Understanding Drivers’ Risk Behaviors from Dashcam Videos

by

Korrawat Pruegsanusak

Submitted to the Department of Electrical Engineering and Computer Science on May 29, 2020, in partial fulfillment of the requirements for the degree of Master of Engineering in Electrical Engineering and Computer Science

Abstract

Dangerous driving causes preventable injuries and deaths. To promote safe driving practices, a forward-facing dashboard camera or dashcam can provide visual context of the environment in which a driver exhibits dangerous driving behaviors. In this thesis, I designed and implemented a system that identifies and analyzes dangerous driving events using monocular videos from dashcams and odometry data from smartphone sensors. To extract useful information such as the position of the ego-vehicle in the lane, the following distance to the next vehicle, and the position and velocity of other vehicles in real-world coordinates, the system performs multiple computer vision tasks, including 2D and 3D object detection and tracking, lane detection, and camera calibration. Then, the information from perception tasks are used to identify risky events, including tailgating and running stop signs, analyze the speed of traffic flow, and classify the cause for hard braking events, such as reacting to the next car in the lane, stopping for a pedestrian, or being inattentive. Using over 8 hours of dashcam videos in multiple driving scenarios, experiments were performed to demonstrate the system’s capabilities and limitations.

Thesis Supervisor: Paresh Malalur
Title: Head of AI and Vision, Cambridge Mobile Telematics

Thesis Supervisor: Hari Balakrishnan
Title: Professor of Computer Science
I wish to express my gratitude to my supervisors, Paresh Malalur and Prof. Hari Bala-
krishnan, for giving me a challenging and impactful project, guiding me through the
process, and providing me with invaluable insights. A lot of data used in this thesis was
collected by Hari himself. The project would have not been possible without their help.

I would like to extend my sincere thanks to my coworkers at Cambridge Mobile Telem-
atics for both technical and social support. Onkar Trivedi performed an early research
on this topic and continued to answer my endless questions, particularly when I started
working on this project. Dheeptha Badrinarayanan never failed to offer any assistance,
and my discussion with her and Onkar deepened my understanding of mathematical con-
cepts and engineering principles for computer vision and deep learning algorithms. Olga
Oleksyuk shared her insightful advice and assisted me in setting up Docker, allowing me
to work on this thesis more easily. I also appreciate everyone who has worked on the
pipeline to collect dashcam videos and process phone data into a format which I could
easily use. I am also grateful to Joe Palin, who always brightened my days in the CMT
office.

I have been fortunate to learn from great teachers. I would like to thank Prof. Fadel
Adib for giving me an opportunity to be a Teaching Assistant for 6.808 for the past two
Spring semesters. He has been a model for teaching, mentoring, and communicating
about exciting research with passion. I wish to also thank Prof. Frédo Durand for the
computational photography course and a UROP opportunity, and Prof. Justin Solomon
for the computer graphics and geometric data processing courses. These courses and the
UROP introduced me to mathematical principles and research problems that are the basis
for this thesis and other projects.

My undergraduate and graduate studies at MIT would not have been the same without
wonderful friends. I wish to specially thank Anupong Tangpeerachaikul, Kritkorn Karn-
tikoon, and Nipun Pitimanaaree, for numerous meals and late-night discussions, always
making sure I have not missed a meal even in the busiest times. I would like to also thank
Suchan Vivatsethachai and Cattalyya Nuengsigkapian for helping with many course as-

5
signments and projects, as well as for sharing an interest in photography. Thanks to Apisada, Watchara, Puwanat, Tossaporn, Tanat, Veerapatr, Varot, Ronnachai, Weerachai, Laphonchai, Thanard, Sorawit, Warittha, SEALNet members, the TS community, MISTI friends, and many others, for providing guidance, sharing moments, and making the past five years a memorable chapter of my life.

Growing up in Thailand, I am grateful to have Noppadon Kamolvilassatian as my lifelong teacher and mentor, who saw potential in me since I was young and taught me complex computational concepts in a simple and engaging way. Always staying ahead of the most advanced technology, he has provided me with countless advice that broadened my horizon and continues to send me interesting articles, papers, and talks about artificial intelligence and other news around the world. I am also lucky to have Bundit Diloknawarit as a friend, a classmate, and a neighbor. His intelligence has helped me grasp challenging materials in many subjects, his curiosity shown me how to never stop learning, and his wit assured me someone still understands my quirky sense of humor.

I would like to express my deepest gratitude to my parents for their unconditional love and support, especially during the ongoing COVID-19 pandemic. I wish to also thank my brother and my sister for sharing enthusiasm in mathematics and sciences and for helping me get to where I am today. Just ten years ago, I had only lived in the same small city and, although I knew that studying abroad was a rare honor, it was incomprehensible what it would really entail. To have a family that supports every decision that I make and helps me navigate through the uncertainties, even when I am far from home, is a privilege I could not have asked from elsewhere.

Finally, Chanoot Sirisoponsilp has been my infinite source of joy and inspiration. She has taught me to be kind to myself and others, and her encouragement and patience have helped me stay on track for this thesis and whenever I get lost in the journey of my life.
## Contents

1 **Introduction** 13  
  1.1 Motivations 13  
  1.2 Challenges 14  
  1.3 Contributions 15  
  1.4 Thesis Outline 16  

2 **Background and Related Work** 17  
  2.1 Driving Behaviors and Dashcam Videos 17  
  2.2 2D Perception Tasks 18  
    2.2.1 2D Object Detection and Segmentation 19  
    2.2.2 Video Object Detection 19  
    2.2.3 Multi-Object Tracking 20  
    2.2.4 Visual Object Tracking 21  
    2.2.5 Lane Detection 22  
  2.3 3D Computer Vision Tasks 22  
    2.3.1 Monocular 3D Object Detection 23  
    2.3.2 Automatic Dashcam Calibration 24  
    2.3.3 Ground Plane Estimation 24  
  2.4 Previous Work Done at CMT 25  

3 **Data Sources** 27  
  3.1 Dashcam Videos 27  
  3.2 Phone Sensors 29
**List of Figures**

3-1  Distributions of the number of videos in terms of duration, time taken, and hours of the day. 28

3-2  Example frames from dashcam videos. 30

4-1  Overview of the system. 34

4-2  The vehicle and camera coordinate systems. 35

4-3  Original and rectified frames of example frames. 39

4-4  Example images of the two camera calibration approaches in action. 42

4-5  Correlation plots for camera extrinsic calibration results from two approaches. 46

4-6  Architecture for 2D object detection and tracking. 49

4-7  Example results for 2D object detection and tracking. 52

4-8  An example frame for 3D object detection of vehicles. 53

5-1  Examples of video frames for hard braking event classification. 57

5-2  Confusion matrix of event classification for hard braking. 58

5-3  Following distance vs. time for hard braking when following another vehicle. 59

5-4  Plots for stop sign braking events. 61

5-5  Scatter plots of average following distance/time to the next vehicle vs. average speed, showing potential tailgating events. 63

5-6  Scatter plots of the ego-vehicle’s speed relative to the traffic flow vs. its speed. 65
List of Tables

3.1 Technical specifications of the camera used in this experiment. . . . . . . 28
3.2 Phone data attributes. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 31
3.3 Example of data from phone sensors. . . . . . . . . . . . . . . . . . . . . . . . . 31
Chapter 1

Introduction

1.1 Motivations

Dangerous driving leads to deaths and injuries. Careless drivers compromise the safety of other drivers on the road as well as passengers, cyclists, and pedestrians. According to National Highway Traffic Safety Administration, there were over 32,000 traffic fatalities every year from 2008 to 2017 in the United States [19]. Despite the evident statistics, many drivers may not realize how often they exhibit dangerous driving behaviors. To promote awareness and encourage safe driving practices, collecting driving data and providing feedback to drivers can help them improve their behaviors and increase overall road safety.

While some newer luxury cars are equipped with an array of sensors, safety features, and Advanced driver-assistance systems (ADAS), the majority of vehicles still lack these features. Here, smartphones serve as a valuable tool to collect driving data without having to install extra sensors to the car, as they already come with a suite of sensors, such as accelerometers, gyroscopes, and GPS. Using the readily available sensors installed on drivers’ smartphones has been shown to reduce risky behaviors such as speeding by 30% and hard braking by 51% for the top 30% of drivers after 30 days of usage [1].

However, while phone sensors can provide estimates of position and movement of an ego-vehicle, no information about other vehicles on the road or the environment is available, making it difficult to identify environment-based behaviors, such as tailgating, or to learn about the context in which a driver performs dangerous behaviors, for example,
what caused a driver to suddenly hit the brake. As a result of rapid progress in technology, dashboard cameras, or *dashcams*, have become increasingly affordable and ubiquitous in new cars. To better evaluate a driver’s behaviors, dashcam videos can provide visual context of the driver’s environment to complement the information available from smartphone sensors, such as position and velocity of other vehicles, lane markings, and traffic signs.

Previous works [66, 6] have proposed systems that use smartphone cameras and sensors to detect risky driving behaviors. However, dashcams are equipped in many cars offer a better image quality and a wider field of view. Combined with new algorithms and recent datasets from the research community, using dashcam videos can improve robustness of dangerous driving detection in challenging driving scenarios, as well as open up new avenues for analyses of behaviors that were not available before.

