoder of *familiar view autoencoder* to estimate the distance between an input sample and the whole training set by evaluating the reconstruction error. Since reconstruction error is calculated based on the learned distribution of the whole training set, in-vehicle detector will have relatively lower accuracy but much less computation cost as it only need perform one time pass of the neural network. While **joint in-vehicle and cloud detector** takes the outputs from the encoder of *familiar view autoencoder* to check the distance between the input sample and its nearest neighbours by encoding training samples into the same space. Such nearest neighbours comparison requires iterating pairwise distance evaluations on all the known samples, thus takes much more time especially with large amount training samples but can produce relatively higher accuracy.

As *convolutional feature extractor* and *familiar view autoencoder* are common modules of both detectors, we will introduce them in the first two subsections, and then discuss vehicle end and vehicle-cloud end unfamiliar view detectors' workflow respectively.

Convolutional Feature Extractor

Since front facing views include redundancy information that ends to end self steering system may not concern, we propose to apply a convolutional feature extractor to filter

steering sensitive properties of the inputs before feeding inputs to the autoencoder. As the convolutional layers in deep neural networks usually serve as the feature extractor, we implement an end to end driving network firstly and then utilize the convolutional layers as our convolutional feature extractor. As shown in figure 4.3, to obtain the convolutional feature extractor, we implement an end to end self steering deep neural network including 5 convolutional layers and 3 fully connected layers. Among the five convolutional layers, shallow layers which are close to input layers tend to keep more visual features, while depth layers will retain more abstract features for steering prediction. To further reduce the dimensionality and obtain abstract properties that may affect the final prediction, we choose to use all of the five convolutional layers as feature extractor. The end to end neural network is similar to Nvidia's architecture [2], whose strided convolutions are used in the first three convolutional layers with a 2×2 stride and a 5×5 kernel, and a non-strided convolution is performed with a 3×3 kernel size in the last two convolutional layers. But to get better performance on our dataset, we tuned our network to use inputs are from RGB space, take training labels with the steering angle in terms of radian, and trained only based on the center camera images. Based on our implementation, the input images will be cropped and resized to 66×200×3, and the output from convolutional feature extractor will be 64×18.

As introduced above, the convolutional feature extractor is obtained from the trained end to end driving neural network on the training set. Given a front-facing camera image X, it will be firstly processed by convolutional layers c, and then fed to fully connected layers f for steering angle prediction. Thus, if θ is the ground truth steering angle for current frame, the end to end steering prediction error is defined as equation 3, in which the overall end to end steering prediction process is $f(c(X))$ and predicted steering angle is $\hat{\theta}$.

$$\theta_{error} = \theta - \hat{\theta} = \theta - f(c(X)) \tag{3}$$

Familiar View Autoencoder

Convolutional feature extractor filters out steering sensitive properties, but the extracted feature vectors of training samples should not be treated equivalently. It is because the

unusual road views that may lead to erroneous steering predictions, are usually unseen samples which are quite different from training samples or seen samples but with relatively large training error. Thus, if a sample is close to well-trained samples, which have lower steering prediction error θ_{error} among the training set, it will more likely to be handled well by the model and identified as a usual familiar view. Besides, extracted feature vectors are still in the order of thousand dimensions, so that a fewer dimensions encoding will be helpful for larger scale system employment and data collection. To address such challenges, we design and implement a familiar view autoencoder, which learns the representation of well-trained samples in lower dimensionality based on extracted feature vectors.

The familiar view autoencoder is designed to have four fully connected layers as shown in figure 4.3. The first two layers are trained as an encoder to map input from 1152 dimensions to 256 dimensions, while the last two layers are trained as decoder to restore the 1152 dimension vectors based on embedded 256 dimension vectors. Each fully connected layer includes a ReLU activation function to prevent overfitting. Note that, the hyperparameters of the autoencoder's architecture are set empirically to achieve a better performance on unusual views detection based on current data set scale.

Sample Weighted Loss Function. To distinguish the training samples with lower steering prediction errors from the ones with larger errors, we propose a sample weighted loss function to train the autoencoder. The sample weighted loss function is designed to assign more weights on well-trained samples, so that the autoencoder will focus more on the representation of such samples which have lower steering prediction error while learning. This motivates the encoding process of familiar view autoencoder to take more properties of well-trained samples into account. Therefore, we define the weight w for each training sample in equation 4.

$$w = \frac{1}{\log_{scalar_1}(\|\theta_{error}\| * scalar_2 + bias)} \tag{4}$$

The θ_{error} is the steering prediction error as defined in equation 3. $scalar_1$ and $scalar_2$ are the two scalars used to control the range and density of weights w. Both $scalar_1$ and $scalar_2$ are monotonically increasing with weight's value if other parameters are fixed.

$scalar_1$ which is the base of the log function, decides the scale of the difference between small θ_{error} samples and large θ_{error} samples. The $bias$ is the value we set to keep the value in the valid domain of log function, which is usually the same value of $scalar_1$. Therefore, the domain of w is (0,1], and the weight w value is monotonically decreasing with θ_{error} to guarantee larger θ_{error} will have less weight while smaller θ_{error} will have larger weight.

$$L(x, x') = w * \|c(x) - \psi(\phi(c(x)))\|^2 + L_2 \tag{5}$$

Based on the weights definition, the loss function of familiar view autoencoder is defined in equation 5. Compared with traditional autoencoder loss function as introduced in equation 2, different weights are applied on different samples according to a sample's θ_{error} during end to end steering prediction training. This will put penalties on the samples, which are not trained well on end to end driving systems, to make sure familiar view autoencoder's is trained learn more characteristics of well-trained samples. Besides, we also add a L_2 regularization term on the loss function, which is defined as the euclidean norm of all the trainable weights in the autoencoder. This regularization term will help prevent over fitting and force the weights to be sparse.

In-vehicle detector

To identify unusual imagery, in-vehicle detector performs unfamiliar view detection based on the reconstruction error. As introduced above, reconstruction error is a difference indicator between an input sample and training samples, so that unusual cases can be automatically identified by reconstruction error thresholding. Based on trained convolutional feature extractor and unfamiliar view autoencoder, reconstruction errors can be obtained through $\|c(x) - \psi(\phi(c(x)))\|^2$ by comparing the difference between an input sample and its corresponding reconstructed sample produced by the decoder. Since the calculation of reconstruction error is just one time pass of the neural network, in-vehicle detector is computational efficient and able to run on lost cost embedded devices like smartphones.

Joint in-vehicle and cloud detector

Although the in-vehicle detector can identify unfamiliar images efficiently, it is challenging to detect unusual cases which do not actually have similar cases included in the training set but still relatively close to the majority of training samples. Such cases might produce lower reconstruction error but are hard to be handled by end to end driving systems. This motivates the joint in-vehicle and cloud detector to perform sample-wise distance comparison based on nearest neighbours instead of relying on the whole sample set distance evaluation according to reconstruction error. To enable the scalability for sample-wise comparison, joint in-vehicle and cloud detector takes outputs from the encoder of familiar view autoencoder in vehicle side and then performs k-nearest neighbours (kNN) checking over the cloud. Such scheme not only guarantees low computational cost in vehicle by running through part of the trained neural network, but also requires low network bandwidth since only encoded 256 dimensional floating vectors need to be transferred over the cloud. Therefore, to identify unusual imagery, an input image will be processed by convolutional feature extractor and the encoder of familiar view autoencoder to generate embedded vectors, then evaluated based on the mean distance of k nearest neighbors in the embedded space. The embedded vectors of the training samples will be pre-stored in the cloud to reduce the computation overhead.

4.6 Evaluation

We evaluate our unusual event identification system with respect to (i) the accuracy of unusual driving event detection, (ii) the efficiency of the proposed unusual event detection system, and (iii) the usefulness of extracted unusual events.

4.6.1 Dataset Description

We use two different data sets to evaluate the performance of our unusual events detection system. The sudden reaction detection is evaluated with a 120-hour dataset collected in Los Angeles, CA. In this dataset, we use GoPros mounted at the bottom center under the windshield to record the full driver's front view videos with 1280×720 resolution at 30

Fps. The $200Hz$ accelerometer and $400Hz$ gyroscope readings are also recorded using the embedded inertial sensors in GoPros. The data collection is finished by ten different drivers under different road situations, including urban road, highway roads in both daytime and nighttime.

Since this dataset we collected does not have accurate driver's steering angle while driving, we use a dataset from Udacity end-to-end driving challenge [62] to train and evaluate our unfamiliar view detection method. Note that the driver's steering angle is necessary since it will be used twofold during the experiments. Firstly, it is used to train an end to end driving neural network, part of which will serve as the feature extractor. Then, it will also be used to evaluate whether the unusual views we identified will cause poor steering angle prediction performance in automated driving systems. The dataset contains 33,808 images with a resolution of 640×480 recorded by the center front facing camera including various driving conditions, such as different sunlight conditions, roads of different lanes, etc. The videos are recorded in $20Fps$, and steering angles are logged in $50Hz$ and interpolated to be synchronized with the front view video frames.

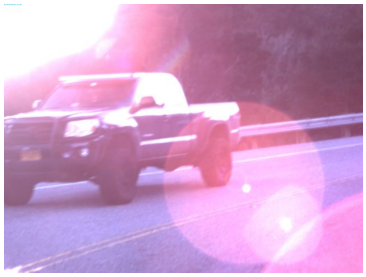
4.6.2 Unusual Event Detection Accuracy

Our system can achieve high unusual event detection accuracy for both sudden reaction detection and unfamiliar view detection, as shown in Table 4.1. Setting 98 percentile of the fused feature value as the threshold, sudden reaction detection can achieve 53.16% and 63.16% accuracy for unusual braking events and swerving events respectively. Note that the detection accuracy is low because there are plenty of general sudden maneuvers performed by drivers, such as aggressive driving behaviors, sudden braking towards red traffic lights, etc., which were detected by our method but not labeled as unusual events in the evaluation. If the 98th percentile is also used for reconstruction error thresholding in unfamiliar view detection, the accuracy of in-vehicle detector is 71.43% and vehicle-cloud end is 80.30% [1]. A sample of detected unusual events are shown in figure 4.4. Figure 4.4(a) and 4.4(b) are detected by the driver's sudden reaction based on the IMU data, and figure 4.4(c) 4.4(d) 4.4(e)

[1] Unusual view detection will be determined as correct if its corresponding steering prediction error is larger than the median.

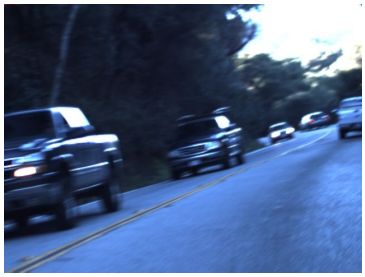
(a) Swerving to avoid a tire segment

(b) Braking due to a cut-in vehicle



(c) Special lightening condition

(d) Complex roadside lightening condition



(e) Unusual traffic and vehicle pose

Figure 4.4: Examples of detected unusual events.

are captured by unfamiliar view detector since they are relatively unusual in the dataset. Detailed evaluation procedures are introduced in the subsections below.

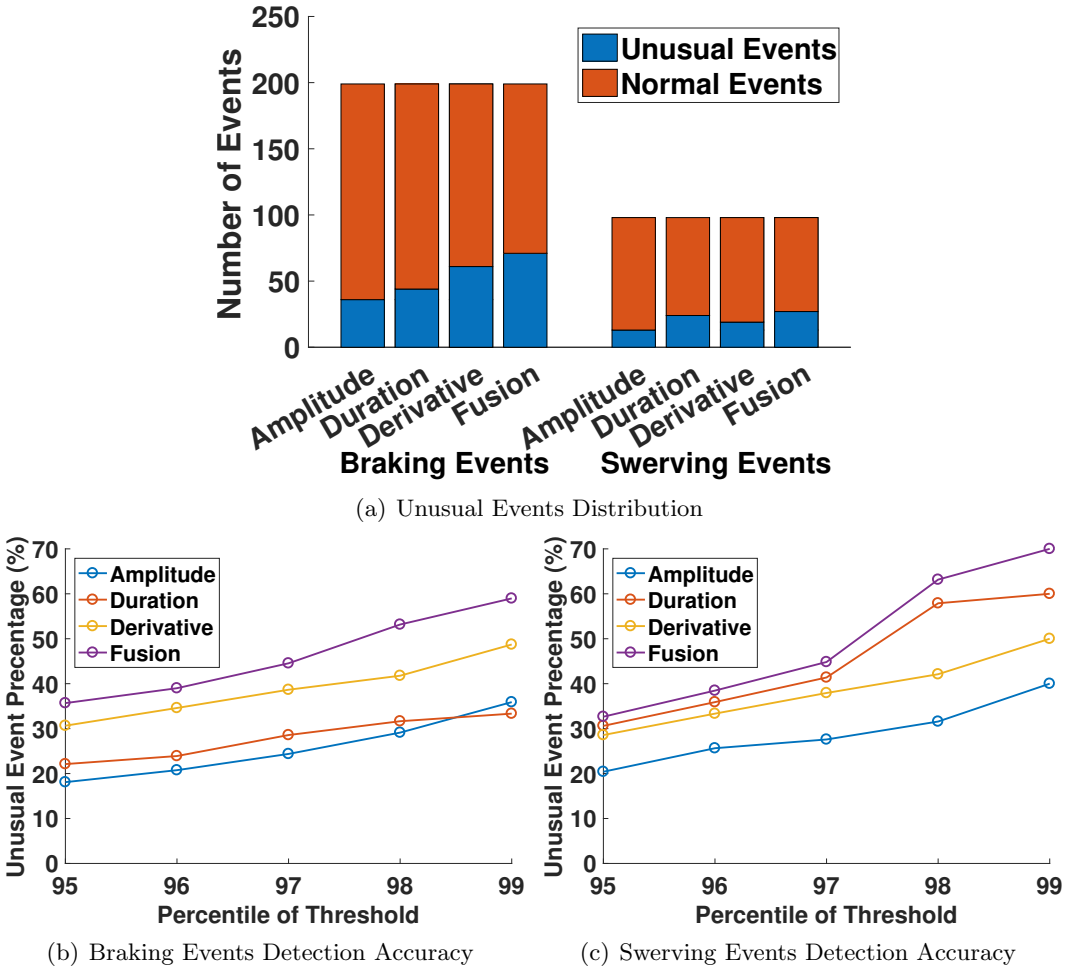
Methods	Proposed Method (%)	Baseline (%)
Sudden Reaction Detection for Braking Events	53.16	29.11
Sudden Reaction Detection for Swerving Events	63.16	31.58
Vehicle End Unfamiliar View Detection	71.43	64.53
Vehicle-Cloud End Unfamiliar View Detection	80.30	64.53

Table 4.1: Unusual event detection accuracy versus baseline/strawman solution accuracy for four methods.

Sudden Reaction Detection Evaluation

To demonstrate the performance of the sudden reaction detection method, we compare the proposed feature fusion detection method with the Strawman solution [2] as well as the approaches which filter unusual events based on the derivative and duration feature

[2]The Strawman solution is used as baseline technique for comparison.