### 1.2 Challenges

Analyzing driving behaviors requires an understanding of driving scenes in the coordinate system of the road to estimate the positions of surrounding objects, for example, other cars, lane markings, and pedestrians crossing the street. This requires a combination of several computer vision tasks, such as object detection and tracking, and transformation of image to real-world coordinates. While advances in these tasks have benefited from a large number of research in autonomous vehicles or ADAS, dashcam videos and phone sensor data from users in a variety of driving scenarios still present several challenges, including lack of depth data, a variation in camera placement, and challenging driving scenarios.

First, dashcam videos are monocular and, unlike systems used in autonomous vehicles, are not augmented with depth sensors such as LiDAR, Radar, or ultrasonic sensor. Stereo or multi-view cameras and depth sensors help autonomous vehicles estimate positions of other vehicles even when those vehicles are not visible from a particular view. As a result, any algorithm that uses depth sensor data in addition to RGB images is not applicable to monocular dashcam videos. It is possible, nevertheless, to train a model
with these datasets using depth data for supervision and apply it to our data for inference. Nevertheless, annotating our monocular dashcam video data for 3D tasks, such as 3D bounding boxes of vehicles and pedestrians, is infeasible, because they are typically annotated on point clouds provided by depth sensors. With no practical solution to annotate and train 3D tasks with only dashcam videos data, the issue of difference in dataset distributions still persists.

Second, the camera’s position and orientation are often unknown and different from one user to another or even across different times for the same user. Because the camera placement is needed to transform from image to real-world coordinates, the error in estimating those parameters translate to inaccuracies of distance estimation. In contrast, cameras in autonomous vehicles and those used in standard road datasets are often fixed or have very small variation. For example, the KITTI dataset [21] provides precise measurements of the camera’s position and orientation for all video frames. Consequently, a deep learning model for 3D-related tasks, such as depth estimation, is often trained on datasets with fixed camera parameters. The trained model implicitly learns the camera parameters and therefore is not directly applicable to our dashcam videos with a large variation in camera parameters.

Lastly, dashcam videos from users are taken from multiple driving scenarios and contain frames with reflection, glare, or dirty windshield. Widely used benchmark datasets, such as KITTI [21] and Cityscapes [13] do not have night or raining scenes and contain images from cameras that are placed outside of the vehicle to avoid reflection. Thus, models trained on these datasets are not always robust to challenging scenes.

1.3 Contributions

In this thesis, I developed a system that performs multiple computer vision tasks to extract information from dashcam videos from real-world driving scenarios. The perception tasks include fisheye rectification, camera calibration, lane detection, and object detection and tracking in 2D and 3D. In particular, the object detection and tracking pipeline leverages temporal properties of videos to improve recall over frame-by-frame detection.
Additionally, two camera calibration approaches are presented, based on lane markings and keypoints tracks. An experiment shows that the results from both methods strongly agree with each other in a large variety of scenarios.

Information from dashcam videos is combined with basic odometry data from smartphone sensors to analyze patterns of dangerous driving behaviors. Hard braking events, detected by large negative acceleration, are classified into six categories of causes for the events, such as reacting to the next car in the lane, stopping for a pedestrian, or being inattentive. Moreover, the following distance estimation is used to identify tailgating, compared to the three-second rule guideline of safe driving. Finally, the tracking of vehicles in 3D is used to estimate the speed of traffic flow compared to the speed of ego-vehicle and the speed limit.

### 1.4 Thesis Outline

The remainder of this thesis is divided into five chapters. Chapter 2 describes related work on analyses of driving behaviors and relevant computer vision tasks. Chapter 3 describes the data sources, dashcam videos and phone sensors. Chapters 4 and 5 together explain the designs and results of the system. Chapter 4 describes the perception tasks, including object detection and tracking, lane detection, and camera calibration. Chapter 5 describes how processed data from dashcam videos and phone sensors are used to identify and analyze dangerous driving behaviors. Finally, Chapter 6 concludes the thesis and discusses potential improvements.
Chapter 2

Background and Related Work

This section describes background and related work in two main areas. The first area is using dashcam videos to study driving behaviors, as explained in §2.1. The second area is perception tasks relevant to this thesis. In §2.2, I discuss detection and tracking tasks performed in 2D pixel coordinates. In §2.3, I discuss tasks related to transforming from pixel coordinates to real-world coordinates in the ground plane.

2.1 Driving Behaviors and Dashcam Videos

Previous works use smartphone cameras and smartphone sensors to detect dangerous behaviors, then alert the driver in real time or evaluate their driving styles. You et al. proposed CarSafe [66], a smartphone app that uses phone sensors data together with both front and back cameras of a smartphone to analyze the road environment and the driver’s face to alert the driver of dangerous behaviors in real time. Alerts based on a road-facing camera include tailgating and lane weaving and drifting, and those based on a driver-facing camera are drowsy driving, inattentive driving, and careless lane change (i.e., not turning head to check blind spots). Similarly, Bergasa et al. proposed Drivesafe [6], which only use the road-facing camera, but also attempt to give a quantitative driving score. They also used a microphone to check whether blinkers’ sound are on during lane changes.

In this thesis, a dashcam is used as a separate device from the smartphone. With larger
image sensors, dashcam videos provide better image quality and usually wider field of view than smartphone cameras. In comparison to earlier works, many components in the earlier works, such as object detection, can be improved with deep learning based methods as a result of more computation power as well as new algorithms and datasets developed in the past few years. This provides better robustness of object detection and lane detection tasks, especially in challenging scenarios, such as driving at night or in a residential area where lanes are not clearly marked. In addition to improved algorithms, more detailed analyses are offered, such as 3D detection of vehicles. Some assumptions about the camera parameters are also relaxed, i.e., camera height and angles are unknown and are inferred from videos. Unlike CarSafe [66], the analysis in this work does not use a driver-facing camera and is therefore unable to detect behaviors based on a driver’s face and gaze, such as drowsy driving. Yet, an extension to include a driver-facing camera using a dual-camera dashcam is an interesting future direction.

Another interesting work worth mentioning is from Chan et al. [11], which use crowdsourced dashcam videos in Taiwan to anticipate accidents in the scene. The proposed system uses an LSTM network to predict whether an accident will happen for each frame of a 5-second video clip. This work does not explicitly transform the vehicles’ coordinates into the ground plane, but lets the LSTM handle given 2D bounding boxes and output an accident anticipation score. While this work does not directly analyze behaviors of the ego-driver, it can be helpful to alert the drivers or autonomous vehicles of potential accidents.

2.2 2D Perception Tasks

Understanding a driving scene in real-world coordinates, e.g., what the positions and velocities of other vehicles in the scene are, begins with detecting the objects in the 2D image pixels. The relevant tasks for dashcam videos include 2D object detection and segmentation, video object detection, object tracking, and lane detection.
2.2.1 2D Object Detection and Segmentation

Huang et al. [33] offer a detailed comparison of multiple state-of-the-art CNN-based 2D object detectors and extensively analyzes speed and accuracy trade-offs among them. Here, I briefly review two main categories of object detectors: single shot detector and two-stage detectors. **Single shot detectors**, such as SSD [41] and variations of YOLO [52, 53, 54], directly predict classes and bounding boxes via a single convolutional neural network. This architecture benefits from fast inference time often at a cost of lower accuracy.

The other category, **two-stage detectors**, such as R-CNN [23], Fast R-CNN [22], Faster R-CNN [55] and R-FCN [14], perform the detection task in two stages: *proposal generation* and *classification*. The first stage takes an input image and encodes it via a backbone and then the region proposal network (RPN) outputs box proposals (also called region of interests or RoIs) for potential objects. In the second stage, the *head* takes encoded features from the backbone for each RoI to classify and refine via a classification branch and a box refinement branch, respectively.

While a two-stage detector typically has a slower inference time than a single shot detector, it can be easily extended to perform a new related task. For example, to perform instance segmentation, Mask R-CNN [29] adds a *mask* branch to the head of Faster R-CNN [55] to predict whether each pixel in a bounding box belongs to the object of interest. It can also be extended to a 3D detection task. Hu et al. [32] adds a branch to predict the 2D projection of the 3D bounding box’s center. With this advantage for availability for extension, this thesis uses the Faster/Mask R-CNN architecture for the 2D object detection/segmentation task.

2.2.2 Video Object Detection

While a naive approach to detecting objects in a video is to apply an object detector frame by frame, it can be improved by leveraging temporal consistency and constraints of videos. An object in one frame should appear in a nearby location and have a similar appearance in the next frame. Consecutive frames also contain similar information. These temporal properties of videos can be used to improve accuracy and/or speed of detection.
Recent approaches to video object detection can be divided into three groups. First, **postprocessing methods**, such as Seq-NMS [27] and T-CNN [36], link single-image detections together to create tracks and change detection confidences based on other detections in the same track. Second, **feature flow methods** use optical flow to propagate intermediate features in a convolutional neural network between video frames. DFF [70] uses this idea to compute detections only on sparse keyframes and propagate information to the rest, leading to faster computation. In contrast, FGFA [69] proposes to compute per-frame detections on all frames and also warp features from neighboring frames to get more accurate results. Lastly, **multi-frame methods**, such as D&T [18], take multiple video frames as inputs simultaneously. However, given that the standard benchmark datasets for this video object detection task are vastly different from road datasets, using available models would require retraining on a large dataset of road-scene videos. This work therefore makes use of a combination of algorithms for multi-object tracking (§2.2.3) and visual object tracking (§2.2.4) to leverage temporal properties of videos instead.