(a) Unusual Events Distribution

(b) Braking Events Detection Accuracy

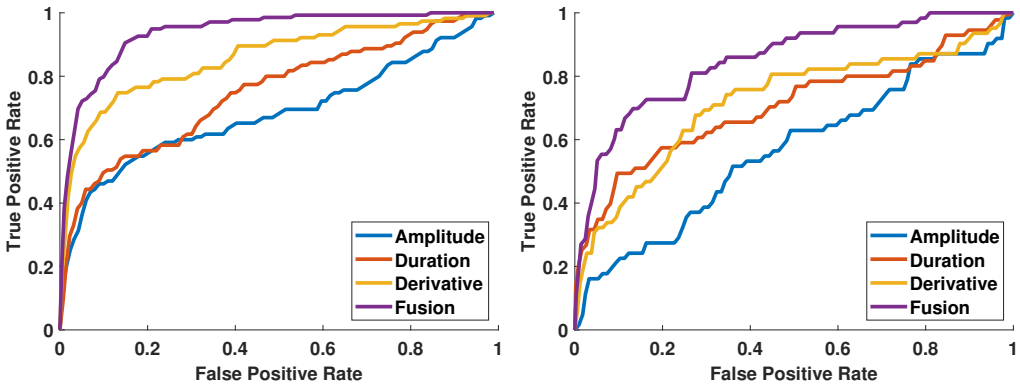
(c) Swerving Events Detection Accuracy

Figure 4.5: Evaluation Results of Sudden Reaction Detection.

individually. Specifically, detected candidate periods are sorted in terms of the amplitude of accelerations, derivative of accelerations, duration of braking events, and the fused value of all the features respectively for braking event, according to Section 4.4.

Then, unusual braking events are identified by thresholding the four metrics values to a percentile of threshold. Same process also applied for swerving detection based on gyroscope reading.

We find 3987 braking events and 981 swerving events in total over the 120 hours driving data. Thresholding the 95th percentile of braking feature values and the 90th percentile



(a) Braking Events ROC Curve (b) Swerving Events ROC Curve

Figure 4.6: Evaluation Results of Sudden Reaction Detection.

of swerving feature values [3] will filter out 199 braking events and 98 swerving events respectively, and the number of unusual events among them are shown in Figure 4.5(a). The unusual events are manually labeled in terms of whether the driver was surprised to perform sudden reactions for the unusual cases, but the usual sudden maneuvers such as aggressive driving behaviors, sudden braking towards red trafficlights, etc., are not included. Among 199 detected braking events, feature fusion approach extracts 71 unusual braking events, which surpasses the amplitude (36 detected), duration (44 detected) and derivative (61 detected) based approaches. Similarly, 27 sudden swerving events are detected out of 98 chosen swerving events with the feature fusion approach, which is better than other approaches (13, 24 and 19 sudden swerving events detected correspondingly). Since the length of unusual events captured by feature fusion is 0.25 hours out of 120 hours driving, our sudden reaction detection can largely save the bandwidth by only uploading detected unusual situations.

To further explore the relationship between sudden reaction detection accuracy and percentile of threshold value for each method, we plot the accuracy for braking events detection in Figure 4.5(b) and swerving events detection in Figure 4.5(c). We can observe that as the percentile of threshold increasing, the detection accuracy all of the four methods are getting higher. Among the four approaches, our proposed fusion method performs better than the other three methods. Since the dataset is too large to label ground truth for every

[3] We chose 90 percentile as threshold for swerving events because smaller amounts of swerving events are included in the dataset.

event, we do not evaluate the recall of this approach in this chapter.

Besides the precision improvement, our system also achieves a high estimated recall compared to the baseline approaches. Due to the large amount of manual effort needed to label the large video dataset with ground truth, we limited labeling to 40% of events with at least one high feature value (a total of 1878 labeled events) and calculate recall over this dataset as an estimate for overall recall. Figure 4.6 shows the ROC curve of our braking events detection and sudden swerving events detection methods correspondingly. We can observe that the feature fusion approach achieves the highest area under the curve (AUC) value, which indicates the performance improvement compared to baseline approaches.

Unfamiliar View Detection Evaluation

In this section, we evaluate our unfamiliar view detection method by comparing the proposed two detectors (in-vehicle detector and joint in-vehicle and cloud detector) with a baseline approach. In particular, we randomly sampled 80% of video frames from the Udacity dataset [62], and use them to train all of the three models. The rest of 20% of video frames are served as test set to evaluate the performance in terms of detection precision and recall.

Baseline Model. In order to show the advantage of our in-vehicle and joint in-vehicle and cloud detectors, we compare our system with a baseline model which uses the raw training images as input for the autoencoder. The baseline model also includes 4 layers: (i) a convolutional layer taking inputs images with $60\times200\times3$ dimensions and apply a $[5, 5]$ kernel, (ii) a fully connected layer further encode inputs to 256 dimensions. (iii) a fully connected layer to decode the inputs from 256 dimensions, and then (iv) a deconvolutional layer to reconstruct input images back to $60\times200\times3$ dimensions also with a $[5, 5]$ kernel. The baseline autoencoder is trained on the same training set as used for our proposed autoencoder. Figure 4.7 shows that the baseline autoencoder is able to reconstruct the input images to similar lossy images, since the embedded layer is much smaller than the original input. Thus, the reconstruction error of baseline autoecnoder can also indicate the distance between a test sample and the training set.

Precision. To evaluate unfamiliar view detectors, we define p in equation 6 as the

Figure 4.7: Sample input images and corresponding reconstructed images of baseline autoencoder.

detection accuracy, which represents the precision for binary classification task. Since unfamiliar view detection aims to identify the corner cases which are challenging for end to end self driving system, we determine a correct unusual detection if the sample's steering prediction error θ_{error} is larger than a threshold $thre_\theta$.

$$p = \frac{\text{number of detected events whose } \theta_{error} > thre_\theta}{\text{number of detected events}} \tag{6}$$

Figure 4.8 shows the detection accuracy p with respect to (i) different thresholds value to identify unusual events and (ii) different $thre_\theta$ to determine correct detections. As illustrated in the legend, the red, blue and yellow curves in the figure represent the detection accuracy p of three approaches respectively. x axis is defined as threshold value to identify unusual events, which is percentile of reconstruction error for baseline detector and in-vehicle detector, and the percentile of 20 nearest neighbour's distance for joint in-vehicle and cloud detector. The $thre_\theta$ is set to 50 percentile, 70 percentile, 90 percentile of the steering prediction error on test set to define different correct unusual events detections. We can observe that all the curves have larger p while the threshold value of x axis increases. This verifies our motivation that driving views which have larger distances with previously observed views are more likely to get poor steering predictions. When x axis value is close to 0, the curves start from 0.5, 0.3, 0.1 respectively, which is basically random guess. However, as the distance threshold on x axis increasing to a larger value, such as 90 percentile, p boost to a very high accuracy. This illustrates that the events which have large distances namely very different from the training samples are more likely to confuse end to end driving system

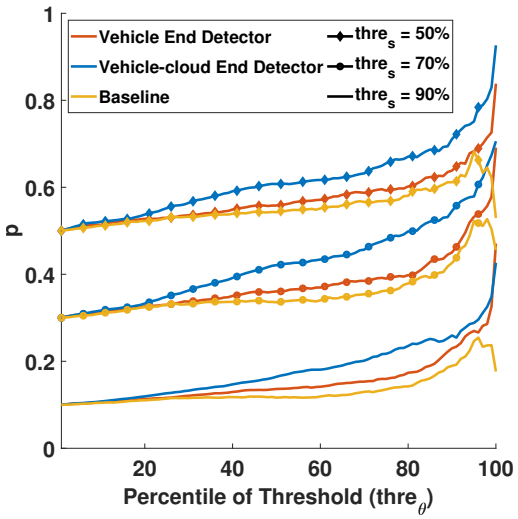


Figure 4.8: The relationship between detection precision and threshold value.

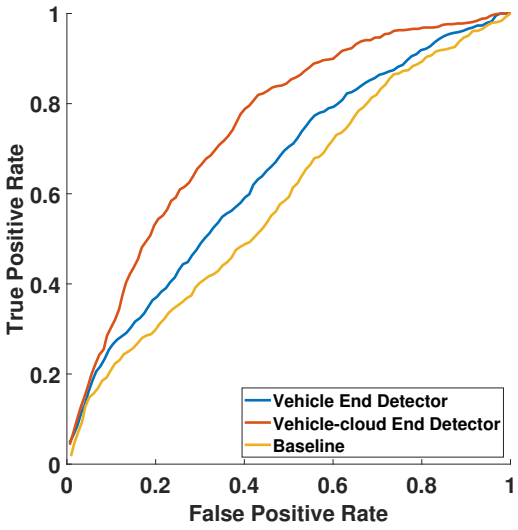


Figure 4.9: The ROC curve of baseline, in-vehicle, joint in-vehicle and cloud detectors.

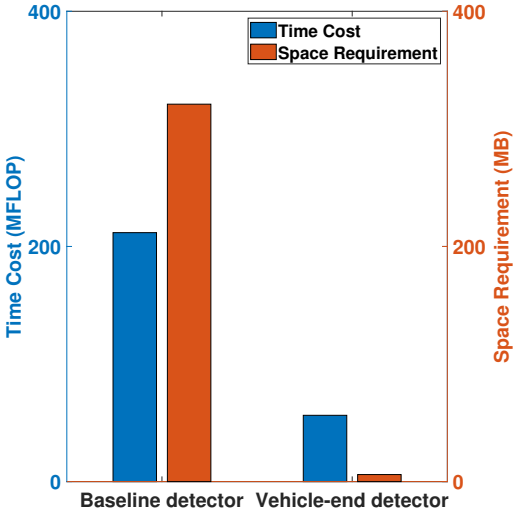


Figure 4.10: Efficiency comparison between baseline detector and in-vehicle detector.

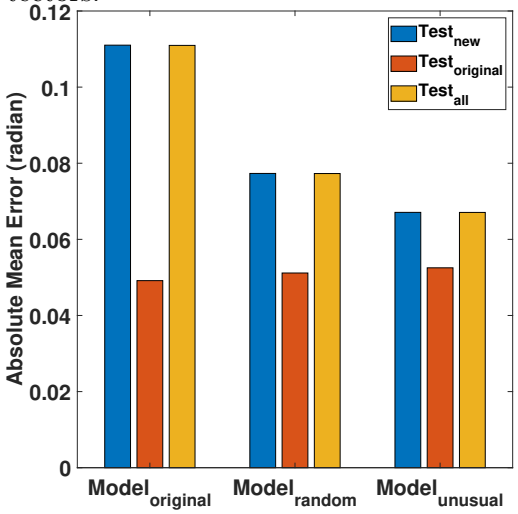


Figure 4.11: Model performance comparison on different training sets and test sets.

and get poor predictions.

For the baseline autoencoder, although it can also quantify the distances between test samples and training samples, the accuracy p is not as high as the two other approaches. It is because that the reconstruction error of baseline autoencoder's is estimated without discrimination, while our familiar view autoencoder uses the feature map from the convolutional feature extractor and focuses on the well-trained samples through weighted loss function. Thus, our approach drops the redundant information and emphasizes more on

the information end to end driving model cares about. Besides, the droppings of p at the tails of baseline autoencoders curves also show that even though some images may visually different from training set, they still work for steering prediction model with similar feature vectors of training samples. For the comparison between in-vehicle detector and joint in-vehicle and cloud detector, the latter one is usually more accurate, since it performs fine-grained sample-wise calculation as discussed in section 4.5.2. Therefore, the experiments show that our unfamiliar view detectors can accurately detect unusual events which self driving models fail to perform good predictions.

Area under the curve. To further explore the performance of unfamiliar view detectors, we plot the Receiver Operating Characteristic (ROC) curves in figure 4.9 with 90 percentile steering prediction error to define unusual events. The area under the curve (AUC) of baseline, in-vehicle, joint in-vehicle and cloud detectors are 0.59, 0.64, 0.75 respectively. Although our proposed detectors work better than baseline detection, they still do not achieve a high AUC score. This is because that the trained end to end driving model could not work well on the views which are similar to previously seen training samples and causes a lower true positive rate. Therefore, it is important to consider the balance between a detector's ability to capture most of unusual events and the cost of network bandwidth for data transmission as high recall to extract more unusual events will bring more false positive detection. Under the context of large scale employment and data collection, the system will more likely to focus on extremely unusual cases with minimum bandwidth which prefers a higher precision.

4.6.3 Efficiency of Unusual Events Detection

As the unusual events identification system is built towards large scale employment based on off-the-shelf devices, we further explore the time and space efficiency of our method. Since sudden driver reaction detection through inertial data is computational inexpensive and does not require extra storage to perform the detection, we only focus on the efficiency evaluation on unfamiliar view detection in this section.

The space requirement of baseline autoencoder and in-vehicle detector are evaluated through the size of inference graph in tensorflow. To obtain the inference graph, we run

through the freezing and optimizing scripts of tensorflow to organize the trained model with only inference required nodes. The size of inference graphs for baseline autoencoder and in-vehicle detector are 321MB and 6MB respectively as shown in figure 4.10. We also evaluate the baseline autoencoder and in-vehicle detector based on the number of floating points operations (FLOP), which can be used to roughly estimate the time cost of models. The FLOP of both models are counted by the benchmark script provided by tensorflow. As shown in the figure 4.10, baseline autoencoder model include 211.74 MFLOPs, which is much higher than online in-vehicle detector's 56.39 MFLOPs. In addition, we also run the in-vehicle detector on a Nexus 5X Android smartphone, and each frame takes around 58ms to process, which showing a 17Hz inference capability. Therefore, the in-vehicle detector has higher efficiency in terms of both time cost and space requirement, and is applicable for real-time employment on off-the-shelf smartphones. Regarding joint in-vehicle and cloud detector, it will be more computational expensive since its complexity is proportional to the number of training samples, and requires significant space due to the storage of training sample vectors.

4.6.4 Usefulness of Unusual Events

In this section, we show that the unusual events extracted by our unfamiliar view detection method can increase the performance of end to end driving system for more complex situations. Specifically, we add the detected unfamiliar events into the training set of the end to end steering system and explore the performance on different test sets.

First, based on the dataset from Udacity as introduced in section 4.6.1, we use 80% of the samples to train an end to end driving model $\text{Model}_{\text{original}}$ and the rest of 20% samples as the test set $\text{Test}_{\text{original}}$. To get new training samples, we use another dataset also provided by Udacity [63], which is collected with the same experiment setup, but mostly driving on roads with heavier traffic. Among this dataset, first 10,000 samples are picked as a sample pool and the next 1000 samples are used as the new test set $Test_{new}$. The new training sets include original training samples as well as 1000 new samples, which are selected from the sample pool by two different strategies. One strategy randomly selects new samples from the sample pool, and the other one utilizes in-vehicle detector to pick 1000 unfamiliar

views. We call the model trained with the first strategy $\text{Model}_{\text{random}}$, and the second model $\text{Model}_{\text{unusual}}$. To evaluate the overall performance on both datasets, we combine the samples from $\text{Test}_{\text{original}}$ and Test_{new} as Test_{all}.