### 2.2.3 Multi-Object Tracking

The goal of **multi-object tracking** (MOT) is to track multiple objects of interest across video frames. For dashcam videos, MOT is important because it provides trajectories of vehicles in the scene, which are used to calculate their relative speeds, estimate the number of cars, and give a better understanding of the driving scene.

The state-of-the-art MOT online methods perform the task in two stages: detection and association. Detecting objects in the frame can be done using any object detector, as discussed in §2.2.1. An internal state (e.g. bounding box position and motion) for each object track is estimated with the Kalman Filter. The data association stage then matches detections in the new frame with existing tracks, and is typically solved with the Hungarian algorithm, as proposed in SORT [8]. To make matching more robust, DeepSORT [62] also keeps the state for visual appearance for each object track and compare detections by calculating cosine similarity between the CNN-based embeddings of those detections. Newer methods, such as DeepMOT [63], use neural networks to construct the association.

Additionally, recent works, such as TrackR-CNN [59] and FairMOT [68], propose
one-shot MOT to address detection and tracking in one network to increase efficiency. However, the accuracy is typically lower than the widely used two-step methods and this type of methods requires training on videos with object track annotations. Recent works [59, 42] also propose algorithms to solve an extended version of the MOT task, *multi-object tracking and segmentation* (MOTS), in which the system needs to track not only the bounding boxes but also the segmentation masks for those boxes.

In this thesis, the typical two-shot detection then tracking architecture is used, with DeepSORT [62] being used in the tracking stage. This allows for using an object detector trained on large image datasets, which is suitable for challenging scenarios in dashcam videos.

### 2.2.4 Visual Object Tracking

The goal of another related task, visual object tracking (VOT), is to track a generic object across frames in a video based on its motion and visual appearance, given an initial bounding box in the first frame. This problem typically assumes no class prior, *i.e.*, the tracked object is not constrained to a particular object type such as car or person.

Modern VOT algorithms can be categorized into three groups: online-trained, offline-trained, and hybrid trackers. **Online-trained trackers**, such as DSST [15] continually learn features of the target object at test-time and are often inefficient. A different group, **offline-trained trackers** use CNNs pre-trained on large labeled video datasets and are very fast at test-time, such as [30, 7]. However, they do not learn from new visual information and can often be confused by occlusions or large changes in appearance. To balance the trade-off between speed and accuracy, **hybrid trackers** train CNNs offline but also adapt to new information, such as MDNet [46] and Re3 [26]. In this work, Re3 [26] is used to improve frame-by-frame object detections in videos by reducing missed detections (thus increasing recall).
2.2.5 Lane Detection

Lane markings provide crucial information for navigating a vehicle on roads, especially with other drivers in the environment. For this thesis, lane markings help us understand the ego-vehicle’s position and orientation compared to the road direction, as well as those of other vehicles and objects in the scene. Under ideal conditions, lane markings can also be used for camera calibration, i.e., estimating the camera’s angles and height.

Many approaches have been proposed for the lane detection problem. Traditional methods use hand-crafted low-level features [5]. However, they require complex feature engineering and are often not robust to a large variation of the environments. Recent works use deep learning models and treat lane detection as a segmentation problem [49, 47, 31]. Hou et al. [31] use self-attention distillation to improve training of lane segmentation and add a branch to predict lane existence for a fixed number of lanes. While most works assume that the road plane is flat, Garnett et al. [20] propose to estimate the coordinates of lane markings in 3D. However, collecting a large dataset of lane markings with 3D annotations is expensive and challenging, so they create a synthetic dataset for their experiment. Another interesting extension is to perform lane detection on a video instead of a single image. Similar to the video object detection task (§2.2.2), Zou et al. [71] propose a method based on Convolutional LSTM to improve temporal consistency of detection for a video input.

In this work, I use a model from Hou et al. [31] with an ERFNet backbone [56], trained on the CULane dataset [49].

2.3 3D Computer Vision Tasks

To make sense of the 2D detected objects, the detections need to be transformed to real-world coordinates. For the context of dashcam videos, the transformation problem is related to multiple tasks, including monocular 3D object detection, automatic dashcam calibration, ground plane estimation, and monocular depth estimation.
2.3.1 Monocular 3D Object Detection

The task of monocular 3D object detection is to localize 3D bounding boxes of desired objects from an image or video frames. Most methods first use the results of 2D object detection or instance segmentation as inputs to the model. Unlike stereo or LiDAR-based 3D object detection, monocular camera provides no direct access to depth information, making the problem ill-posed because different 3D object position and orientation can project to the same 2D image. Therefore, recent works have opted for deep learning models to learn from large datasets with supervision from annotated 3D bounding boxes. For detecting objects in road scenes, such as vehicles and pedestrians, many works proceed with a reasonable assumption that objects are on flat ground and most estimate 3D bounding boxes that are parallel to the ground.

Chen et al. [12] propose to sample candidate 3D bounding boxes that are situated on the ground plane and score their 2D projections based on multiple priors, such as location, context, and instance segmentation. Mousavian et al. [45] propose a 2D CNN to estimate each object’s size and orientation and use constraints of 2D-3D bounding box consistency to calculate the object’s center location. Recently, state-of-the-art methods are based on pseudo-LiDAR representation. These approaches [60, 61, 50] leverage success in 3D object detection of LiDAR point-cloud by performing monocular depth estimation (??) to project 2D pixels into 3D pseudo-LiDAR point clouds. Some studies [39, 44] extend the problem to predict not only 3D bounding boxes but also 3D shapes, based on a collection of 3D CAD models of vehicles.

Like the 2D case, recent studies also leverage temporal properties of videos to improve frame-by-frame 3D detection. Hu et al. [32] propose an algorithm to detect 3D bounding boxes and perform multi-object tracking (§2.2.3). They show that performing MOT in 3D improves detection and tracking over 2D. For state estimation of the each object track, they experiment with both the Kalman Filter and an LSTM.

Another related task is a recent work by Kampelmühler et al. [35] which proposes to directly estimate each vehicle’s position and velocity without explicitly localizing 3D bounding boxes, by combining networks for flow estimation, depth estimation, and tracker
to regress directly to position and velocity outputs for each 2D bounding box.

While the pseudo-LiDAR based methods are state-of-the-art, they require accurate monocular depth estimation. However, available models for MDE are trained on datasets with minimal variation in camera placement and are not directly applicable to dashcam videos with large variation. With these limitations, this work uses an algorithm based on Mousavian et al. [45] because of its simplicity and fast inference time, then performs smoothing based on the 3D Kalman Filter from Hu et al. [32].

2.3.2 Automatic Dashcam Calibration

User-installed cameras are subject to unknown extrinsic camera calibration parameters, i.e. camera’s position and orientation. (The intrinsic camera parameters are assumed to be known.) The camera’s orientation is defined by three angles—roll, pitch and yaw—and its position defined by its height from the ground.

The angles are typically calculated from the forward vanishing point (FVP) and other straight-line structures in the image. The camera height is more challenging, as it often requires other assumptions, such as known vehicle’s speed or known lane width. de Paula et al. [16] use parallel lane markings to estimate pitch and yaw, and estimate the camera height by tracking dashed lane markers, assuming either that the longitudinal distance between consecutive dashed markers is constant or that the vehicle’s speed is known. Tummala et al. [58] leverage a straight motion of the ego-vehicle to track feature points in the scene, which yields an estimate of the forward vanishing point. The algorithm estimates the camera height with known vehicle’s speed similar to [16].

2.3.3 Ground Plane Estimation

Ground plane estimation is strongly related to the extrinsic calibration problem. The problem aims at finding the direction of the normal vector of the ground, which is equivalent to estimating the roll and pitch angles of the camera. Another representation is the horizon line, which can be calculated from the two angles. Dragon and Van Gool [17] propose a method based on a Hidden Markov Model (HMM) for a video data. Some recent works use
deep learning models in solving the problem. Man et al. [43] propose GroundNet, which performs normal estimation, depth estimation, and ground segmentation with a shared encoder. The normal and depth are improved by imposing a geometric consistency loss.

2.4 Previous Work Done at CMT

This thesis is conducted at Cambridge Mobile Telematics (CMT). Previous work done at CMT that I utilize for this thesis include the data acquisition and preprocessing pipeline (Chapter 3), the baseline algorithm for 2D object detection and tracking (§4.5.1), and the parameters from intrinsic calibration (§4.2).
Chapter 3

Data Sources

The data used in this thesis are obtained from dashcam videos and phone sensors. The following sections explains the detail of each data source.

3.1 Dashcam Videos

In this thesis, all dashcam videos were collected from Cambridge Mobile Telematics’ test users by front-facing cameras mounted behind windshields. Videos are monocular (not stereo), i.e., only a single camera was mounted in each vehicle. They were taken by the same camera model, Yi Smart Dash Camera [3], which allows for a one-time calibration for intrinsic and undistortion parameters (see §4.2). The technical specifications of the camera according to the manufacturer are detailed in Table 3.1. The videos were taken at the resolution of 1280 × 720 and 30 fps. Though not at the maximum resolution and frame rate, these settings are adequate for practical use cases of this thesis and decrease storage space and video processing time.

The videos were taken in real driving situations. While there is no experimental setup to collect data for dangerous driving events with specific instructions for the drivers like [66, 6], many videos were collected when mobile sensors determined some secondary risk features, such as hard braking, and triggered the dashcam to start recording. The number of videos in the dataset is sufficiently large, containing a variety of different driving scenarios and risky driving behaviors.
Table 3.1: Technical specifications of the camera used in this experiment.

<table>
<thead>
<tr>
<th>Camera Attribute</th>
<th>Camera Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model name</td>
<td>Yi Smart Dash Camera</td>
</tr>
<tr>
<td>Model number</td>
<td>C10</td>
</tr>
<tr>
<td>Min Aperture (F/#)</td>
<td>F/1.8</td>
</tr>
<tr>
<td>FOV</td>
<td>165°</td>
</tr>
<tr>
<td>Sensor</td>
<td>1/2.7” CMOS Sensor</td>
</tr>
<tr>
<td>Max resolution</td>
<td>1920 × 1080 (60 fps) or 2304 × 1296 (30 fps)</td>
</tr>
<tr>
<td>Resolution used</td>
<td>1280 × 720 (30 fps)</td>
</tr>
</tbody>
</table>

Figure 3-1: Distributions of the number of videos in terms of duration, time taken, and hours of the day.
For each drive, videos were recorded into multiple short files. To improve consistency, any video that does not have the corresponding phone data throughout the video duration or is not between 40 to 60 seconds (about 9% of all videos) is not considered in this experiment. The remaining 544 videos add up to 8.16 hours with an average duration of 54.1 seconds. The videos were taken between November 2018 and June 2019, covering Winter, Spring, and Summer. They also include multiple weather conditions, including clear, partly cloudy, overcast, and rainy, and various times of the day. The dataset also covers multiple driving scenarios, including highway, city street, and residential areas, mostly in Massachusetts and some in Connecticut and New York. Challenging scenarios that affect image quality include dirty windshield and glares from the sun during the day or other light sources at night. The statistics of the videos are shown in Figure 3-1 and some example frames are shown in Figure 3-2.

3.2 Phone Sensors

Today’s smartphones are equipped with multiple sensors that provide useful information about their position, orientation, and movement. The phone sensors used in this thesis include IMU (accelerometer and gyroscope) and GPS. Table 3.2 explains important features in the phone data. An example of phone data is shown in Table 3.3.

The IMU data were collected at 15 Hz and preprocessed to match the orientation of the car. The acceleration data were extracted into the longitudinal and lateral components, i.e., the ego-vehicle’s forward and right direction, and were smoothed with a window size of 1 s. The gyroscope data were also transformed to align with ego-vehicle’s frame of reference, where the $x$, $y$ and $z$ axes are in the forward, left, and up direction, respectively. The algorithmic details for phone data pre-processing are beyond the scope of this thesis.

The GPS data were collected at 1 Hz. Because raw GPS locations are noisy and do not necessarily match road segments, map-matching is performed as a pre-processing step by matching the GPS trajectories to road lines using real-world map information.

To match the phone data with the video frames, the phone data were linearly interpolated to 30 fps and matched with dashcam videos based on their timestamps. However,
Figure 3-2: Example frames from dashcam videos. The selection presents multiple scene types (highway, city street, and residential), multiple weather conditions, and challenging scenes, such as glare, reflection, low light, and dirty windshield. All frames are from the original footage without any pre-processing.

because the phone sensor data and dashcam videos were taken from different devices and the video’s timestamp is only precise to 1 s, the two streams of data are not always synchronized.
Table 3.2: Phone data attributes.

<table>
<thead>
<tr>
<th>Modality</th>
<th>Field</th>
<th>Explanation</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>time</td>
<td>Unix timestamp</td>
<td>ms</td>
</tr>
<tr>
<td>GPS</td>
<td>mm_lat</td>
<td>Map-matched latitude</td>
<td>deg</td>
</tr>
<tr>
<td></td>
<td>mm_lon</td>
<td>Map-matched longitude</td>
<td>deg</td>
</tr>
<tr>
<td></td>
<td>gps_speed</td>
<td>GPS speed</td>
<td>m/s</td>
</tr>
<tr>
<td></td>
<td>gps_heading</td>
<td>GPS heading</td>
<td>deg</td>
</tr>
<tr>
<td>Accelerometer</td>
<td>accel_lon</td>
<td>Longitudinal acceleration</td>
<td>m/s²</td>
</tr>
<tr>
<td></td>
<td>accel_lat</td>
<td>Lateral acceleration</td>
<td>m/s²</td>
</tr>
<tr>
<td>Gyroscope</td>
<td>rwx</td>
<td>Angular velocity (around x-axis)</td>
<td>rad/s</td>
</tr>
<tr>
<td></td>
<td>rwy</td>
<td>Angular velocity (around y-axis)</td>
<td>rad/s</td>
</tr>
<tr>
<td></td>
<td>rwz</td>
<td>Angular velocity (around z-axis)</td>
<td>rad/s</td>
</tr>
</tbody>
</table>

Table 3.3: Example of data from phone sensors.

<table>
<thead>
<tr>
<th>time</th>
<th>mm_lat</th>
<th>mm_lon</th>
<th>gps_speed</th>
<th>gps_heading</th>
<th>accel_lon</th>
<th>accel_lat</th>
<th>rwx</th>
<th>rwy</th>
<th>rwz</th>
</tr>
</thead>
<tbody>
<tr>
<td>1556486618046</td>
<td>42.480890</td>
<td>-71.222597</td>
<td>23.14</td>
<td>334.0</td>
<td>0.30</td>
<td>-0.39</td>
<td>0.00</td>
<td>0.02</td>
<td>-0.01</td>
</tr>
<tr>
<td>1556486618113</td>
<td>42.480903</td>
<td>-71.222606</td>
<td>23.14</td>
<td>334.0</td>
<td>0.26</td>
<td>-0.38</td>
<td>0.03</td>
<td>0.01</td>
<td>-0.05</td>
</tr>
<tr>
<td>1556486618179</td>
<td>42.480916</td>
<td>-71.222614</td>
<td>23.15</td>
<td>334.0</td>
<td>0.20</td>
<td>-0.38</td>
<td>-0.01</td>
<td>0.02</td>
<td>0.01</td>
</tr>
<tr>
<td>1556486618246</td>
<td>42.480930</td>
<td>-71.222622</td>
<td>23.15</td>
<td>334.0</td>
<td>0.13</td>
<td>-0.35</td>
<td>0.02</td>
<td>0.00</td>
<td>-0.03</td>
</tr>
<tr>
<td>1556486618313</td>
<td>42.480943</td>
<td>-71.222631</td>
<td>23.16</td>
<td>334.0</td>
<td>0.06</td>
<td>-0.34</td>
<td>0.00</td>
<td>0.00</td>
<td>-0.01</td>
</tr>
<tr>
<td>⋮</td>
<td>⋮</td>
<td>⋮</td>
<td>⋮</td>
<td>⋮</td>
<td>⋮</td>
<td>⋮</td>
<td>⋮</td>
<td>⋮</td>
<td>⋮</td>
</tr>
<tr>
<td>1556486673579</td>
<td>42.492975</td>
<td>-71.233895</td>
<td>27.72</td>
<td>323.7</td>
<td>0.35</td>
<td>0.47</td>
<td>0.04</td>
<td>-0.02</td>
<td>-0.02</td>
</tr>
</tbody>
</table>
Chapter 4

Perception

This chapter begins with the mathematical background related to perception tasks, followed by the design and discussion of each task, including fisheye rectification, camera calibration, lane detection, and object detection and tracking in 2D and 3D. The relationship between each task is summarized in the system overview diagram, as shown in Figure 4-1.

4.1 Background

In this section, I will review relevant mathematical concepts related to camera model and set up common notations for variables that are used in this thesis.

4.1.1 Vehicle and Camera Coordinate Systems

To understand position and orientation of vehicles in the environment, we are interested in the coordinate systems that are aligned with the road plane and the road direction. However, the camera is arbitrarily mounted in the ego-vehicle at a certain height and angle, which usually is not parallel to the road plane.

Let’s define two coordinate systems, vehicle coordinate system (VCS) and camera coordinate system (CCS). As shown in Figure 4-2, VCS is axis-aligned with the ego-vehicle and the road plane with its origin point on the road plane, below the camera. CCS is
Figure 4-1: Overview of the system.
Figure 4-2: The vehicle and camera coordinate systems.

axis-aligned with the camera’s orientation with its origin at the center of projection of the camera. The directions of the $x$-axis is to the right, $y$-axis down, and $z$-axis forward along the driving direction. The transformation between $x_v$ in VCS and $x_c$ in CCS is

$$
x_c = R x_v + t$$

$$\begin{bmatrix}
  x_c \\
  y_c \\
  z_c 
\end{bmatrix} =
\begin{bmatrix}
  r_{11} & r_{12} & r_{13} \\
  r_{21} & r_{22} & r_{23} \\
  r_{31} & r_{32} & r_{33} 
\end{bmatrix}
\begin{bmatrix}
  x_v \\
  y_v \\
  z_v 
\end{bmatrix} +
\begin{bmatrix}
  t_1 \\
  t_2 \\
  t_3 
\end{bmatrix}$$  \hspace{1cm} (4.1)

where $R$ is the rotation matrix and $t$ is the translation vector, representing VCS’s origin in CCS.