All of the three different models are evaluated on three different test sets in terms of the absolute mean error of steering prediction, and the performance is shown in Figure 4.11. For the new test set Test_{new}, $\text{Model}_{\text{original}}$ performs worst since it had never trained on those roads. $\text{Model}_{\text{random}}$ shows better performance than $\text{Model}_{\text{original}}$, since the new training set contains randomly selected views with the same road condition. $Model_{unusual}$ further shows performance gain, which illustrates that the unfamiliar cases identified by our method is more representative of the new dataset and improve the model performance on the new route more efficiently. For the original test set $\text{Test}_{\text{original}}$, $\text{Model}_{\text{unusual}}$ and $\text{Model}_{\text{random}}$'s accuracy are slightly lower than $\text{Model}_{\text{original}}$, since they are trained to handle more cases, which create a neglectable performance compensation on $\text{Test}_{\text{original}}$. From the results based on the overall test set Test_{all}, $\text{Model}_{\text{unusual}}$ still shows the best performance than the other two, which shows the detected unusual events are able to increase model's overall performance and robustness on different roads and traffic conditions.

4.7 Related Work

Detecting unusual events is of great importance to driving assistant system developments, since the analysis of such corner cases will be helpful to improve current systems [2,34,57–61, 64]. There has been extensive research on vehicle sensing and unusual events detection based on inertial measurements. [37] utilizes the IMU of smartphones to detect and differentiate vehicle maneuvers, but only focusing on steering related maneuvers. [38] also take inertial reading from smartphones, and then perform aggressive driving style recognition based on dynamic time warping. However, such algorithm could only detect known aggressive driving patterns, while not diverse human driver reactions which are naturally performed during emergency periods. [39] is close to our vision that identifies unusual events based on inertial measurements, but the proposed solution heavily relies on large number of thresholds which are defined based on the amplitude of sensor readings. Besides, there are some other driving sensing techniques built upon IMU, but towards different goals, such as [65] for

drunk detection, [32, 66, 67] for driver determination, and [68, 69] for driver tracking, etc.

As visual imagery captures vehicle's surrounding condition and moving pattern, vision based techniques are also used to detect unusual driving situations. [40] [41] [42] leverage the spatial and temporal information from videos to detect high level anomaly patterns by comparing with previously observed views. While [43] and [44] specialized on concrete type of road emergency detection such as a collision based on vehicle tracking. However, these approaches rely on a fixed point of view, such as video feeding from surveillance camera, thus are not suitable for on-board imagery processing. [45] and [46] are able to utilize on-board cameras to capture unusual cases, but only cover on a subset of challenging situations such as abnormal pedestrian movements and unclear drivable roads.

Another body of work focuses on exploring the weakness and vulnerability of current self driving systems. [47] [48] [49] could be used to generate the synthetic cases which systems tend to have erroneous behaviors, but such situations can not extend the coverage of real unusual cases. [50] explored the robustness of traffic sign recognition model under physical world attacks. However, a detection mechanism for such attacks was not covered.

4.8 Discussion

As more unusual driving events get collected by our proposed system, the previously obtained training set need to be extended to cover more diverse cases. This will require the familiar view autoencoder to be re-trained with more number of observations. Depending on the order of increased training samples, the familiar view autoencoder may have to incorporate more neurons and more layers to achieve equivalent detection performance. Such updates on the network architecture will increase the computational cost of the familiar view autoencoder with larger trained network size and longer processing time. To mitigate the computation requirement on the vehicle, the decoder and reconstruction error check module of the detector could be shifted to the cloud. Since the computational cost of the autoencoder is not proportional to the number of samples, the shifting is only necessary when the size of the training set is very large.

The low cost design of our unusual identification approach could enable large scale

data collection potentially through crowd sourcing, but privacy could be a concern. When unusual events are recorded, drivers' privacy may be compromised by the imagery and inertial sensor readings, such as visited location which can be inferred from the front view camera, and driving decisions for an emergency event sensed from inertial sensor. Therefore, the vehicle end software may need additional functionality to buffer the detected unusual events and only upload the events if the driver confirms that there is no privacy concern. Besides, as the scale of crowd sourcing increases, the heterogeneity of mobile devices may become another challenge. For the inertial based sudden reaction events, there may be differences in sample frequency and sensitivity but we expect this to cause relatively low accuracy loss. It would be more challenging for the unfamiliar view detection to run on heterogeneous cameras with different resolutions and intrinsic parameters but this can be addressed in future work.

4.9 Conclusion

In this chapter, we aim to automatically identify unusual driving events to allow scaling the collection of driving data and corner cases to a much larger fleet of human-driven vehicles without requiring upload or human review of all data. The proposed system is able to capture various unusual circumstances, including hazardous event like sudden braking and swerving events through a three-stage process involving inertial sensing and detecting outliers of current trained self-driving systems based on autoencoder deep neural network. The evaluation is based on more than 120 hours of real road driving data and shows that it outperforms baseline methods on unusual event with 82% accuracy improvement over baseline on sudden reaction detection and above 71% accuracy on unfamiliar views identification. While the dataset size allowed validating the techniques only with less rare events, one might expect that similar patterns hold also in the much rarer events that would eventually be of interest. The event identification process requires only inertial measurements and front view driving videos, allowing collection of data from smartphones or dashcams. The computational cost is only subject to the complexity of our pre-trained neural network. Thus, the light-weight design and minimal infrastructure requirement of this approach will

allow large-scale unusual driving events identification and collection. We hope that an extensive dataset of driving corner cases collected with this approach would provide a better understanding of potential limitations of current systems and accelerate the development of robust automated driving technology.

Chapter 5

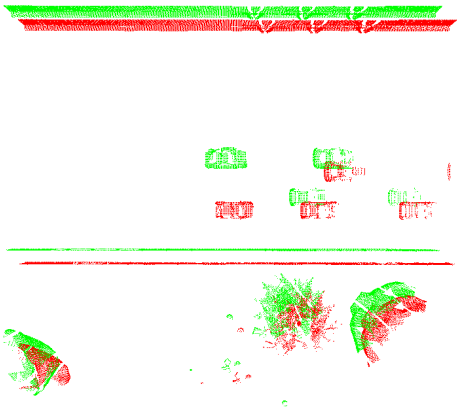
Bridging the Gap Between Point Cloud Registration and Connected Vehicles

5.1 Introduction

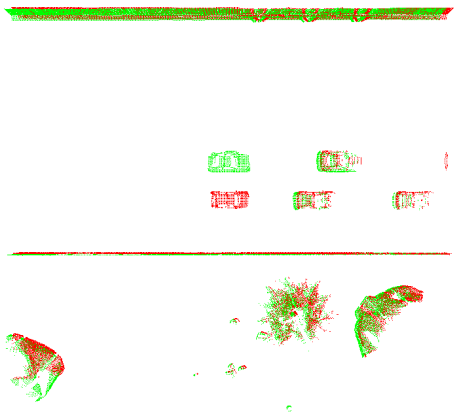
As driving is becoming increasingly automated, vehicles rely on multiple sensors (e.g., ultrasound, cameras, RADAR, or LiDAR) to maintain comprehensive awareness of the surrounding traffic environment. While much progress has been made, it remains challenging to ensure the dependability over the long tail of events and traffic situations that vehicles can encounter. In particular, vehicles must contend with: (i) physical occlusions, in which objects are blocked by others and are only partially observable or unobservable; (ii) sensing limitations, including field of view, resolving power, or lighting conditions that may limit the sensing range and quality. Connected vehicles have the potential to overcome such limitations by sharing observations across a wireless network and merging them across different vehicles, since such physical occlusions and sensing limitations from one perspective can often be easily addressed when viewing the scene from a different perspective.

In this chapter, we specifically focus on the fusion of 3D point clouds from different vehicles, which are usually generated by stereo cameras or LiDARs, and broadly used for on-vehicle applications such as object detection, object tracking, etc. The previous work [70] has shown that the object detection accuracy can be improved about 10% for the detection within 20 meters and 30% for longer distances by fusing the point clouds from other viewpoints. In order to benefit from such point cloud fusion in real world, one main challenge is to align point clouds captured by different vehicles. Since the non-negligible vehicle localization error in real system will make the simply merged point clouds even more noisy, point clouds from different vehicles should be well aligned before feeding to applications. Although

(a) Simulation sample snapshot 1



(b) Input point cloud with localization error of snapshot 1

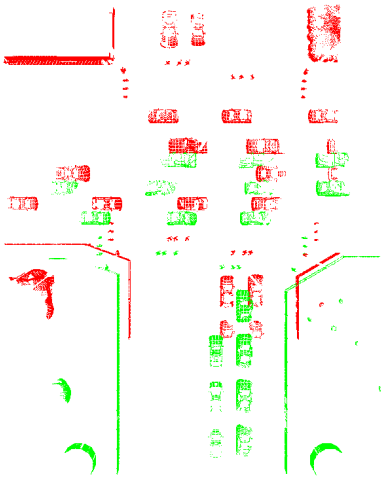


(c) ICP alignment result (mean error = 0.34m) of snapshot 1

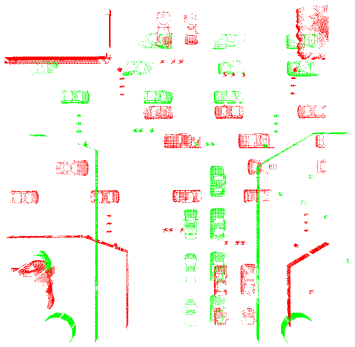
(d) Alignment ground truth of snapshot 1

Figure 5.1: Illustration of ICP point cloud registration performance in bird's eye view.

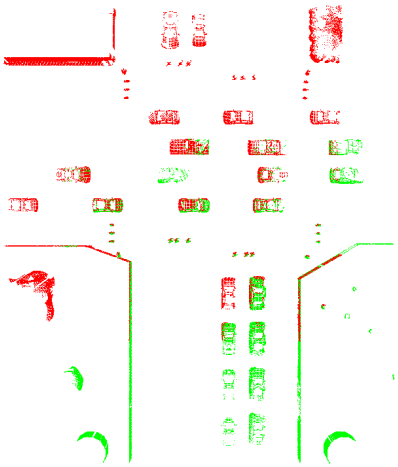
(a) Simulation sample snapshot 2



(b) Input point cloud with localization error of snapshot 2



(c) ICP alignment result (mean error = 13.21m) of snapshot 2



(d) Alignment ground truth of snapshot 2

Figure 5.2: Illustration of ICP point cloud registration performance in bird's eye view.

there has been extensive research on point cloud alignment/registration, the state-of-the-art methods cannot be directly applied to align the pairwise point clouds from vehicles, as they require large overlapping ratio between point cloud pairs [71–74]. (Note that the scope of this chapter is to align point cloud pairs without high definition maps, since they are expensive to create and maintain, and only available for limited areas). Due to occlusions from surrounding objects or vehicles perceiving the scene from different directions, the observations from different vehicles usually have little overlap ratio and fail to be aligned by the state-of-the-art point cloud registration algorithms. Considering the most widely used point cloud registration algorithm, Iterative Closet Point (ICP) [73], as an example, the alignment results are shown in figure 5.1 and 5.2. When vehicles are close to each other and driving towards the same direction as shown in figure 5.1(a) marked with red and green rectangles, ICP can align the point clouds from these two vehicles when decimeter level localization error is introduced, as (i) the inputs combined with localization error could still be considered as a good initialization and (ii) there are large enough overlapping ratio. However, as for the scene in figure 5.2(a), the overlapping ratio will become much lower since vehicles are driving towards different direction and there are objects in between. Thus, ICP fails to fuse the point clouds accurately as shown in figure 5.2(c). The requirement of the overlapping ratio for the state-of-the-art point cloud registration methods largely restricts the potential peer vehicles which can benefit from the vehicle networks based point cloud sharing.

To overcome such limitations, we design a two-phase point cloud alignment system that can fuse point cloud accurately even when vehicles have large viewpoint difference. Our intuition is to detect the overlapping region between two views and only align point clouds based on that, so that the overlap ratio of input point clouds could be largely increased. Specifically, the system first identifies and matches co-visible objects, that is objects visible from both perspectives, using hyper-graph matching based on the extracted location and label information. It then estimates the co-visible region for each of them and cropped out the larger overlap region. The selected co-visible area acts as an anchor points and its point cloud will be used to estimate the transformation. The estimated transformation will then be applied to the entire point cloud from the same viewpoint. We

evaluate the accuracy of point cloud registration and co-visible matching based on both real-world KITTI [75, 76] dataset and synthetic CARLA [48] dataset. Our contribution can be summarized as followed:

- We propose the first system which can accurately align point cloud pairs under complex traffic conditions, such as scenes with various occlusions and large view angle difference[1] (e.g., 90 180.
- We introduce a technique to identify co-visible objects by combining multiple similarity metrics obtained in 3D object detection results to distinguish co-visible objects from single-visible objects.
- We show that fusion accuracy is improved when point cloud registration is focused on the co-visible object with the overlapping area among the views.
- We evaluate our system based on both synthetic scenes and real-world experimental data across highway and intersection scenarios and show that it can improve point cloud registration algorithms with a significant margin.

Note that this chapter is not proposing new point cloud registration algorithms, but enable existing ones to be applicable in complex traffic scenes, which further activates the potential of vehicle network based data sharing.

5.2 Related Work

As our work lies at the intersection of point cloud registration and vehicle information fusion, the related work in these two areas is summarized in this section.

The term "point cloud" refers to a set of data points in 3D space which are usually used to represent objects or scenes. Since point clouds are generally produced by depth-capable sensors such as LiDAR [77] or RGBD cameras [78] with a partial view of a scene, two or more partially overlapping point clouds are often combined to represent the full 3D geometry of the sensing region. This process of finding a translation and rotation transformation of

[1] defined as the bearing difference between two vehicles.

one point cloud so that the overlapping portion matches that of another point cloud is called point cloud registration or alignment. Note that, the terminology **point cloud registration or alignment** in this chapter refers to the specific algorithms to match the point clouds, and **point cloud fusion** refers the complete pipeline of combing point clouds from a system perspective, including prepossessing, sharing, and registration or alignment.

5.2.1 Point Cloud Registration

Pairwise point cloud registration

Generally, there are two popular paradigms for point cloud registration: *correspondence*-based methods and *correspondence-free* methods, depending on whether correspondences between point clouds are extracted explicitly.

Correspondence-based methods first detect and match 3D keypoints across point clouds and then infer the transformation from these putative correspondences. Since it is too inaccurate to match points based on position alone, the matching process is based on features the describe the shape surrounding a point. Traditional hand-crafted features commonly summarize pairwise or higher-order relationships in histograms such as Fast Point Feature Histograms [79], Viewpoint Feature Histograms [80], or Clustered Viewpoint Feature Histograms [81]. With the development of deep learning, a number of neural network based feature descriptors have been proposed, such as PointNet [82], 3DMatch [83], PPFNet [84], 3DSmoothNet [71], Multi-view Descriptor [72], FCGF [85] etc. Although the robustness of the learned 3D descriptors is improved compared to the hand-crafted features, their registration pipelines still rely on the same process of matching geometric features across point clouds. All these methods require significant overlap between two point clouds to accurately combine them. For example, the evaluation of these methods, such as [71, 72, 84], require the input point cloud pair to overlap by at least 30%, which is not necessarily the case when vehicles approach an intersection from different directions as in Fig. 5.1 and 5.2.

Iterative Closest Point (ICP) [73] and its variants, e.g., point-to-plane ICP [86] and point-to-line ICP [87], are the most commonly used correspondence-free methods. These

algorithms perform optimization by iteratively refining a point correspondence and the associated rotation from an assumed starting correspondence, but they are not robust against outliers and converge to a global optimum only when starting with a reasonable initial alignment. To overcome this, correspondence-based methods can be used before ICP to provide the coarse alignment. To remove the need for initial alignment, recent work either integrates such two stage registrations into an end-to-end learning algorithm [74, 88, 89] or proposes non-training global registration pipelines [90–92]. Moreover, NDT [93] represents point clouds by a combination of normal distributions to apply standard numerical optimization, and TEASER [94] reformulate the registration problem using a truncated least squares to yield a fast computation and provides readily checkable conditions to verify if the returned solution is optimal. Although the applicability of pairwise registration methods are extended, they still cannot work for vehicle point cloud registration directly due to the high outlier ratio caused by distinct vehicle views and occlusions.

Scene-based Optimization

The aforementioned methods can register point clouds in a pairwise manner, but the ambiguous cases that arise in pairwise matching can be mitigated by incorporating cues from multiple views. Projects such as [95–97] posed the task of finding a global alignment as picking the best candidates from a set of putative pairwise registrations, such that they satisfy the loop constraints. However, such approaches are less desirable for vehicle point cloud generation since they require the presence of a larger numbers of neighbouring vehicles that share their point cloud in order to perform single registration. Similar to our setting, the most relevant work to ours is arguably [98], which matches and aligns point clouds in different LiDAR scans. It can recover the correct alignment over larger vehicle displacements, when vehicles are traveling in the same direction, but not the intersection scenario we consider. Finally, although [99, 100] aim to register vehicle point clouds at the object level, these methods assume a complete point cloud of the surroundings from a high-resolution 3D map as inputs. Given the substantial overhead of maintaining such maps, we aim for a map-free solution.

Baselines. In order to illustrate our contribution, 3 baseline algorithms are chosen

as the benchmark for point cloud performance comparison, including (1) the most widely used correspondence-free registration algorithm ICP [73] (2) deep learning based feature descriptor FCGF [85] which is one of the most recent progress on correspondence based registration algorithm, and (3) the closest work to ours SSM [98].

5.2.2 Vehicle-to-Vehicle Information Fusion

Other pioneering work shows the potential benefit of vehicle information fusion, but did not consider the various vehicle viewpoint differences and localization errors at intersection scenarios as we do. [101] proposed to fuse vehicle information for perception, but focused on fusing compressed LiDAR features. Although [102] studies the full stack of multi-vehicle cooperative perception and driving, the design and implementation are limited to when vehicles are following each other. [103] proposed a system to share vehicle's view through extracted features using SLAM [104]. However, the focus of their proposed system is on visualizing and reconstructing the shared camera view. Both [70] and [105] explored the benefits of point cloud fusion, but no localization error was involved in the pipeline, which always leads to perfect point cloud fusion.

5.3 System Overview

Based on vehicular application scenarios and foregoing review of the state-of-the-art in point cloud registration algorithms we identify the following challenges for point cloud fusion across vehicles:

- Aligning point clouds in complex traffic situations. Complex traffic scenes challenge the point clouds registration between vehicles in two aspects: (i) the presence of multiple traffic participants leads to participants experiencing different occlusions in their observations. This significantly decreases the amount of overlap between point clouds from different vehicle; (ii) the same object can be observed by vehicles from very different view angles, for example when approaching from different legs of an intersection. The resultant observations can thus contain the same object observed from different sides, which again leads to relatively distinct point clouds with little

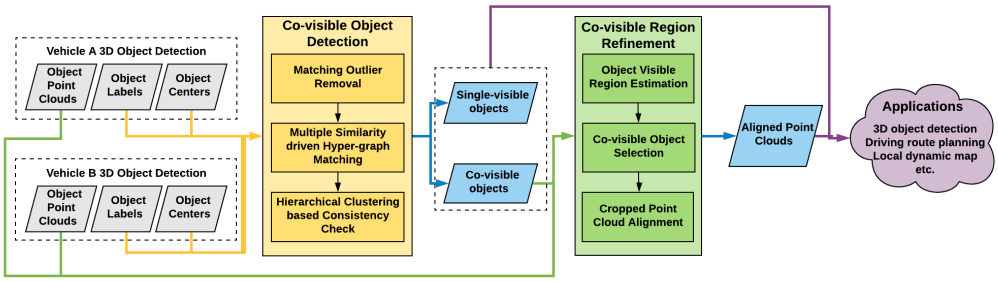


Figure 5.3: System Design.

overlap.

- Limit bandwidth consumption. The system should be able to exchange any required data over a wireless network between vehicles. While this information does not necessarily have to be exchanged over very bandwidth-limited technologies such as Dedicated Short Range Communications (DSRC), the system should be able to exchange the data over emerging technologies such as millimeter-wave (mmWave) communications.

- Tolerate vehicle localization errors. Vehicle localization errors will affect the transformation from vehicle sensor coordinates to world coordinates, which is based on vehicle locations. Although increasing sensor capabilities and improved localization lead to more accurate vehicle localization, the system should still be able to handle localization errors at least at the decimeter level.

In order to merge vehicle point clouds and meet the design objectives, we propose a two-phase point cloud fusion system which first identifies objects that are co-visible from each vehicle's perspective and then refine the point clouds based on co-visible region of these objects to align the point clouds. The system is designed based on the outputs of 3D object detection, since it is usually available when the vehicle has depth sensing capability and the robust 3D object detection results provide reliable hints during point cloud fusion. As shown in figure 5.3, the input of the system based on the 3D object detection results include the labels, centers, and point clouds of detected objects. The coordinates of inputs

are transformed into world coordinates based on each vehicle's own localization. The *Co-visible Object Detection* module extracts multiple similarity metrics based on the detected object labels and centers to distinguish co-visible objects from those that are visible only from a single perspective. Even though co-visible objects can be observed from two vehicles' viewpoints, they may not have enough overlapping visible area to yield an accurate point cloud registration. Thus, *Co-visible Region Refinement* further trim the point cloud to keep the overlapping area of the co-visible objects for transformation estimation between two views.

The resulting aligned point clouds can be combined to create a more complete representation of the traffic scene, which better supports advanced driving assistance applications. Note that not all pairs of vehicle point clouds can be fused in our system, only the ones include co-visible objects and co-visible areas can be fused by *Co-visible Object Detection* and *Co-visible Region Refinement*, respectively. If fusion is not possible, vehicles can fall back on their individual perception. The detailed fusion requirements of each module will be discussed in section 5.4 and section 5.5.

Note that our system design limits bandwidth consumption since it only needs abstract information to determine whether point cloud pairs can be potentially aligned, and then requests the raw point cloud to estimate the transformation. Specifically, the input data volume of our fusion system to determine the eligibility of point cloud alignment is in the order of kilobytes, which includes the label, center and visible region of each detected objects. Such information can be shared through messages transmitted using periodic V2V communications, e.g., Cooperative Perception Messaging transmitted via DSRC. With varied number of objects detected in the vehicle's view, the raw point cloud of detected objects could be as large as in the order of a few megabytes, which could be transmitted via large-bandwidth mmWave communications when needed.

5.4 Co-visible Objects Detection

Co-visible objects detection module is designed to identify co-visible objects. Specifically, we formulate the problem of detecting co-visible objects as a hyper-graph matching problem [106], which is broadly used in computer vision for key points correspondence determination. Since the single-visible objects are considered outliers in graph matching, we first coarsely filter out the single-visible objects using a threshold-based filtering. Based on the remaining objects in two views, different similarity metrics will be extracted for hyper-graph matching to take advantage of the label and location information obtained from object detection. As hyper-graph matching can pair objects but not distinguish single visible objects, we design a distance consistency check based on hierarchical clustering to identify the correctly matched co-visible objects. The illustration of co-visible objects detection is shown in figure 5.4. After outlier removal, the detected objects are plotted, where different color represents objects detected from different view and different shape indicates different object labels. Hyper-graph matching is able to generate the matching between objects, and the consistency check will further extract the correct matching based on the matched pairs.

5.4.1 Preliminary

Hyper-graph matching was originally proposed to match correspondences between images and can be solved efficiently through reweighted random walk [106]. A hyper-graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A})$ consists of nodes $v \in \mathcal{V}$, hyper-edges $e \in \mathcal{E}$ as well as the attributes $a \in \mathcal{A}$ associated with the hyper-edges. A hyper-edge e encloses a subset of nodes with size $\delta(e)$ from $\mathcal{V}$, where $\delta(e)$ denotes the order of each hyper-edge. The goal of hyper graph matching is to establish the mapping between nodes of two graphs $\mathcal{G}^P = (\mathcal{V}^P, \mathcal{E}^P, \mathcal{A}^P)$ and $\mathcal{G}^Q = (\mathcal{V}^Q, \mathcal{E}^Q, \mathcal{A}^Q)$.

Suppose a set of all possible node correspondences $\mathcal{C} = \mathcal{V}^P \times \mathcal{V}^Q$, and k tuples $c_{\omega_1} = (v^P_{p_1}, v^Q_{q_1}), ..., c_{\omega_k} = (v^P_{p_k}, v^Q_{q_k}) \in \mathcal{C}$ among them. For kth order hyper graph matching, the similarities of the k-tuples can be measured by comparing attributes of two kth order hyper-edge $e^P_{p_1,...,p_k}$ and $e^Q_{q_1,...,q_k}$, which means the hyper-edges connecting $v^P_{p_1,...,p_k}$ and $v^Q_{q_1,...,q_k}$ respectively. Denoting the kth order similarity function by Ω, the kth order similarity of

the k-tuple can be measured by $\Omega(a^P_{p_1,\ldots,p_k}, a^Q_{q_1,\ldots,q_k})$. Therefore, the affinity tensor including kth order similarities can be generalized in a recursive manner as follows:

$$\begin{gathered}\mathbf{H}^{(k)}_{\omega_1,\ldots,\omega_k} = \mathbf{\Omega}_k(a^P_{p_1,\ldots,p_k}, a^Q_{q_1,\ldots,q_k}) + \lambda^{(k-1)} \sum\nolimits_{l=1}^{k} \mathbf{H}^{(k-1)}_{\omega_1,\ldots,\omega_k\omega_l} \\ \mathbf{H}^{(1)}_{\omega_i} = \mathbf{\Omega}(a^P_{p_i}, a^Q_{q_i})\end{gathered} \tag{1}$$

where $\lambda^{(k)}$ represents the weighting factor of kth order similarity value and the superscript on $\mathbf{H}$ denotes the dimension of a tensor. Therefore, the object function of the hyper-graph matching can be formulated to equation 2, where $\mathbf{X}$ is a binary assignment matrix, m^P and n^Q denote the number of nodes in $\mathcal{G}^P$ and $\mathcal{G}^Q$, $\mathbf{1}^P_m$ and $\mathbf{1}^Q_n$ represent all-ones vector with size m and n respectively. By maximizing the matching score of objective function under the one-to-one constraints, the hyper-graph matching problem can be solved by the Hungarian method [107] to find the assignment matrix $\mathbf{X}^*$.

$$\begin{gathered}\mathbf{X}^* = \underset{\mathbf{X}}{\operatorname{argmax}}\, \mathbf{H}(k) \otimes \mathbf{X} \\ \text{s.t. } \mathbf{X}\mathbf{1}_{n^Q \times 1} \leq \mathbf{1}_{m^P \times 1},\ \mathbf{X}^\mathbf{T}\mathbf{1}_{m^P \times 1} \leq \mathbf{1}_{n^Q \times 1}\end{gathered} \tag{2}$$

5.4.2 Matching Outlier Removal

To improve the object matching accuracy, our proposed system first removes matching outliers. In the hyper-graph matching task, matching outliers refer to the nodes which only consist in one graph and can not be matched. In our vehicle view matching context, matching outliers are the single-visible objects. As the increasing of overlapping region between vehicle's view, there will be more single-visible objects involved each view. Therefore, removing such single-visible objects will reduce the number of outliers and increase object matching accuracy. At current stage, the single-visible can be coarsely excluded based on nearest neighbour search. As all the objects from two views are transformed into the same world coordinates, if a object does not have any neighbours in the other view within a threshold distance, the object can be classified as single-visible object and removed before matching. The distance threshold can be determined based on the localization accuracy, such as the accuracy value provided by Android Location API [108], which indicates horizontal accuracy in meters as the radius of 68% confidence.

5.4.3 Hyper-graph Matching with Multiple Similarity Measures

Inspired by the existing work [106, 109, 110], our work extends the hyper-graph matching by combing multiple similarity measures, including attribute-based similarity measures, geometry-based similarity measures, etc., in matching.

Each vehicle can first build a hyper-graph based on its locally detected objects, and additionally, it can also build a hyper-graph for a remote vehicle based on the shared information from that vehicle. In the built hyper-graphs, nodes denote detected objects by the observing vehicle and edges represent the spatial relationship between detected objects. The attributes of each node, such as the category the object belongs to, the size of the object, etc., can be utilized to distinct one node from others. In our work, we focus on exploiting the category information of objects since it is invariant under different viewpoints.

$$\mathbf{H}^{(1)}_{\omega_1,\omega_2,\omega_3} = exp\Big[- \frac{1}{\sigma_{s^1}} \sum_{k=1}^{3} \mid \sin(\theta^P_{\omega_k}) - \sin(\theta^Q_{\omega_k}) \mid \Big] \tag{3}$$

$$\mathbf{H}^{(2)}_{\omega_1,\omega_2,\omega_3} = exp\Big[- \frac{1}{\sigma_{s^2}} \sum_{\substack{i,j\in\{1,2,3\} \\ i\neq j}} \mid d^P_{\omega_i\omega_j} - d^Q_{\omega_i\omega_j} \mid^2 \Big] \tag{4}$$

$$\mathbf{H}^{(3)}_{\omega_1,\omega_2,\omega_3} = \frac{1}{3} \sum_{k=1}^{3} \text{diff}(l^P_{\omega_k} - l^Q_{\omega_k})^{\sigma_{s^3}} \tag{5}$$

In order to qualify the hyper graph similarity, we specifically extract the angle, distance and label similarities based on hyper edges. The angle similarity [111] is defined in equation 3 based on a pair of 3rd order hyper-edges, $e^P_{a,b,c} \in \mathcal{E}^P$ and $e^Q_{x,y,z} \in \mathcal{E}^Q$, ($a, b, c \in \mathcal{V}^{\mathcal{P}}$ $a \neq b \neq c$ and $x, y, z \in \mathcal{V}^{\mathcal{Q}}$ $x \neq y \neq z$), where $\theta^P_{\omega_k}$ and $\theta^Q_{\omega_k}$ denote the angles in the triangle pairs formed by the correspondence ω_k in $\mathcal{P}$ and $\mathcal{Q}$. The distance similarity [106] is quantified based on equation 4, where $d^P_{\omega_i\omega_j}$ and $d^Q_{\omega_i\omega_j}$ represent the length of edges formed by the nodes within the hyper-edge. σ_{s^1} and σ_{s^2} are scale factors, which are set empirically to 0.5 and 0.15 as in [106].