The rotation matrix can be represented by three Euler angles, i.e., three angles of rotations around the $x$, $y$, and $z$ axes. In this work, the order for the decomposition is chosen to be in the $x$, $y$, then $z$ order, with the three angles—pitch ($\gamma$), yaw ($\beta$), and roll ($\alpha$)—respectively. Therefore, the rotation matrix can be expressed as follows.

$$R = R_z(\alpha) R_y(\beta) R_x(\gamma)$$

$$\begin{bmatrix}
  \cos \alpha & -\sin \alpha & 0 \\
  \sin \alpha & \cos \alpha & 0 \\
  0 & 0 & 1 
\end{bmatrix}
\begin{bmatrix}
  \cos \beta & 0 & \sin \beta \\
  0 & 1 & 0 \\
  -\sin \beta & 0 & \cos \beta 
\end{bmatrix}
\begin{bmatrix}
  1 & 0 & 0 \\
  0 & \cos \gamma & -\sin \gamma \\
  0 & \sin \gamma & \cos \gamma 
\end{bmatrix}$$  \hspace{1cm} (4.2)
4.1.2 Pinhole Camera Model

The pinhole camera model projects a point in 3D in a straight line that passes through a focal point on to the focal plane of the camera. This projection is called perspective or rectilinear projection. A point with 3D coordinates $\mathbf{x}_c = Rx_v + t$ is mapped to a 2D coordinates $(u, v)$ via the transformation

$$s \cdot \mathbf{u} = K \mathbf{x}_c$$

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_u & 0 & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}$$

(4.3)

where $s$ is an arbitrary scaling factor and $K$ is the intrinsic matrix, defined by focal lengths of the camera, $f_u$ and $f_v$ along the $u$ (horizontal) and $v$ (vertical) axes, and the principal point’s coordinates $(c_u, c_v)$.

Substituting Equation (4.1) into Equation (4.3) yields the full equation as

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_u & 0 & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

(4.4)

where the intrinsic and extrinsic matrices together are defined as the projection matrix, $P = K[R \mid t]$.

For a fixed lens whose focal length does not change (including the dashcam used in this thesis), the intrinsic matrix $K$ only depends on the camera’s model and only has to be calibrated once. In contrast, $R$ and $t$ are referred to as the camera’s extrinsic parameters.

4.1.3 Fisheye Distortion Model

Many commodity dashcams today use fisheye lenses that provide a wide field of view to capture more of the environment around the cameras. However, fisheye lenses vio-
late perspective projection of the pinhole camera model and introduce radial distortion, which makes processing images and estimating 3D real-world coordinates from 2D pixel coordinates less straightforward.

A projection model can be described with a mapping function that maps from $f$, the focal length, and $\theta$, the angle from the principal axis, to $r$, the position of the object from the principal point.

For perspective projection, the mapping function is

$$ r = f \tan \theta $$

which satisfies the condition that every light ray travels in a straight line, passing through the center of projection (i.e., the pinhole). The mapping function for a fisheye lens can be approximated by a general polynomial [37], i.e.,

$$ r = f \cdot \theta_d = f \cdot \theta(1 + k_1 \theta^2 + k_2 \theta^4 + k_3 \theta^6 + k_4 \theta^8 + \ldots) $$

where $\{k_i\}_{i=1}^n$ are the distortion coefficients. This model gives a good approximation with $n = 5$ [37] or $n = 4$ [65].

If the values of distortion coefficients are known, fisheye image rectification (or fisheye distortion correction) remaps the points projected with fisheye projection to perspective projection. With the correction applied to all image frames, the corrected frames can be processed based on the pinhole camera model to detect lane markings, estimate distances to other vehicles, and perform other tasks.

### 4.1.4 Intrinsic Calibration and Fisheye Rectification

The intrinsic calibration process estimates the distortion coefficients $\{k_i\}$ and the intrinsic matrix $K$. For the same camera model with a fixed lens, these parameters are constant and the calibration only has to be done once. Given an access to the camera, the calibration process uses a calibration image (such as a chessboard pattern) with user inputs of multiple correspondences between image 2D coordinates and real-world 3D coordinates.
A discussion of the results of intrinsic calibration and fisheye rectification can be found in §4.2.

In this thesis, the calibration process could be carried out once as mentioned above. However, if a new camera is used and no access to the camera is available, the new distortion coefficients may need to be estimated on the fly. Recent works [65, 64] propose deep-learning based methods to learn distortion parameters from an image and may be used in such case.

4.1.5 Extrinsic Calibration

While the intrinsic calibration can be done once, the extrinsic parameters depend on the position and orientation of the camera with respect to the ego-vehicle, which vary between users and even for the same user on different drives.

The following subsections will focus on automatic calibration of extrinsic parameters, as the calibration problem that has to be performed for each video refers to only the extrinsic parameters, \( R \) and \( t \).

Assuming that the camera is centered in the car’s horizontal axis, the translation vector \( t \) only depends on the camera height \( h \). Consider Equation (4.1) for the camera’s center of projection: \( \mathbf{x}_c = \mathbf{0} \) and \( \mathbf{x}_v = \left[ \begin{array}{c} 0 \\ -h \end{array} \right] \) (because the \( y \)-axis points down) yields \( t = \mathbf{x}_c - Rx_v = -R \left[ \begin{array}{c} 0 \\ -h \end{array} \right] \). Therefore, the calibration of extrinsic parameters translates to estimating the camera angles \( \alpha, \beta, \gamma \), and the camera height \( h \).

4.2 Fisheye Rectification

As a preprocessing step for videos, each frame is rectified with the fisheye distortion parameters from intrinsic calibration. The calibration for the intrinsic matrix and distortion parameters were done using the OpenCV library [2], which estimates the distortion for \( k_1, k_2, k_3, k_4 \). At runtime, the remapping function can be pre-calculated and then applied to all video frames. Examples of the result are shown in Figure 4-3.

The rectification can be qualitatively verified by observing straight structures, such as lane markings, traffic signs, and building. In perspective projection (after rectification), a
Figure 4-3: Original and rectified frames of example frames. The first row shows the original frames. The second row shows the full rectified frames. The red and green rectangles show crops with the original and wide aspect ratios. The last row shows the final crops, which correspond to the green rectangle. Note that lines are straightened after rectification.
straight line should appear straight in the image. While the rectification results are not perfect, as seen in the traffic sign’s horizontal line near the top edge in the second column of Figure 4-3 that has a small curvature, the lane markings and other vertical structures do not suffer from this issue.

The rectification process also offers a choice for choosing the dimension of the output. In the second row of Figure 4-3, the red rectangles show the crop with the original dimension and aspect ratio (1280 × 720). This leads to a significant loss in field of view. The green rectangles show the wide crop that is used in further processing. The image resolution of 1640 × 590 is chosen to match the CULane dataset [49], but can be easily changed to match other datasets as well.

It is worth noting that some recent works also propose to perform 3D-related tasks on the fisheye images directly without rectification. For example, Kumar et al. [38] propose an algorithm for monocular distance estimation on fisheye images, arguing that rectification leads to loss of resolution in the areas near the image edge. However, performing rectification as a pre-processing step to all video frames is easier to understand and to perform subsequent tasks.

4.3 Camera Calibration

To understand real-world positions of other vehicles and objects captured on a dashcam video, the camera parameters must be known. The same pixel coordinates can infer different distance from the ego-vehicle if the camera’s position or orientation changes. Many 3D object detection algorithms for vehicles assume knowledge of camera parameters. Standard road datasets, such as KITTI [21], contain camera calibration parameters for all videos. This assumption that calibration values are known for applications related to autonomous vehicles is not surprising, because cameras can be carefully placed at specific positions and orientations by researchers and car manufacturers. However, commodity dashcams or smartphone cameras are often placed by drivers or car owners themselves to record the environments around them. Thus, it is not uncommon for the camera’s orientation of the same driver to vary across videos from different days, because the camera
can be easily moved upon touching. To address this issue, camera calibration is a crucial step for giving accurate localization related to other objects in the scene.

The estimation of camera’s extrinsic parameters can be done in two steps: Rotation Matrix and camera height. For both problems, I apply the widely used flat ground assumption.

### 4.3.1 Rotation Matrix Estimation

The rotation matrix can be represented with the three angles, pitch, yaw, and roll, around the $x$, $y$, and $z$ axes. To simplify the problem, I use the same assumption as in de Paula et al. [16] that the roll angle is zero, because the majority of dashcam videos have near-zero roll. The pitch ($\gamma$) and yaw ($\beta$) angles can be calculated from the *forward vanishing point* (FVP), where the lines parallel to the $z$ axis intersect at $z \to \infty$. Consider a point with $z \to \infty, x/z \to 0, y/z \to 0$. Equation (4.4) yields

$$\begin{bmatrix}
\cos \alpha \sin \beta \cos \gamma + \sin \alpha \sin \gamma \\
\sin \alpha \sin \beta \cos \gamma - \cos \alpha \sin \gamma \\
\cos \beta \cos \gamma
\end{bmatrix} = sK^{-1}\begin{bmatrix}u_z \\ v_z \\ 1\end{bmatrix}$$

where the left-hand side is the third column of the rotation matrix $R$ (Equation (4.2)), $s$ is the scaling factor that normalizes the vector on the right hand side to unit magnitude. With a known roll angle ($\alpha = 0$ in our case), the system of equations in Equation (4.7) can be solved for the yaw ($\beta$) and pitch ($\gamma$) angles. Note that there are only two independent equations to solve for two variables, excluding $s$.