To take advantage the label information predicted by vehicle object detector, we define the label similarity $\mathbf{H}^{(3)}_{\omega_1,\omega_2,\omega_3}$ in equation 5, where $l^P_{\omega_k}$ and $l^Q_{\omega_k}$ are the labels of the correspondence. diff function is designed to output value ranging from 0 to 1, which 1 indicates

labels of the correspondence are completely the same, and 0 means completely different. Depending on the representation of the shared label from vehicles, diff function can be implemented in various ways. If only the final predicted label of each object is available, then diff function can be implemented as piecewise function, where same labels outputs 1 and different label outputs 0. If the predicted confidence vector across all categories are available, then diff function can be computed based on the cross-entropy of the two confidence vectors. The scale factor σ_{s^3} is set to 3 empirically in our implementation. Although the distance and label similarity can be implemented based on 2rd order and 1st order edge respectively, we define them based on 3rd order here for easier probability combination.

$$\mathbf{H}_{\omega_1,\omega_2,\omega_3} = \left[\lambda^{(1)}\mathbf{H}^{(1)} + \lambda^{(2)}\mathbf{H}^{(2)}\right]\mathbf{H}^{(3)} \tag{6}$$

To merge the three similarity metrics, we combine them as defined in equation 6, where the subscript of $\mathbf{H}^{(1)}$ $\mathbf{H}^{(2)}$ $\mathbf{H}^{(3)}$ are omitted as they share the same subscript as defined in equation 3,4,5. Instead of linearly adding all the metrics as generalized in [106], we propose to use the label similarity as a conditional probability. It is because not only the correct correspondence in label similarity can generate higher values, the incorrect correspondence happened to have same labels will also produce higher value. Therefore, linearly combining the label similarity will actually increase the overall similarity score for incorrect matching, which yields lower matching accuracy. Using the label similarity as a conditional probability can benefit the hyper-graph matching because the overall similarity score will only be higher if the correspondence have matched labels. $\lambda^{(1)}$ and $\lambda^{(2)}$ are the weights for linear combining the angle and edge length similarity, and we set both to 0.5 for equal weights assignment in our implementation.

5.4.4 Hierarchical Clustering based Consistency Check

Since hyper-graph matching only assigns matched objects between views but can not distinguish co-visible objects from single-visible objects, we propose a hierarchical clustering based consistency check to extract co-visible objects from hyper-graph matching results.

For detected objects in two observations, the distance between a pair of matched objects

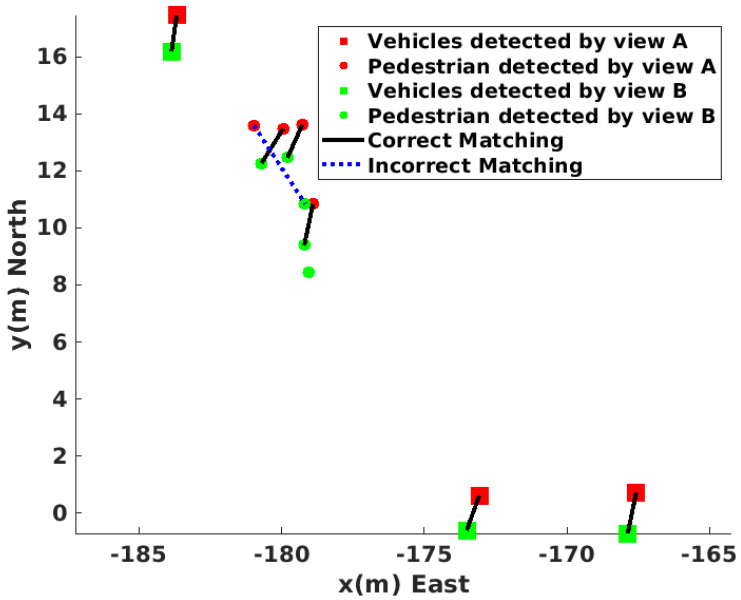


Figure 5.4: Object matching illustration

can be computed according to the euclidean distance between the centers of objects in the world coordinate system. If the matching is correct, then the distance between the pair of objects consists two parts: (i) the localization error and (ii) the error from inaccurate 3D object detection. As the state-of-the-art algorithms can archive 81.43% accuracy for vehicle 3D detection based on 0.7 IoU threshold [112] but the localization error is still as high as meter-level in the urban area, it is reasonable to infer the localization error is the major component of the distance between matched pairs. Since the objects detected by the same vehicle shares the same localization error, the pairwise distance between the correct matched objects should share such same component in distance, which is the combination of localization error of two vehicles. But the distance between incorrect matched objects maybe largely diverse. Although the direction of matched pairs have similar characteristics, it is not resilient to the object detection error as the distance between matched pairs. Small errors in object center location may cause large direction error of matched co-visible object pairs.

Based on the analysis that the pairwise distance between correct matched objects should be relative consistent, clustering method can be applied on graph matching results to extract the correct matched co-visible objects. Specifically, we perform the hierarchical clustering on

the hyper graph matching outputs, and classify the objects cluster with consistent pairwise distance as co-visible objects and the others as single-visible objects. The threshold distance variance in hierarchical clustering to select the cluster can be determined based on the 3D object detection performance, because it mainly comes from the 3D object detection error. In order to increase the precision of co-visible objects detection, the number of co-visible objects, namely the number of objects within the selected cluster, should be at least three to produce the final output. Otherwise, the system will ignore the information and does not perform fusion based on received data. Such design will make sure the system only fuse the information when it has enough confidence to do so.

5.5 Co-Visible Region Refinement

Although the module of co-visible objects detection can identify single-visible objects and match co-visible objects, it remains challenging to perform point cloud registration accurately. It is because that the matched co-visible objects may not have enough common seen area due to the large view-difference and occlusions. For example, in figure 5.5, objects are observed by two different viewpoints and the resulting point clouds are colored by red and green, respectively. Although the two point clouds in figure 5.5(d) refer to the same co-visible object, no overlapping area exists between them. On the contrary, the two points in figure 5.5(a) shows a clear overlapping area of the a co-visible object. As discussed in section 5.2, the overlapping region between the two point clouds is essential for correct point clouds alignment. In order to address the challenge of lack of overlapping area, the Co-Visible Region Refinement is designed to first, among all detected co-visible objects, quantify the overlap region of the co-visible objects from two vehicle's views and then align point cloud based on cropped co-visible point clouds.

5.5.1 Object Visible Region Estimation

In order to identify the overlap region of co-visible objects, the visible region of co-visible objects needs to estimate based on each vehicle's viewpoint. To address such challenge, we propose to quantify the object visible region approximately from the bird's eye view based

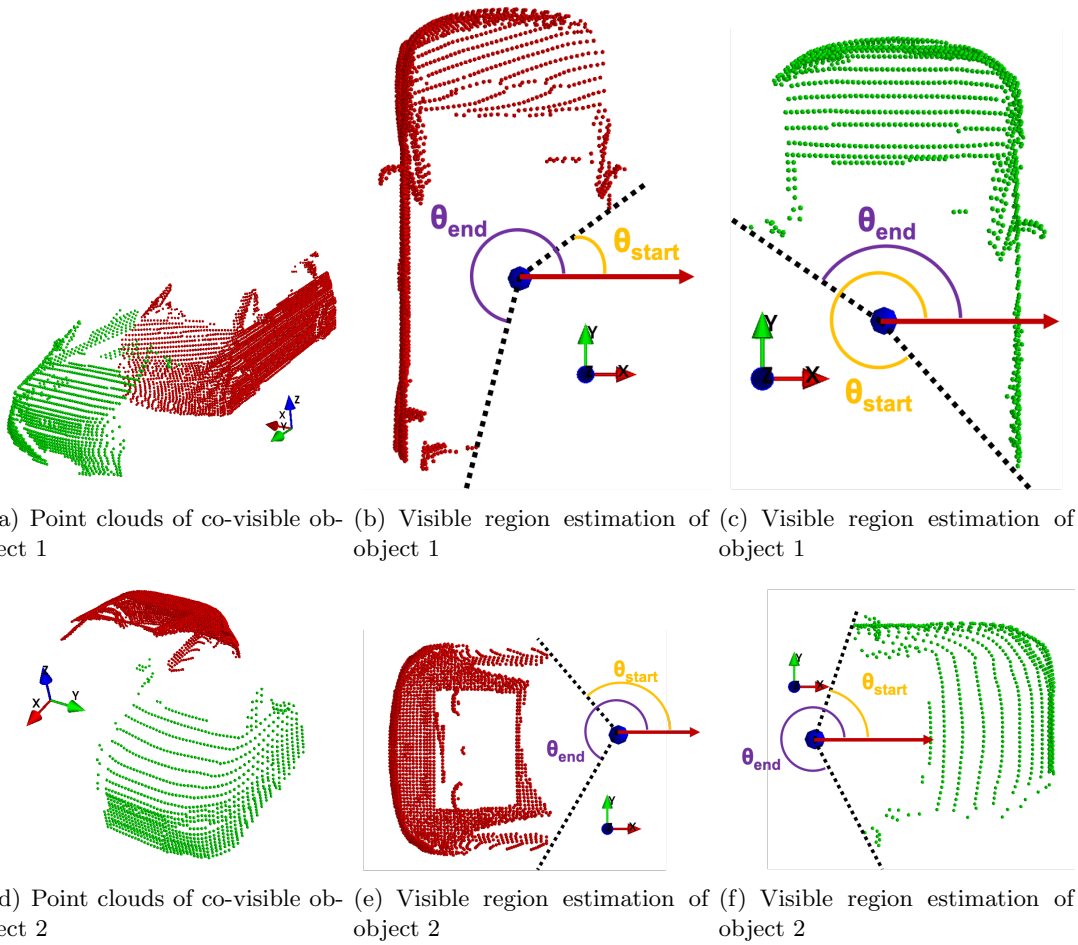
on the relative location between detected object center and corresponding point cloud of the object. As point clouds are generated by sensors with depth sensing capability, e.g. stereo cameras and LIDARs, such depth sensing capability is compliant with the line-of-sight rule, which can not see through objects and only sense line of sight object regions. Therefore, the point cloud generated by these sensors must locates between the detected object's center and the observer vehicle. Inspired by these characteristics, we approximate the object visible region by estimating the angle of the point cloud's coverage region with respect to the object's center. By projecting the point clouds to a bird's eye view, the coverage region of a point cloud can be represented as $\theta^i = [\theta^i_{start}, \theta^i_{end}]$, where θ^i_{start} and θ^i_{end} are the starting and ending angle of the point cloud coverage area for object i. Both θ^i_{start} and θ^i_{end} are computed according to the same axis such as west to east. Thus, the point cloud coverage can be quantified based on the counter-clockwise circular angle difference from θ^i_{start} to θ^i_{end}. The demonstration of visible region estimation are shown in figure 5.5(b)5.5(c)5.5(e)5.5(f).

The points which are closer to object center or reflected by vehicle roof will be noisy to estimate object visible region. Thus, a prepossessing can be applied to improve the robustness by filtering out such as points based on distance threshold or surface detection.

5.5.2 Co-visible Object Selection

Although co-visible objects can be identified based on object matching, there is no guarantee that the observation of co-visible objects from each vehicle's view will have overlapping area. Given the fact that, the state-of-the-art point cloud registration methods require the overlapping area between point clouds to estimate the transformation. The co-visible objects which include no or little overlapping area is generally not suitable for point cloud registration.

Based on this observation, we propose to measure the intersection area based on the overlapping of point cloud coverage angle which can be defined as $\mid \text{intersect}(\theta^i, \theta^j) \mid$. In general, the intersection between two point cloud coverage angle indicates the overlapping area between two point clouds. For example, the estimated object visible region shown in figure 5.5(e) and 5.5(f) don't have any overlapping area since the intersection of them is zero. However, the visible region shown in figure 5.5(b) and 5.5(c) shows the intersection



(a) Point clouds of co-visible object 1

(b) Visible region estimation of object 1

(c) Visible region estimation of object 1

(d) Point clouds of co-visible object 2

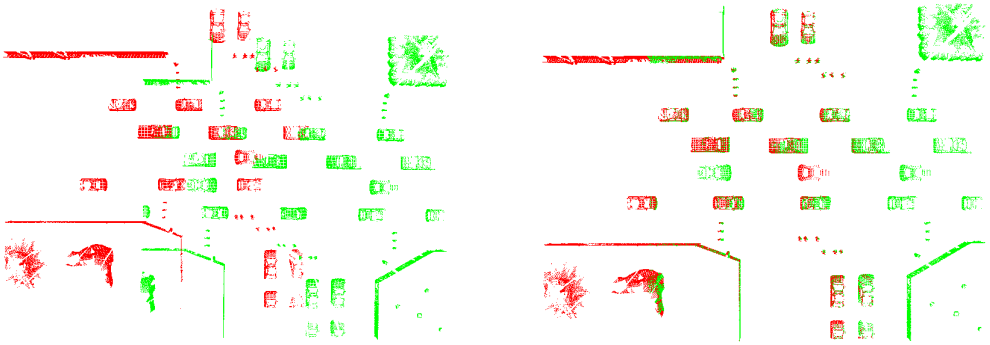
(e) Visible region estimation of object 2

(f) Visible region estimation of object 2

Figure 5.5: Object Visible Region Estimation.

angle between two point clouds is around 120 degree, which can be potentially used for point cloud registration.

Generally, larger overlapping area between point clouds will have better registration performance. In order to improve to the point cloud registration accuracy, we propose to examine the visible region of each pair of detected co-visible objects, and only keep the point clouds for the pairs whose visible region is larger than a threshold. If there are more than one of such pair is found, the system will further to crop and align the point clouds. However, the system will reject to align the point cloud pair if the intersection visible region of all co-visible objects is smaller the threshold and only align the point cloud based on minimizing the distance between co-visible object centers. In our implementation, the threshold is set to 30°as it is commonly required for state-of-art point cloud registration



(a) ICP alignment result of sample 0 view angle difference (mean error = 6.92m)

(b) Our alignment result of sample 0 view angle difference(mean error = 0.03m)

Figure 5.6: Sample alignment results with 0 view angle difference in Carla simulation

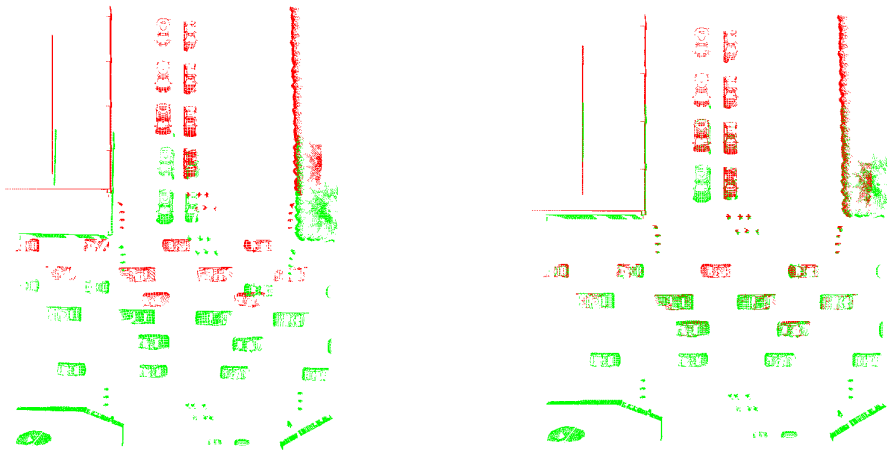
algorithms.