The FVP can be found by finding the intersection of straight lines present in a video. In particular, I implement two approaches for finding the FVP, based on lane markings and keypoint tracks.

**Intersection of Lane Markings**

When the ego-vehicle is parallel to the lane, the lane markings intersect at the FVP. This method is simple and works well for ideal conditions, e.g. on a highway. The algorithm
Figure 4-4: Example images of the two camera calibration approaches in action. For both approaches, a forward vanishing point (FVP) estimate are shown as a yellow dot. The blue grid shows lines on the ground plane various distances. The images on the right panel show an inverse perspective mapping of the scene with the estimated calibration. (a) shows the lane-based approach, in which the intersections of the straight lane markings from multiple frames are used to estimate the FVP. (b) shows the keypoint-based approach. Lines of different colors show multiple tracks of keypoints in the image, generated by the ego-vehicle’s forward movement. The intersections of the tracks provide an estimate for the FVP.
for lane detection is discussed in §4.4.

The detected lane markings are fit with a straight line in the image coordinates, mapping from $v$ to $u$, i.e. vertical to horizontal coordinates. To avoid wrong intersection point such as when the road is curved, the markings are rejected if the line least-square fit error is larger than a minimum threshold.

While this approach works for a single image, the lane detection can be unstable, especially when the markings are not clear. Therefore, the lane lines are tracked and only used when every line moves less than a small distance for $n = 10$ consecutive frames. If they are stable, an average line of the past $n$ frames for each lane is used to finally calculate the intersection as the FVP. Figure 4-4a shows an example image of this approach.

**Intersection of Keypoint Tracks**

The second approach is based on Tummala et al. [58], which leverages the ego-vehicle’s motion to create virtual straight, parallel lines across video frames. When the ego-vehicle moves in a straight line, keypoints in the video (typically strong corners of objects in the scene) can be tracked and the intersection of those tracks then gives an estimation of the FVP. Compared to the first approach, this method works even when the lanes are not marked or the lane detection algorithm cannot find at least two stable lane lines, especially in the residential area. However, this method relies on having strong corners to track and therefore usually gives a less stable estimate than that of the first method. The accelerometer data from the phone is used to filter out the duration in which lateral acceleration is greater than a threshold (empirically, $0.5 \text{ m/s}^2$ is used). Figure 4-4b shows an example image of this approach.

For the implementation details, the keypoints are detected using the classic Shi-Tomasi corner detector [57] and are tracked with with a sparse iterative version of the Lucas-Kanade optical flow in pyramids [9]. I use the OpenCV library for these algorithms. Compared to the original method in Tummala et al. [58], the following changes are made to improve the results:

- **RANSAC-based intersection of multiple lines.** Each tracked keypoint is fitted with a line, and a number of those lines should intersect at the FVP. The original
work uses the centroid of the intersections of all pairs of lines, which can lead to many outliers given that some tracks are false positives. To avoid the issue, this work finds the best intersection point of multiple lines based on RANSAC (Random sample consensus). In each iteration, two lines are randomly selected to find its intersection. Then, each of the remaining lines is considered an inlier if the distance between the intersection point and the line is less than a threshold. Using only inlier lines, the best intersection point for those lines can be calculated using least square fit. The final best intersection from all iterations is the point which has the largest number of inliers. (Other criteria can also be used, e.g., the smallest least square fit residual.)

- **Aggregating FVP estimates with median.** In Tummala et al. [58], each FVP estimate is aggregated into the overall estimate with an exponential moving average. This method gives more weight to recent estimates than estimates from many frames ago. However, because the camera typically does not move within the duration of video, weights can be given equally to old and new estimates, making the final output less sensitive to estimation errors from the last few frames.

### 4.3.2 Camera Height Estimation

Estimating the camera’s height is less obvious, because it requires an extra knowledge that relates the pixel coordinates to real-world dimensions. In this thesis, I explore two methods, using the lane slopes and tracked keypoints.

**Lane Slopes with Known Lane Width**

Lane slopes can be used to find the camera height, given that the lane width is known. This approach works well with a highway, because the lane width is standard, (12 ft in the US). Errors in this method originate from incorrect lane width (e.g., large merging lane, small lane in city streets) and when there are no lane markings (in residential area). A larger lane width underestimates the camera height, and a smaller lane width leads to an overestimation. The advantage for this approach is its efficiency, as it only requires the
fitted line slopes without using the visual data again.

**Tracked Keypoints with Known Vehicle’s Speed**

Another approach presented in de Paula *et al.* [16] and Tummala *et al.* [58] is to track keypoints on the ground plane, assuming that the ego-vehicle’s speed is known. As discussed in [58], the speed provided by GPS typically lags when there is a large change of speed. Therefore, it works well when the speed is stable for a period of time.

The success of this approach also relies on having strong corners to track on the ground plane, such as dashed lane markings. Objects on the curb or shadows on the ground can also be used for tracking. Under ideal conditions (accurate speed measurement, strong corners to track), this can give a better height estimate, especially for residential areas where lanes are not marked or the lane width varies from the highway standard.

The mathematical details of this approach can be found in [16, 58]. In terms of implementation, I use the same implementation as the keypoint-based approach for estimating the FVP with some modification. While [16, 58] explicitly track dashed lane markings, the lanes are typically not marked, especially in the residential area. Because relying on lane markings may lead to failure to obtain a height estimate, I extend the region to track keypoints to a triangular area with FVP as a top vertex. Additionally, any area corresponding to another detected vehicle (from object detection in §4.5) is removed. Keypoints detected on another vehicle will lead to an incorrect estimation, as they are higher from the ground plane and their speed are not zero if the vehicle was moving.

**4.3.3 Results and Discussion**

The two approaches to find camera angles and height, *i.e.*, lane-based and keypoint-based, are experimented with 544 videos from one user. To verify the correctness of the two approaches, the values are plotted against each other as shown in Figure 4-5. Each data point on the plot represents a video where both approaches yield an answer. For the plot of the camera height, outliers whose value in either axis is below the 0.01 percentile or
Figure 4-5: Correlation plots for camera extrinsic calibration results from two approaches. Each row corresponds to pitch, yaw, and height, respectively. For each plot, the horizontal axis represents the lane-based approach and the vertical axis represents the keypoint-based approach. Each point corresponds to a single video. The color of each point is based on the time that the video was taken in the first column and the average speed in the second column. (The data points in the left and the right columns are the same.)
above the 0.99 percentile are removed, as they correspond to very small or large values that are physically impossible. With the conditions above, the pitch and yaw plots contain 345 data points and the height plot contains 310.

The plots for pitch and yaw (the first two rows) show a very promising result. The two approaches have a near perfect correlation with slopes very close to 1. The plots in the first column, where the points are colored based on the time taken, display different clusters that move over time, indicating that the driver moved the camera over the duration of several months. In contrast, the camera height estimates from the two approaches show no correlation at all. However, this does not mean that both approaches fail, but the estimates are subject to large errors. The left plot (colored by time) shows a small cluster for December 2018 in the lower left area. The cluster corresponds to a low camera height, which is confirmed by visually inspecting the videos. The remaining videos establish a large cluster centered around 1.25 m with noise in both axes. The reason was that the camera was mounted at a fixed height, so noises from the two approaches lead to no correlation instead of high correlation. The right plot (colored by average speed in each video) shows a red cluster which represents high speed, i.e., driving on a highway. The cluster has a relatively small horizontal variation, meaning that the lane-based approach works well with highways where the lane markings are clear and the lane width is standard. However, the keypoint-based approach has a high large error variance, which may be improved by better tracking of keypoints and better measurement of the ego-vehicle’s speed.

There are multiple ways to improve the calibration process. The two approaches can be combined, for example, using the lane-based approach to estimate the camera angles and the keypoint-based approach for the camera height, especially if the lane width is unknown but the speed measurement is reliable. Moreover, because a longer video can give a better estimate, multiple consecutive videos that were taken from the same drive may be used to perform calibration together. Particularly, using multiple videos can largely decrease the large variance in height estimates. If the camera height is known to be fixed, e.g., the user used a fixed mount for the camera, a global average (weighted towards videos with higher ego-vehicle’s speeds) may decrease errors in subsequent tasks.
4.4 Lane Detection

Lane markings are important because they set trajectories for drivers to follow. For the perception module, detected lane markings are used in two steps. First, straight lane markings can be used to for calibration during the duration that the ego-vehicle moves parallel in the lane, as discussed in §4.3. The intersection point of the lane markings approximates the Forward Vanishing Point, and the slopes can be used to estimate the camera’s height given known vehicle speed. Then, when the camera’s projection matrix is calibrated, the lane markings’ coordinates can be transformed to the VCS, i.e. real-world coordinates. Each lane marking is typically processed by fitting a polynomial or spline curve in the inverse perspective mapping (IPM) or bird’s eye view.

Some approaches perform lane detection using the IPM because the lane markings have better structures, i.e., the lane markings are mostly vertical rather than slanted at an angle. For example, DVCNN [28] presents a dual-view CNN that uses both the original and IPM images. However, transforming the original image to the IPM requires the camera’s projection matrix, which needs to be known beforehand or estimated within the network. For better interpretability, this work adopts a lane detection algorithm that only uses the original image as an input. Once the lane markings are used to explicitly estimate the camera calibration parameters, the lane coordinates can then be transformed to the IPM and fit with curves, similar to typical approaches.