5.5.3 Cropped Point Cloud Alignment

Based on the selected co-visible objects, point cloud registration can be applied to align two point clouds. In order to improve the robustness of point cloud registration, we propose to crop the point cloud based on the intersection of visible regions. Specifically, only the points within the intersection region intersect(θ^i, θ^j) will be used for point cloud registration. Such process will remove outliers and increase the inlier ratio for point cloud registration. Eventually, the transformation estimated based on the selected co-visible object will be applied to the whole point cloud captured by the sharing vehicle. General point cloud registration algorithms can be used here for transformation estimation, such as ICP.

5.6 Experiment Setup and Implementation

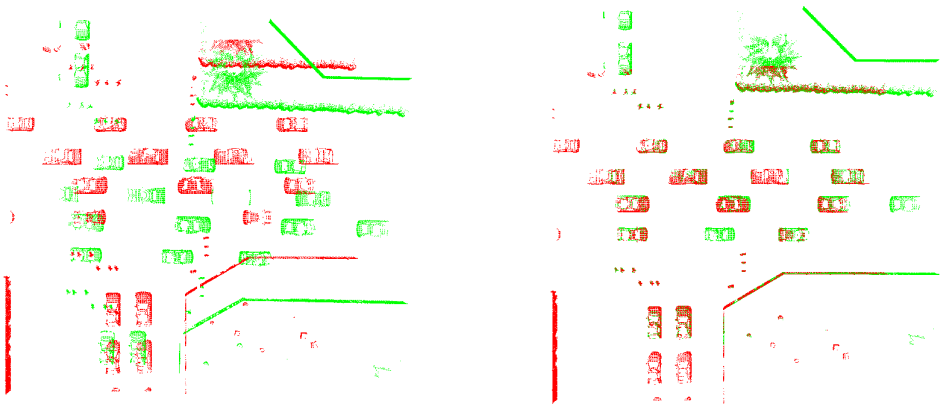
In order to explore the system performance, we evaluate our system using experimental data from the KITTI [75, 76] dataset and synthetic data generated by the CARLA [48] simulator respectively, and also implemented 3D object detection and three baseline algorithms.



(a) ICP alignment result of sample 90 view angle difference(mean error = 5.54m)

(b) Our alignment result of sample 90 view angle difference(mean error = 0.02m)

Figure 5.7: Sample alignment results with 90 view angle difference in Carla simulation



(a) ICP alignment result of sample 180 view angle difference(mean error = 4.76m

(b) Our alignment result of sample 180 view angle difference(mean error = 0.05m)

Figure 5.8: Sample alignment results with 180 view angle difference in Carla simulation

5.6.1 KITTI based experimental data

The KITTI [75, 76] dataset includes detailed information of a single autonomous vehicle travelling through a wide range of road scenarios. It contains a trove of sensor readings from a variety of sensor modalities such as high resolution color and grayscale stereo cameras, a Velodyne 3D laser scanner and a GPS/IMU inertial navigation system. For this work,

however, the KITTI dataset is not directly applicable since we need two sets of point clouds captured at different perspectives of the same scene.

We address this limitation by leveraging Lidar point clouds obtained at different timestamps of the same route from the vehicle, to imitate two cars travelling by following each other. (There are no eligible cases found to imitate cars facing each other.) More specifically, we examined KITTI's 3D object detection dataset [75] and identified a plethora of time-instance pairs where more than three same street object are detected at both scenes and vehicles are traveled more than 4 meters based on GPS. We removed pairs where the transformation between object pairs are not consistent to filter out inaccurate ground truth labeling and moving street objects. This process ended up generating 668 pairs of different time instances that satisfied the requirement of our fusion pipeline, which includes at least 3 pairs of co-visible objects and at least one of co-visible objects has more than 30°overlapping region. As our system built based on the object detection results, which can only be performed on the front view camera, the input point clouds in this experiment are limited to the LIDAR points in the front view.

5.6.2 CARLA based synthetic data

As there is no labeled large view angle difference dataset available, we use CARLA [48] to render realistic intersection scenarios, which provides open digital assets (urban layouts, buildings, vehicles) and supports flexible specification of sensor suites. The intersection scenario is rendered in Town 5 of CALRA 0.9.5 builtin map, includes 4 directions and each direction with two lanes. Each lane has 5 vehicles, which are set to be the same model to avoid rear-ended collisions, since vehicles are controlled based on throttle in CALRA 0.9.5 and different models may have different acceleration based on same throttle value. But the outlook color of each set of 5 vehicles are different and randomly picked. In addition, 6 pedestrians are considered on 4 corners of the intersection in groups of three, which are standing in line and trying to cross the intersection if there is not conflicting traffic. Overall, the intersection scenario includes 40 vehicles and 24 pedestrians in total.

For each vehicle, 4 cameras are mounted on the center top of vehicles' roofs with the height as 2m from the ground to cover the full 360°field of view. In order to generate dense

point clouds, we use 'depth camera' in CARLA to obtain the pixel depth, which is the ground-truth pixel range perfectly aligned with the corresponding RGB image. Besides, the position and bounding box of vehicles, and the pose transform of cameras are also logged. In order to simulate the unstable GPS reading, we randomly sample localization error from a Gaussian Distribution $X \sim \mathcal{N}(0, 1)$ with zero mean and 1 meter standard deviation. Among the process of vehicles and pedestrians completed cross the intersection (positioned on the other side of intersection), we evenly take 16 snapshots with 1 second interval. Applying the same filtering of KITTI dataset, 3228 pairs of vehicle observations meets our fusion system requirement and are extracted for evaluation.

5.6.3 3D object detection implementation

To obtain 3D object detection results, we implemented two detectors on both datasets, respectively. For the KITTI dataset, we reproduce the 3D object detection workflow proposed in [112] to obtain the results. As there is no pre-trined 3D object detection model available for Carla synthetic data, we implemented a 2D-driven detector inspired by [112] to intimate the state of art 3D object detection performance based on ground-truth. Specifically, 2D object detection is performed on RGB images, and the detected 2D bounding box will be projected into 3D space based on depth. If a projected 2D bounding box intersects with the ground-truth bounding box, then the corresponding ground-truth bounding box will be used by adding a 3D noise vector which sampled from the uniform distribution within range [-0.2m, 0.2m]. Note that the precision and recall of such 3D object detector still depends on the 2D detection performance, where we use the pre-trained SSD-ResNet50 model provided Tensorflow Object Detection API [113].

5.6.4 Baseline algorithms implementation

As introduce in the section 2.1, we implemented 3 baseline algorithms to serve as the benchmark for point cloud performance comparison. Specifically, (1) ICP [73] is implemented based on MATLAB pcregistericp function. (2) the deep learning feature descriptor

	KITTI			CARLA		
	Recall	Mean	STD	Recall	Mean	STD
ICP	8.53	10.22	9.26	5.79	10.56	10.47
FCGF	72.75	8.76	8.34	33.58	9.57	8.88
SSM	9.88	9.72	8.87	35.29	8.79	8.03
Ours+ICP	**86.68**	**7.84**	**7.20**	**75.71**	**7.66**	**6.40**
Ours+FCGF	84.73	9.14	8.75	65.49	10.12	9.54

Table 5.1: Point cloud registration accuracy on KITTI and CARLA dataset.

FCGF [85] [2] is extracted based on the pre-trained model on KITTI dataset and combines with RANSAC as a state-of-art correspondence-based baseline. (3) The closest work to ours, SSM [98] is also implemented in MATLAB. Note that, SSM [98] uses ICP to align the point clouds after its object matching.

5.7 Evaluation Results

In this section, we evaluate the proposed method in terms of (i) the achievable accuracy of point cloud registration; (ii) the accuracy of object matching in the co-visible objects detection; (iii) the benefit eligibility of our method under different system settings.

5.7.1 Point cloud registration accuracy

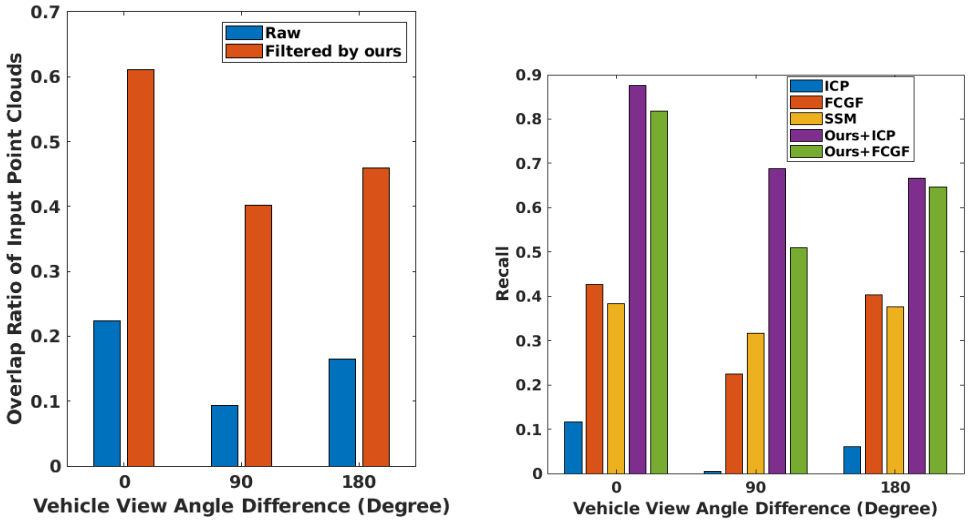
Following the evaluation setup in [71–74], we select recall, mean error and error standard deviation (STD), as the primary evaluation metrics. Specifically, the recall is computed based on the average distance between the ground-truth alignment and the estimated alignment. If the average distance is less than a pre-defined threshold (0.2 m in our evaluation), the estimated alignment is considered a true positive. The recall is the ratio of the number of true positives to the number of input point cloud pairs. The mean error and the STD are then computed for the true positive pairs.

In Table 5.1, we show the performance of three baseline point cloud registration algorithms, *ICP*, *FCGF*, *SSM* as well as two enhanced variants by using our method (denoted by *Ours + ICP* and *Ours + FCGF*. (The unit of recall is percentage, and it of mean error and STD is cm.) Across all metrics, Ours + ICP outperforms all other solutions in both

[2] Although the following work [74] shows better performance on point cloud registration, but is trained based on 360oV LiDAR of KITTI, which is not considered as a fair comparison.

CARLA and KITTI dataset by a significant margin. Comparing to the vanilla ICP and FCGF, our method can boost the alignment accuracy by 900% and 16%, respectively. Such a large gain is due to our method can extract co-visible regions between the two input point clouds and thus largely reduce the ambiguity in the point cloud registration. Note that, the performance of method Ours+FCGF can be potentially further improved if the feature extractor of FCGF is re-trained on the extracted point clouds using our method. SSM does not work well on both datasets, as it only crops point cloud based on the object matching but not further refine co-visible regions. As KITTI dataset involves large localization errors, the object matching in SSM, which largely relies on pairwise distance between objects, fails. Our method, on the other hand, is much more robust to the large localization errors, as our method takes advantage of multiple similarity measures between observed objects and these similarity measures are resilient to the localization errors. Except SSM, other methods show a lower accuracy on CARLA compared to KITTI dataset, since the scenes in CARLA dataset are more complex as they include more surrounding objects and vehicles with diverse view-angles.

We also evaluate the performance of our method with respect to different view-angles in CARLA dataset. As it is generated for a four-leg intersection and the vehicles in the scenario are either stopped or driving in straight along the road, the view-angle difference between any two vehicles in the scenario is 0°, 90°, or 180°. To quantify the input point clouds overlapping region over three view-angle differences, we define the overlap ratio as the number of overlapped voxels over all voxels when downsampling point clouds with 0.1m. Figure 5.9 shows the overlap ratio of input point clouds and the recall of point cloud registration for 0°, 90°, and 180°view-angle difference, respectively. Compared to the raw inputs which are the points of within all the detected object bounding boxes, the filtered point clouds using our method yield a much higher overlap ratio for different view-angle difference. The results indicate that the performance of the methods highly correlates with the overlap ratio of the input point clouds. For example, the performance of ICP, FCGF, SSM in the 90°view-angle difference case, with the raw point cloud as input yield the lowest recall, while Ours+ICP and Ours+FCGF taking the filtered point clouds as input show the highest accuracy. In summary, the results show that our method can increase overlap ratio



(a) Point cloud overlap ratio across different vehicle view angle difference

(b) Point cloud registration recall across different vehicle view angle difference

Figure 5.9: Point cloud registration performance across different vehicle view angle difference.

in the input point clouds and thus improves point cloud registration accuracy significantly compared with baselines.

Additionally, we also qualitatively compare the point cloud registration results in sample test cases across different view angle difference between ICP and our methods in figure 5.6,5.7 and 5.8. Figure 5.6(a) and Figure 5.6(b) show the comparison when two vehicles are closely following each other with the same view angle. Even though such case pairs include overlapping area, there are still large portion of single-visible objects involved, which makes the ICP fails to align the point cloud correctly. Figure 5.7(a) and Figure 5.8(a) demonstrate the results of ICP in 90°and 180°vehicle view angle difference respectively, where the alignment is not performed accurately due to low overlapping. However, our two phase point cloud registration method can can align the point clouds accurately as show in Figure 5.7(b) and Figure 5.8(b).

5.7.2 Object matching accuracy

Since the two phase design of our system takes the output of object matching to perform co-visible region refinement, we evaluate the co-visible objects detection individually in

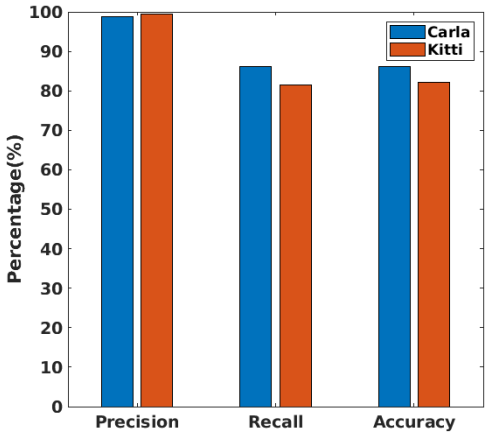


Figure 5.10: Object matching precision, recall and accuracy.