Among approaches that perform lane detection in the original image space, I use a model from Hou et al. [31] thanks to its fast inference time and reasonable robustness. The model outputs a single-channel segmentation mask image for each of the $n = 4$ lanes and a vector of size $n$, corresponding to the confidence of each lane’s existence. The model uses an ERFNet backbone [56] and were trained on the CULane dataset [49].

To perform lane detection in videos, the algorithm is applied to each video frame individually. As expected, it works well in when the lane markings are clear. An interesting result to note is that it sometimes predicts the lanes even when they are not marked, such as in a residential area. This is because 11.7% of the training data, the CULane dataset [49], are images with no line, each of which contains annotations for where the lane markings
Figure 4-6: Architecture for 2D object detection and tracking. Given an input frame $t$, Mask R-CNN finds object bounding boxes and masks. The bounding boxes are feed into DeepSORT for association, matching the same object across frames. When a new track is first created, the track’s state is tentative (shown as a yellow box in the last column). Once a track that has been matched in at least some number of consecutive frames, it becomes a confirmed track (a green box). Re3, a visual object tracker, is used to track objects belonging to confirmed tracks. Its output is added to Mask R-CNN proposals to avoid missed detections.

The lane detection algorithm can be further improved in several ways. In addition to improving the detection accuracy itself, an extension to classify each marking’s color (yellow or white) and type (dashed or solid or two lines) will improve understanding of driving behaviors, such as detecting events in which a driver does not follow the markings’ rules. Another extension is to detect other pavement markings, such as stop lines, arrows, or a bicycle lane symbol.

4.5 Object Detection and Tracking

In this section, I discuss the system design and challenges for detecting and tracking objects in 2D image space and the conversion of detections from 2D into 3D world space.
4.5.1 2D Object Detection and Tracking

The fundamental tasks for understanding location and movement of vehicles, pedestrians, and other objects in a video are object detection and multi-object tracking in 2D. Given a series of video frames as an input, the task is to detect each object with a bounding box, from \((u_1, v_1)\) to \((u_2, v_2)\), in the image space in each frame and to assign the same unique identifier for the same object that appears in multiple frames.

The overall pipeline for 2D object detection and tracking is shown in Figure 4-6. As discussed in the related work section for multi-object tracking (MOT) (§2.2.3), this work uses the typical two-shot architecture, in which the detection stage is performed first, then the tracking stage associates detected objects into multiple tracks across frames. This makes it possible to use an object detector trained on a large image dataset without learning from video data. For the purpose of dashcam videos, I use Mask R-CNN [29] trained on the COCO dataset [40] as the object detector, with an additional output of mask for each object (instance segmentation). The trained model is modified to limit its outputs to categories of interest—including multiple types of vehicles (car, truck, bus, train, motorcycle, bicycle), person, stop sign, and traffic light—and ignore irrelevant categories.

For the tracking (association) stage, DeepSORT [62] is used to match detections in the new frame with existing tracks, by keeping an internal state for each track based on its position, motion, and visual appearance. At a given time, each track has one of the three possible states: tentative, confirmed, or deleted. If a new detection is not matched with any existing track, a new track is initiated with a tentative state. Once an existing track is matched with detections for a fixed number of consecutive frames \((n = 5)\) is used), the track becomes confirmed. If a confirmed track is not matched for a number of consecutive frames, the track is deleted. Therefore, detections belonging to tentative tracks that are never confirmed are likely false positive detections that can be rejected, which improves the overall precision.

Additionally, the temporal properties of videos can also be leveraged to improve recall, as discussed in the related work section for video object detection (§2.2.2). However, typical models are trained on different types of datasets. To retrieve detections that have
been assigned low confidence scores by the object detector, an algorithm for visual object tracking can be used instead. Specifically, Re3 [26] is an LSTM-based network that predicts a bounding box of an object in the current frame, given a bounding box in the previous frame based on location and visual appearance. Because a visual tracker requires an initial bounding box to start tracking, Re3 is initiated from confirmed tracks, not tentative ones. As Re3 predicts a bounding box without a mask, each Re3 detected bounding box can be fed into the mask branch of Mask R-CNN, in addition to the original boxes from the region proposal network (RPN) to obtain a segmentation mask.

Figure 4-7 shows example outputs of 2D object detection and tracking on a sequence of video frames. With an experiment from our dataset, about 12% of all detections are proposed by Re3, i.e. Mask R-CNN at the selected confidence threshold (0.8) misses these objects. However, a small percentage of these Re3 tracked detections are incorrect. A typical failure case for the tracker happens when the tracker picks a wrong object for one frame, which causes the tendency to continue tracking the wrong target for the subsequent frames. This type of errors is reduced by enforcing the area of the mask within the bounding box, as predicted by Mask R-CNN, to be at least some minimum threshold. Additionally, the mask is be used to refine the proposed bounding box by limiting the bounding box to tightly enclose the mask.

### 4.5.2 3D Object Detection and Tracking

Understanding positions and motion of vehicles in the scene requires real-world coordinates in the vehicle coordinate system (VCS). With the camera’s parameters from calibration, the 2D bounding boxes can be transformed to locations in the ground plane. To detect objects in road scenes, such as vehicles and pedestrians, this work applies the widely used flat ground assumption, i.e., objects are situated on the same plane as the ego-vehicle and 3D bounding boxes are parameterized to be parallel to the ground.
Detections only

With tracking

Figure 4-7: Example results from 2D object detection and tracking across multiple frames. Similar to Figure 4-6, each row correspond to a video frame. The first column shows Mask R-CNN detections with their confidence scores (blue) and tracked detections from Re3 (green). The second column shows results with tracking based on DeepSORT, including tentative tracks (yellow) and confirmed tracks (green). In all frames, the segmentation masks are overlaid in white. The images are cropped from the full field of view for better visualization.
Figure 4-8: An example frame for 3D object detection of vehicles. In the main view, a 3D bounding box is shown for each vehicle together with its track identifier. The right panel shows an inverse perspective mapping of the scene, displaying 2D bounding box for each vehicle on the ground plane.

**Following distance estimation**

When the vehicle of interest is directly in front of the ego-vehicle, the following distance to that particular vehicle can be directly estimated. Assuming that the roll and yaw angles are close to zero, the middle point of the 2D bounding box’s bottom edge, \((u_0, v_0)\), corresponds to the point on the ground, \(i.e., y = 0\), that is aligned with the next vehicle’s rear bumper. Therefore, Equation (4.4) yields

\[
\begin{bmatrix}
    s & u \\
    v & 1 \\
\end{bmatrix} = \begin{bmatrix}
    p_{11} & p_{12} & p_{13} & p_{14} \\
    p_{21} & p_{22} & p_{23} & p_{24} \\
    p_{31} & p_{32} & p_{33} & p_{34} \\
\end{bmatrix} \begin{bmatrix}
    x \\
    0 \\
    z \\
    1 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
    x \\
    z \\
    1 \\
\end{bmatrix} = s \begin{bmatrix}
    p_{11} & p_{13} & p_{14} \\
    p_{21} & p_{23} & p_{24} \\
    p_{31} & p_{33} & p_{34} \\
\end{bmatrix}^{-1} \begin{bmatrix}
    u \\
    v \\
    1 \\
\end{bmatrix}
\]

(4.8)

where \(z\) is the following distance to the next vehicle.

**3D Bounding Boxes**

For other vehicles that are not directly in front of the ego-vehicle, each 3D bounding box’s center is not aligned with the 2D center. Therefore, the direct homography approach
cannot be used.

I apply the algorithm by Mousavian et al. [45] to convert 2D into 3D boxes. The algorithm uses a 2D CNN that estimates each object’s size and orientation and solves for each 3D bounding box’s center position by assuming the 2D-3D bounding box consistency constraint, i.e., each side of the 2D box should contain at least one projection of the 3D box corners. However, directly applying the 2D-to-3D conversion leads to temporal inconsistency for each object’s 3D bounding box over time. The 3D Kalman Filter is applied to smooth the track, as presented by Hu et al. [33]. An example of 3D detection outputs is shown in Figure 4-8.
Chapter 5

Analyzing Behaviors

Dashcam videos provide visual information for analyzing driving behaviors that are not available with only phone’s inertial or GPS sensors. Common dangerous driving behaviors include hard braking, tailgating, and speeding. While smartphone sensors alone can identify hard braking events, they cannot indicate the causes of the event, e.g., whether the driver reacted to a sudden event or was just inattentive. Moreover, tailgating cannot be detected by smartphone sensors alone without vision or depth information. Finally, speeding compared to the speed limit alone does not provide a full picture of the driver’s speed compared to the traffic flow, which can be more significant in practical scenarios. For each behavior, I discuss how dashcam videos are used to identify its occurrences or to provide context into understanding its causes, and what challenges arise in analyzing each particular behavior.

5.1 Hard Braking

A hard braking event occurs when a driver suddenly slows down or stops with high deceleration (large negative acceleration). It is often an indicator that the driver was not paying attention or a sudden event occurred on the road. While hard braking events can be detected from smartphone’s acceleration data, dashcam videos can provide contextual insights into the causes of the events and illustrate the varying severity and likelihood of each cause.
5.1.1 Event Classification: Why did the drivers hit the brake?