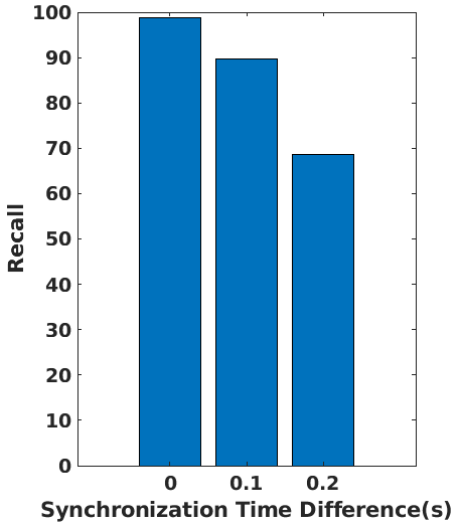


Figure 5.11: Recall across different synchronization time difference

terms of precision, recall and accuracy. Note that the metrics are only calculated when our co-visible objects detection can produce a result, i.e., when there are at least three pairs of objects are kept after consistency check. Specifically, the precision and recall are defined as the number of correct co-visible matching over the number of all detected co-visible objects and the ground-truth number of co-visible objects, respectively. But the correct detection in the accuracy evaluation requires to not only match the co-visible objects, but also identify the single-visible objects correctly. As shown in Figure 5.10, our method can achieve 98.76% and 99.43% precision, 86.14% and 81.46% recall, 86.22% and 82.21% accuracy for the KITTI and the CARLA dataset, respectively. The high precision of our system guarantees that co-visible objects can be identified and matched accurately and thus guarantees the correctness of the input to the following co-visible region refinement and the final point cloud registration. The high recall of our system makes sure that the most co-visible objects are extracted for next phase.

5.7.3 Benefit Eligibility

In order to push our system to real deployment, we study the system benefit eligibility in this section, including: (1) how likely does a vehicle can find a peer to share and align point

Field of View	Data sharing volume Per Vehicle		Median Point Fusion Benefit Ratio	Recall
	Abstract Info	Point Cloud		
90	1.69KB	0.42MB	51.88%	85.16%
360	4.98KB	1.53MB	99.23%	98.81%

Table 5.2: Data sharing volume and system performance across different field of view.

cloud based on our system (2) what is network requirement and performance trade-off for our system between different vehicle settings (3) how does the point cloud synchronization affect the system performance.

Since our system requires to discover at least three pairs of co-visible objects to fuse the point cloud, we explored how likely these cases will occur in our CARLA intersection simulation. If a vehicle can fuse the data from at least one neighboring vehicle, it can potentially benefit from our system. In order to quantify such benefits, we define **Benefit Ratio**, which is the ratio of the number of vehicles which can gain benefit at current timestamp over the number of all vehicles. The benefit ratio is calculated for each of the 16 snapshots in our simulation. We perform the experiments based on two different field of views(FoV), i.e., 360°FoV and 90°FoV. Figure 5.12 shows the empirical cumulative distribution of benefit ratios. Since a larger field of view can increase the overlapping sensing area between vehicles, the benefit ratio of 360°FoV are generally higher than that of 90°for both co-visible object detection and co-visible region refinement. Even though some cases include more than 3 pairs of co-visible objects, the overlapping area of the co-visible objects are still too small to perform the co-visible region refinement. Therefore, the co-visible region refinement of both FoVs are lower than the co-visible object detection. The variation of the benefit ratios depends on the location distribution of the objects. The benefit ratio of the co-visible region refinement with 90°FoV is greater than 0.7 for some snapshots. It is because the crossing vehicles of these snapshots are close to the center of the intersection and make themselves to be identified as co-visible objects for others. Among the cases where vehicles can perform the co-visible region refinement, we also check the number of neighboring vehicles whose information can be potentially fused. We observe that the median number of the candidate neighbours for the co-visible region refinement is 2 and 14 for the 90°FoV case and the 360°FoV case, respectively.

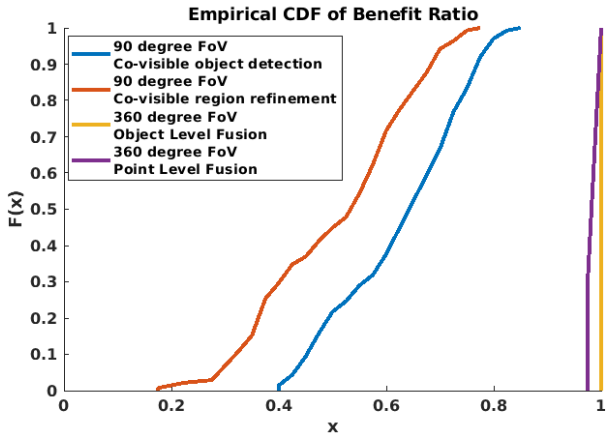


Figure 5.12: Cumulative distribution of benefit ratio

We explore the network requirements and system performance across different field of views, as shown in table 5.2. The volume of data to be shared is calculated based on the size of required information by each vehicle to perform the fusion. The recall in this subsection is evaluated based on the same definition as in section 7.1, but with a loose threshold 1m. Specifically, the volume of data required by the system to determine whether it is eligible to perform the fusion is the size of abtract information including detected object center label and point cloud visible region, and the point cloud data sharing volume is considered as the size of whole point cloud which is arguably to enable various applications. 90°FoV vehicle to vehicle information fusion requires to share 1.69KB for the abstract information and 0.42MB for the whole point cloud. 360°FoV needs more shared information, with 4.98KB and 1.53MB for the two steps respectively, but also consequently increases the benefit ratio and decreases the mean errors of point cloud registration.

Additionally, we also evaluate the system performance with different synchronization time difference. Although the point cloud can be shared with a timestamp, the synchronization between point cloud pairs may not be perfect due to hardware clock bias, sensor sample frequency, etc. Thus, the point cloud registration recall is evaluated according to different synchronization error by selecting input point clouds with different timestamps. As shown in figure 5.11, the point cloud registration accuracy decreases as more synchronization difference introduced. Given the relatively large synchronization error in 100ms and 200ms, the system can still align the point cloud with 89.66% and 68.55% recall respectively.

5.8 Conclusion and Discussion

In order to overcome the limitations of state-of-art point cloud registration algorithms and enable the point cloud fusion across connected vehicles, a two-phase point cloud fusion system is proposed. The system first identifies and matches co-visible objects using hypergraph matching based on the extracted location and label information. It then estimates the co-visible region for each of them and crops out the larger overlap region. The selected co-visible area acts as an anchor point and its point cloud will be used to estimate the transformation. We evaluate the accuracy of point cloud registration and co-visible matching based on both real-world KITTI [75, 76] dataset and synthetic CARLA [48] dataset, and shows it can achieve 86.68% and 75.71% recall with a 0.2m mean point-wise error threshold.

We believe that the system performance can be further improved if more objects are detected and considered during object matching. Our existing implementation focuses on detecting moving objects in the scene but static objects such as traffic lights and light poles could increase the probability of discovering 3 co-visible objects across two vehicle views and thereby improve the benefit ratio of surrounding vehicles.

Chapter 6

Conclusion

In this dissertation, we presented novel techniques and system designs to scale data sharing among vehicles to tackle long tail traffic situations. Compared with existing research which mostly focus on heavy instrumented vehicles for data collection and human labeled unusual scenes, we take advantage of the widely available inertial and camera sensing data to construct the low cost systems, which can be easily deployed at a larger scale. Inspired by recent progress on inertial sensing and deep neural networks, our design of the sensing system can accurately model driver's behaviors and detect uncommon driving views. The minimum infrastructure and accurate sensing design of our proposed system can also enable any dashcam-equipped vehicle to take advantage of the billions of miles that humans already drive, which will accelerate the long tail traffic events collection. Besides, this thesis provides a point cloud fusion system which not only improves the point cloud registration accuracy between two vehicles, but also illustrates the potential benefits of existing V2V network data sharing. We believe these systems can ease the data sharing between vehicles to collect, identify and tackle the long tail traffic situations, which potentially helps to build more robust automated driving systems.

In summary, this thesis details the following contributions:

- We present BigRoad, a light-weight sensing and driving data logging system that can derive internal driver inputs (i.e., steering wheel angles, driving speed and acceleration) and external perceptions of road environments (i.e., road conditions and front-view video) using very few off-the-shelf sensing devices (i.e., a smartphone and an IMU) in a vehicle. The low-cost design of BigRoad can collect fine-grained driving data to support developing dependable automated driving and traffic safety technologies. Evaluation based on over 140 real-driving trips shows that BigRoad can generate

accurate and robust driving data from different vehicle models and drivers.

- We propose an automatic unusual driving events identification system to allow scaling the collection of driving data and corner cases to a much larger fleet of human-driven vehicles without requiring upload or human review of all data. The proposed system is able to capture various unusual circumstances, including hazardous event like sudden braking and swerving events through a three-stage process involving inertial sensing and detecting outliers of current trained self-driving systems based on auto-encoder deep neural network. The evaluation is based on more than 120 hours of real road driving data and shows that it outperforms baseline methods on unusual event with 82% accuracy improvement over baseline on sudden reaction detection and above 71% accuracy on unfamiliar views identification. While the dataset size allowed validating the techniques only with less rare events, one might expect that similar patterns hold also in the much rarer events that would eventually be of interest.

- A two-phase point cloud fusion system is proposed to overcome the limitations of state-of-art point cloud registration algorithms and enable the point cloud fusion across connected vehicles. The system is able to identify and match co-visible objects using hyper-graph matching, and then estimate the co-visible region for each of them and crops out the larger overlap region. Such two-phase design can not only improve the overlapping ratio between the point clouds from two vehicles, but also meet the network requirement for connected vehicles. We evaluate the accuracy of point cloud registration and co-visible matching based on both real-world KITTI dataset and synthetic CARLA dataset, and shows it can achieve 86.68% and 75.71% recall with a 0.2m mean point-wise error threshold.

6.0.1 Future directions

Scaling data sharing among vehicles can help to extend the coverage of the long tail traffic situations, but it is still difficult to explore and understand the further end of the long tail. To increase the long tail coverage and improve the system robustness, there are several directions to build on this thesis work:

- With the increasing computational power and sensing capability on mobile devices, it is interesting to explore the possibility for road information acquisition with less device requirement such as using only smartphone by taking advantage of the front and multiple rear cameras and LiDARs.

- When driving events are recorded, drivers' privacy may be compromised by the imagery and inertial sensor readings. It would be interesting to explore the system implementation in privacy conservative manner, such as building the deep learning framework based on Federated Learning [114] to mitigate the privacy concern by keeping the raw measurements on devices.

- With the deployment of the system, there may be differences of sample frequency, sensitivity, and resolution in various hardware sensors, such as inertial sensors, cameras and LiDARs. Additional modules to handle such heterogeneity can also be discussed.

References

[1] "Waypoint - the official waymo blog: Introducing the 5th-generation waymo driver: Informed by experience, designed for scale, engineered to tackle more environments," Mar 2020.

[2] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[3] M. Herger, "Waymo reveals more details on new sensor hardware that could indicate imminent mass production," Mar 2020.

[4] S. W. Kim, B. Qin, Z. J. Chong, X. Shen, W. Liu, M. H. Ang, E. Frazzoli, and D. Rus, "Multivehicle cooperative driving using cooperative perception: Design and experimental validation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, pp. 663–680, April 2015.

[5] J. B. Kenney, "Dedicated Short-Range Communications (DSRC) Standards in the United States," *Proceedings of the IEEE*, vol. 99, pp. 1162–1182, July 2011.

[6] F. Ahmed-Zaid, F. Bai, S. Bai, C. Basnayake, B. Bellur, S. Brovold, G. Brown, L. Caminiti, D. Cunningham, and e. a. H. Elzein, "Vehicle Safety Communications-applications(VSC-A) Final Report," tech. rep., National Highway Traffic Safety Administration (NHTSA), 2011.

[7] "Vehicle Safety Communications Project: Task 3 Final Report: Identify Intelligent Vehicle Safety Applications Enabled by DSRC," tech. rep., National Highway Traffic Safety Administration (NHTSA), 2005.

[8] F. Bai, D. D. Stancil, and H. Krishnan, "Toward Understanding Characteristics of Dedicated Short Range Communications (DSRC) from a Perspective of Vehicular Network Engineers," in *Proceedings of the Sixteenth Annual International Conference on Mobile Computing and Networking*, MobiCom '10, (New York, NY, USA), pp. 329–340, ACM, 2010.

[9] T. Rappaport, R. Heath, R. Daniels, and J. Murdock, *Millimeter Wave Wireless Communications*. Prentice Hall Communications Engineering and Emerging Technologies Series from Ted Rappaport, Pearson Education, 2014.

[10] "Ieee standard for information technology–telecommunications and information exchange between systems–local and metropolitan area networks–specific requirements-part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications amendment 3: Enhancements for very high throughput in the 60 ghz band," *IEEE Std 802.11ad-2012 (Amendment to IEEE Std 802.11-2012, as amended by IEEE Std 802.11ae-2012 and IEEE Std 802.11aa-2012)*, pp. 1–628, Dec 2012.

[11] T. Krisher and J. Lowy, "Tesla driver killed in crash while using car's 'autopilot'," *ASSOCIATED PRESS(June 30, 2016)*.

[12] C. Urmson, "Google Self-Driving Car Project." `https://www.transportation.gov/sites/dot.gov/files/docs/AV%20policy%20guidance%20PDF.pdf`. Talk at SXSW Interactive 2016.

[13] National Highway Traffic Safety Administration, "Federal Automated Vehicles Policy." `https://www.transportation.gov/sites/dot.gov/files/docs/AV%20policy%20guidance%20PDF.pdf`.

[14] National Highway Traffic Safety Administration, "Fatality Analysis Reporting System." `https://www.transportation.gov/sites/dot.gov/files/docs/AV%20policy%20guidance%20PDF.pdf`. Talk at SXSW Interactive 2016.

[15] "Waymo: Google's parent company on self-driving." https://waymo.com/.

[16] K. Korosec, "GM's Cruise Automation Is Testing Self-Driving Chevy Bolts in Arizona." `http://fortune.com/2016/08/09/cruise-automation-arizona-gm/`.

[17] "Mercedes-Benz Research & Development, Autonomous Driving." `http://mbrdna.com/divisions/autonomous-driving/`.

[18] "Openpilot." `https://github.com/commaai/openpilot`.

[19] "UMTRI Naturalistic Driving Data." http://www.umtri.umich.edu/our-focus/naturalistic-driving-data.

[20] "VTTI Naturalistic Driving Study." http://www.vtti.vt.edu/impact/index.html.

[21] "The OpenXC platform." http://openxcplatform.com/.

[22] T. Maze, M. Agarwai, and G. Burchett, "Whether weather matters to traffic demand, traffic safety, and traffic operations and flow," *Transportation research record: Journal of the transportation research board*, no. 1948, pp. 170–176, 2006.

[23] H. G. Min and E. T. Jeung, "Complementary filter design for angle estimation using mems accelerometer and gyroscope," *Department of Control and Instrumentation, Changwon National University, Changwon, Korea*, pp. 641–773, 2015.

[24] "GV200 Vehicle Tracker." `http://www.queclink.com/GV200`.

[25] J. Casselgren, M. Sjödahl, and J. LeBlanc, "Angular spectral response from covered asphalt," *Applied optics*, vol. 46, no. 20, pp. 4277–4288, 2007.

[26] Y. Kim, N. Baik, and J. Kim, "A study on development of mobile road surface condition detection system utilizing probe car," *Journal of Emerging Trends in Computing and Information Sciences*, vol. 4, no. 10, 2013.

[27] V. V. Viikari, T. Varpula, and M. Kantanen, "Road-condition recognition using 24-ghz automotive radar," *IEEE transactions on intelligent transportation systems*, vol. 10, no. 4, pp. 639–648, 2009.

[28] D. Gailius and S. Jačenas, "Iec detection on a road by analyzing tire to road friction ultrasonic noise," *Ultragarsas*, vol. 62, no. 2, pp. 17–20, 2007.

[29] Y. Kim, S. Oh, and Y. Hori, "Road condition estimation using acoustic method for electric vehicles," in *The 10th FISITA Student Congress*, pp. 30–6, 2010.

[30] R. Rajamani, *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

[31] L. Zhang and P. Prevedouros, "Motorist perceptions on the impact of rainy conditions on driver behavior and accident risk," in *Proceedings of the 84th Annual Meeting of the Transportation Research Board, Washington, DC*, 2005.

[32] Y. Wang, J. Yang, H. Liu, Y. Chen, M. Gruteser, and R. P. Martin, "Sensing vehicle dynamics for determining driver phone use," in *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*, pp. 41–54, ACM, 2013.

[33] "Methode Electronic Steering Wheel Angle Sensor." `http://www.methode.com/Documents/TechnicalLibrary/Steering_Angle_Sensor_Data_Sheet.pdf`.

[34] "Tesla autopilot." `https://en.wikipedia.org/wiki/Tesla_Autopilot`.

[35] "2019 audi a8 level 3 autonomy first-drive: Chasing the perfect 'jam'," *Slash Gear (November 7, 2017)*.

[36] L. Liu, H. Li, J. Liu, C. Karatas, Y. Wang, M. Gruteser, Y. Chen, and R. P. Martin, "Bigroad: Scaling road data acquisition for dependable self-driving," in *Proceedings of MobiSys 2017*, pp. 371–384, ACM, 2017.

[37] D. Chen, K.-T. Cho, S. Han, Z. Jin, and K. G. Shin, "Invisible sensing of vehicle steering with smartphones," in *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, pp. 1–13, ACM, 2015.

[38] D. A. Johnson and M. M. Trivedi, "Driving style recognition using a smartphone as a sensor platform," in *IEEE ITSC*, pp. 1609–1615, 2011.

[39] C. Oh, E. Jeong, K. Kang, and Y. Kang, "Hazardous driving event detection and analysis system in vehicular networks: Methodology and field implementation," *Transportation Research Record: Journal of the Transportation Research Board*, no. 2381, pp. 9–19, 2013.

[40] F. Jiang, J. Yuan, S. A. Tsaftaris, and A. K. Katsaggelos, "Anomalous video event detection using spatiotemporal context," *Computer Vision and Image Understanding*, vol. 115, no. 3, pp. 323–333, 2011.

[41] W. Sultani and J. Y. Choi, "Abnormal traffic detection using intelligent driver model," in *Pattern Recognition (ICPR), 2010 20th International Conference on*, pp. 324–327, IEEE, 2010.

[42] V. Saligrama and Z. Chen, "Video anomaly detection based on local statistical aggregates," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 2112–2119, IEEE, 2012.

[43] K. Yun, H. Jeong, K. M. Yi, S. W. Kim, and J. Y. Choi, "Motion interaction field for accident detection in traffic surveillance video," in *Pattern Recognition (ICPR), 2014 22nd International Conference on*, pp. 3062–3067, IEEE, 2014.

[44] S. Kamijo, Y. Matsushita, K. Ikeuchi, and M. Sakauchi, "Traffic monitoring and accident detection at intersections," *IEEE transactions on Intelligent transportation systems*, vol. 1, no. 2, pp. 108–118, 2000.

[45] D. M. Gavrila and S. Munder, "Multi-cue pedestrian detection and tracking from a moving vehicle," *International journal of computer vision*, vol. 73, no. 1, pp. 41–59, 2007.

[46] C. Guo, S. Mita, and D. McAllester, "Robust road detection and tracking in challenging scenarios based on markov random fields with unsupervised learning," *IEEE Transactions on intelligent transportation systems*, vol. 13, no. 3, pp. 1338–1354, 2012.

[47] Y. Tian, K. Pei, S. Jana, and B. Ray, "Deeptest: Automated testing of deep-neural-network-driven autonomous cars," *arXiv preprint arXiv:1708.08559*, 2017.

[48] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, pp. 1–16, 2017.

[49] Udacity, "Udacity: A self-driving car simulator built with unity," Feb 2018.

[50] I. Evtimov, K. Eykholt, E. Fernandes, T. Kohno, B. Li, A. Prakash, A. Rahmati, and D. Song, "Robust physical-world attacks on deep learning models," *arXiv preprint arXiv:1707.08945*, vol. 1, 2017.

[51] "Safety report – waymo," 2017.

[52] "Uber's self-driving cars hit 2 million miles as program regains momentum," 2017.

[53] "Self-driving uber vehicle strikes and kills pedestrian," 2018.

[54] D. Xu, E. Ricci, Y. Yan, J. Song, and N. Sebe, "Learning deep representations of appearance and motion for anomalous event detection," *arXiv preprint arXiv:1510.01553*, 2015.

[55] R. Chalapathy, A. K. Menon, and S. Chawla, "Anomaly detection using one-class neural networks," *arXiv preprint arXiv:1802.06360*, 2018.

[56] Nvidia, "Nvidia autonomous car." `https://www.youtube.com/watch?v=qhUvQiKec2U`, May 2016.

[57] C. A. research. `https://github.com/commaai/res`, 2016.

[58] Cg23. `https://github.com/udacity/self-driving-car/tree/master/steering-models/community-models/cg23`, 2017.

[59] Rambo. `https://github.com/udacity/self-driving-car/tree/master/steering-models/community-models/rambo`, 2017.

[60] Chauffeur. https://github.com/udacity/self-driving-car/tree/master/steering-models/community-models/chauffeur, 2017.

[61] Komanda. https://github.com/udacity/self-driving-car/tree/master/steering-models/community-models/komanda, 2017.

[62] "Udacity self driving car challenge dataset ch2_002." https://github.com/udacity/self-driving-car/tree/master/datasets/CH2.

[63] "Udacity el camino training data for self driving car challenge." http://academictorrents.com/details/e9b47deb3391e33df794e5ec4399d38ef8767c07.

[64] Autumn. https://github.com/udacity/self-driving-car/tree/master/steering-models/community-models/autumn, 2017.

[65] J. Dai, J. Teng, X. Bai, Z. Shen, and D. Xuan, "Mobile phone based drunk driving detection," in *PervasiveHealth*, 2010.

[66] Y. Wang, Y. J. Chen, J. Yang, M. Gruteser, R. P. Martin, H. Liu, L. Liu, and C. Karatas, "Determining driver phone use by exploiting smartphone integrated sensors," *IEEE Transactions on Mobile Computing*, vol. 15, no. 8, pp. 1965–1981, 2016.

[67] L. Liu, C. Karatas, H. Li, S. Tan, M. Gruteser, J. Yang, Y. Chen, and R. P. Martin, "Toward detection of unsafe driving with wearables," in *Proceedings of the 2015 workshop on Wearable Systems and Applications*, pp. 27–32, ACM, 2015.

[68] C. Karatas, L. Liu, H. Li, J. Liu, Y. Wang, S. Tan, J. Yang, Y. Chen, M. Gruteser, and R. Martin, "Leveraging wearables for steering and driver tracking," in *Computer Communications, IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on*, pp. 1–9, IEEE, 2016.

[69] C. Karatas, L. Liu, M. Gruteser, and R. Howard, "Single-sensor motion and orientation tracking in a moving vehicle," in *2018 15th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, IEEE, 2018.

[70] Q. Chen, X. Ma, S. Tang, J. Guo, Q. Yang, and S. Fu, "F-cooper: feature based cooperative perception for autonomous vehicle edge computing system using 3d point clouds," in *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*, pp. 88–100, 2019.

[71] Z. Gojcic, C. Zhou, J. D. Wegner, and A. Wieser, "The perfect match: 3d point cloud matching with smoothed densities," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5545–5554, 2019.

[72] L. Li, S. Zhu, H. Fu, P. Tan, and C.-L. Tai, "End-to-end learning local multi-view descriptors for 3d point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1919–1928, 2020.

[73] P. J. Besl and N. D. McKay, "Method for registration of 3-d shapes," in *Sensor fusion IV: control paradigms and data structures*, vol. 1611, pp. 586–606, International Society for Optics and Photonics, 1992.

[74] C. Choy, W. Dong, and V. Koltun, "Deep global registration," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2514–2523, 2020.

[75] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[76] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[77] B. Wu, A. Wan, X. Yue, and K. Keutzer, "Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1887–1893, IEEE, 2018.

[78] J. Park, Q.-Y. Zhou, and V. Koltun, "Colored point cloud registration revisited," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 143–152, 2017.

[79] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (fpfh) for 3d registration," in *2009 IEEE international conference on robotics and automation*, pp. 3212–3217, IEEE, 2009.

[80] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu, "Fast 3d recognition and pose using the viewpoint feature histogram," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2155–2162, IEEE, 2010.

[81] A. Aldoma, F. Tombari, R. B. Rusu, and M. Vincze, "Our-cvfh–oriented, unique and repeatable clustered viewpoint feature histogram for object recognition and 6dof pose estimation," in *Joint DAGM (German Association for Pattern Recognition) and OAGM Symposium*, pp. 113–122, Springer, 2012.

[82] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, 2017.

[83] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, and T. Funkhouser, "3dmatch: Learning local geometric descriptors from rgb-d reconstructions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1802–1811, 2017.

[84] H. Deng, T. Birdal, and S. Ilic, "Ppfnet: Global context aware local features for robust 3d point matching," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 195–205, 2018.

[85] C. Choy, J. Park, and V. Koltun, "Fully convolutional geometric features," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 8958–8966, 2019.

[86] K.-L. Low, "Linear least-squares optimization for point-to-plane icp surface registration," *Chapel Hill, University of North Carolina*, vol. 4, no. 10, pp. 1–3, 2004.

[87] A. Censi, "An icp variant using a point-to-line metric," in *2008 IEEE International Conference on Robotics and Automation*, pp. 19–25, Ieee, 2008.

[88] Y. Wang and J. M. Solomon, "Deep closest point: Learning representations for point cloud registration," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3523–3532, 2019.

[89] W. Lu, G. Wan, Y. Zhou, X. Fu, P. Yuan, and S. Song, "Deepvcp: An end-to-end deep neural network for point cloud registration," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 12–21, 2019.

[90] J. Yang, H. Li, D. Campbell, and Y. Jia, "Go-icp: A globally optimal solution to 3d icp point-set registration," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 11, pp. 2241–2254, 2015.

[91] Q.-Y. Zhou, J. Park, and V. Koltun, "Fast global registration," in *European Conference on Computer Vision*, pp. 766–782, Springer, 2016.

[92] D. Campbell and L. Petersson, "Gogma: Globally-optimal gaussian mixture alignment," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5685–5694, 2016.

[93] M. Magnusson, A. Lilienthal, and T. Duckett, "Scan registration for autonomous mining vehicles using 3d-ndt," *Journal of Field Robotics*, vol. 24, no. 10, pp. 803–827, 2007.

[94] H. Yang, J. Shi, and L. Carlone, "Teaser: Fast and certifiable point cloud registration," *arXiv preprint arXiv:2001.07715*, 2020.

[95] P. W. Theiler, J. D. Wegner, and K. Schindler, "Globally consistent registration of terrestrial laser scans via graph optimization," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 109, pp. 126–138, 2015.

[96] Z. Gojcic, C. Zhou, J. D. Wegner, L. J. Guibas, and T. Birdal, "Learning multiview 3d point cloud registration," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1759–1769, 2020.

[97] P. W. Theiler, J. D. Wegner, and K. Schindler, "Keypoint-based 4-points congruent sets–automated marker-less registration of laser scans," *ISPRS journal of photogrammetry and remote sensing*, vol. 96, pp. 149–163, 2014.

[98] B. Douillard, A. Quadros, P. Morton, J. P. Underwood, M. De Deuge, S. Hugosson, M. Hallström, and T. Bailey, "Scan segments matching for pairwise 3d alignment," in *2012 IEEE International Conference on Robotics and Automation*, pp. 3033–3040, IEEE, 2012.

[99] B. Gálai, B. Nagy, and C. Benedek, "Crossmodal point cloud registration in the hough space for mobile laser scanning data," in *2016 23rd International Conference on Pattern Recognition (ICPR)*, pp. 3374–3379, IEEE, 2016.

[100] B. Nagy and C. Benedek, "Real-time point cloud alignment for vehicle localization in a high resolution 3d map," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 0–0, 2018.

[101] T.-H. Wang, S. Manivasagam, M. Liang, B. Yang, W. Zeng, J. Tu, and R. Urtasun, "V2vnet: Vehicle-to-vehicle communication for joint perception and prediction," *arXiv preprint arXiv:2008.07519*, 2020.

[102] S.-W. Kim, B. Qin, Z. J. Chong, X. Shen, W. Liu, M. H. Ang, E. Frazzoli, and D. Rus, "Multivehicle cooperative driving using cooperative perception: Design and experimental validation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 663–680, 2014.

[103] H. Qiu, F. Ahmad, F. Bai, M. Gruteser, and R. Govindan, "Avr: Augmented vehicular reality," in *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*, pp. 81–95, ACM, 2018.

[104] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part i," *IEEE robotics & automation magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[105] Y. Wang, V. Menkovski, I. W.-H. Ho, and M. Pechenizkiy, "Vanet meets deep learning: The effect of packet loss on the object detection performance," in *2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*, pp. 1–5, IEEE, 2019.

[106] J. Lee, M. Cho, and K. M. Lee, "Hyper-graph matching via reweighted random walks," in *CVPR 2011*, pp. 1633–1640, IEEE, 2011.

[107] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the society for industrial and applied mathematics*, vol. 5, no. 1, pp. 32–38, 1957.

[108] Google, "Android location."

[109] M. Cho, J. Lee, and K. M. Lee, "Reweighted random walks for graph matching," in *European conference on Computer vision*, pp. 492–505, Springer, 2010.

[110] R. Guo, H. Lu, P. Gao, Z. Zhang, and H. Zhang, "Collaborative localization for occluded objects in connected vehicular platform," in *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*, pp. 1–6, IEEE, 2019.

[111] O. Duchenne, F. Bach, I.-S. Kweon, and J. Ponce, "A tensor-based algorithm for high-order graph matching," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 12, pp. 2383–2395, 2011.

[112] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum pointnets for 3d object detection from rgb-d data," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 918–927, 2018.

[113] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, *et al.*, "Speed/accuracy trade-offs for modern convolutional object detectors," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7310–7311, 2017.

[114] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečnỳ, S. Mazzocchi, H. B. McMahan, *et al.*, "Towards federated learning at scale: System design," *arXiv preprint arXiv:1902.01046*, 2019.