From our dashcam video data, the causes of hard braking can be grouped into the following categories:

1. **The car in front in the ego-lane** (CAR_EGO_LANE), i.e., when it slows down or stops for any reason. This leads to the question of how early the driver reacted to the next vehicle, which is discussed in §5.1.2.

2. **Stop sign** (STOP_SIGN). This also extends to an analysis of whether the ego-vehicle came to a complete stop in §5.1.3.

3. **Traffic light** (TRAFFIC_LIGHT). This is very similar to the stop sign case. However, running red lights is not found in our dataset.

4. **Pedestrian crossing** (PEDXING). A detected person in the scene is only considered if their position actually crossed the current lane to avoid counting irrelevant detections, such as people jogging on the sidewalk.

5. **Another vehicle in the scene** (CAR_OTHERS). Another vehicle not originally in the same lane can also cause the driver to brake, such as when they turned or merged into the lane or when they backed up from their house.

6. **Self-braking** (SELF). If no causes above are detected, it is likely that the braking event is caused by the driver and should have been preventable if they were more attentive. From our dataset, this type of events typically occurred when the driver *almost missed* something, such as missing a turn to a small street from the main road or missing a highway exit, then abruptly slowing down to turn or change lane.

Even though each event can be caused by multiple reasons simultaneously, such as stop sign and pedestrian, in this work the problem is simplified as a one-way classification task. If at least two possible causes are detected, the priority is given to the earlier item on the previously stated list.

To test the algorithm, hard braking events are extracted from the dataset by considering the longitudinal acceleration data (accel1_lon) with \( a_z \leq -0.3g \) where \( g = 9.8 \text{ m/s}^2 \). To
Figure 5-1: Examples of video frames for hard braking event classification. (a) - (f) show events that are classified correctly for all categories. (g) - (j) show challenging scenes that the algorithm misclassified. In each image (zoom in for details), a bounding box is drawn for each detected object and is marked with different colors. Green represents a vehicle or person in the ego-lane, blue for those in a neighboring lane, yellow for a stop sign or a traffic light, and white for any other detection. For a green or blue rectangle, the \((x_v, z_v)\) position is labeled in addition to the object category and track id.
provide more context for the algorithm, the previous 3 seconds and following 3 seconds of the event are also used for the visual analysis, which helps avoid the synchronization issue between the phone data and video frames. If an event’s starting time (when \(a_z\) becomes lower than the threshold) is within 3 seconds of the previous event’s ending, it is merged with the previous event and considered a single event.

From the 86 extracted events, 66 events are classified correctly. The confusion matrix of the classification result is shown in Figure 5-2. Examples of video frames from different event categories are also shown in Figure 5-1. (a) - (f) show correctly classified events, while (g) - (j) show failure cases of the algorithm. The explanation of the mistake in each challenging scenario is written in the caption of each image.

### 5.1.2 Following distance for hard braking events

Figure 5-3 shows the following distance vs. time when the driver reacted to the next vehicle by braking suddenly. The distance is smoothed by an averaging filter with a window size of 0.5 s. The time is offset to when the following distance becomes less than or equal to 15 m or when the next vehicle was detected, whichever is later. To make the plot easier to interpret, it only shows events in which the next driver was stopping, e.g. at an inter-
Figure 5-3: Following distance vs. time for hard braking when following another vehicle. Time is relative to when the following distance becomes less than or equal to 15 m or when the vehicle in front was detected.

section, filtering out those in which the next vehicle was still moving (such as slowing down) or started moving while the ego-vehicle was slowing down.

Most lines follow an expected trend; the following distance decreases and plateaus to a stopping distance. The flat lines correspond to events with challenging lighting conditions, because the next vehicle was not detected until it was really close to the ego-vehicle. Other error sources of distance estimation include imperfect camera calibration and bounding box positions.

To compare different braking events shown in the plot with one another, each line is colored according to the minimum longitudinal acceleration \(a_z\) of the braking event, where red represent a larger magnitude. While an event with a large deceleration magnitude is expected to display a generally smaller stopping distance, this trend is not apparent from a small subset of data shown here. Analyses with more data points may show such trend and give a better understanding of the driver’s reaction time and attentiveness.
5.1.3 Running stop signs

For hard braking events that are caused by stop signs, the question of whether the driver came to a complete stop naturally arises. Figure 5-4 shows plots related to stop sign events. The first plot, the histogram of minimum speed, shows that for many events the speed was never less than 2 m/s. (The median is 2.18 m/s from 31 events.) To compare all events in the time domain, the second plot shows speed vs. time (relative to when speed reached the minimum). For some events (towards the blue color), the speed decreased and stayed at close to zero for a few seconds, which means the driver stopped and waited to proceed. For others (towards the red color), the speeds at 3 seconds before and after the minimum are generally higher. However, the caveat of using GPS speed is that it is not very accurate for a short period of time, especially when there is a large variation in speed. The GPS speed was originally sampled at 1 Hz. Nevertheless, upon investigating videos with large minimum speeds, the driver in fact did not come to a complete stop (which is common among many drivers).

The accelerometer data were collected at a higher sampling rate (15 Hz) and provide another perspective for analyzing the events. The third plot shows longitudinal acceleration ($a_z$) vs. time, where the time is relative to when $a_z$ first becomes 0 from a negative value. The color of each event is the same as the second plot, i.e. the blue lines correspond to events with lower minimum speed than the red lines. As expected, the blue lines are generally closer to the $a_z = 0$ line after $t = 0$, i.e., the driver stopped (with some noise in the acceleration data) or accelerated more slowly after slowing down, compared to most of the lines towards red. (There is one green line that has exactly zero acceleration for all $t \geq 0$, but the curve is non-smooth. It corresponds to invalid data.)

This simple analysis for stop sign events can be further improved in multiple ways. First, the time duration in which the driver completely stopped may have been instantaneous, making it difficult to analyze using the speed from GPS alone. Visual odometry, i.e., inferring the ego-vehicle’s motion from videos, may provide more insights. Additionally, dashcam videos can be used to estimate the distance that the driver stopped (or slowed down) compared to the stop line, the stop sign, or the intersection. However, this is non-
Figure 5-4: Plots for stop sign braking events. The first plot shows the histogram of minimum speed. The second plot shows speed vs. time, where the time of each event is offset to zero when the speed is minimum. The color reflects the minimum speed. The third plot shows longitudinal acceleration ($a_z$) vs. time, where the time is offset to zero when $a_z$ first becomes 0. The color of each event is the same as in the second plot.
trivial without accurate monocular depth estimation or detection of stop lines, which is separate from typical lane detection.

5.2 Tailgating

Tailgating is when a driver drives behind another vehicle without leaving enough distance to stop, potentially causing a rear-end collision if the vehicle in front suddenly stops or decelerates. A tailgating event can be identified when the distance to the next vehicle is lower than a threshold, especially over a long period of time. The higher the vehicle’s speed is, the greater the following distance should be. A well-known simple guideline is called the *three-second rule* [48], which states that a driver should stay at least three seconds behind the vehicle in front. The following time should be even longer than three seconds for dangerous situations, such as during hazardous weather or on downhill slopes.

To identify potential tailgating events, the following analysis is performed on a subset of 280 videos in which the speed limit information is valid and is relatively high, ranging from 11 m/s (25 mph) to 27 m/s (60 mph). Upon detecting a vehicle in the ego-lane for at least 3 seconds, the following distance is estimated as described in §4.5.2) using the lane-based calibration approach. For each of the 224 detected events, the average following distance/time and the average speed is calculated and plotted against each other, as shown in Figure 5-5. Note that events with an average speed less than 10 m/s or an average following distance greater than 100 m are excluded, as they give unreliable estimates and are not significant for analyzing tailgating events.

To visualize the differences between the events, the marker size represents each event’s duration (ranging from 3 s to 55 s) and the color represents the average speed relative to speed limit during the event. The gray line in each plot corresponds to the three-second rule. Data points below the line indicate that the following distance/time to the next vehicle is closer than 3 seconds, thus violating the guideline.

The plot shows 186 occurrences of the three-second rule’s violation (139 with the threshold of two seconds). Events with a short following time over a long duration at high speed—represented by large points towards the bottom right corner of each plot—
Figure 5-5: Scatter plots of following distance/time to the next vehicle vs. average speed, showing potential tailgating events. Each data point corresponds to an event where a vehicle is detected in the ego-lane. The value is plotted according to the average speed and the average following distance/time during the event. The marker size represents the duration of the event (as shown in the legend) and the color represents the speed relative to the speed limit. The gray line in each plot shows the three-second rule.
are particularly dangerous. In particular, these events also correspond to speeding, with up to 7 m/s (16 mph) over the speed limit. Note that there are also instances when the driver was following safely with sufficiently large following distances. However, when the distance was over about 100 m, the next vehicle typically appeared too small to detect and track using dashcam videos. Regardless, such events are not as important to identify as the events with short following distance that can be detected as explained above.

5.3 Speeding

As one of the most common traffic violations, speeding is dangerous and is involved in over 26% of 37,133 traffic fatalities in 2017 in the US [19]. While driving faster than the speed limit is generally considered dangerous, speed limits are not always the best indicator of risk. In practical scenarios, the vehicle’s speed relative to other vehicles may be more important. For example, if the traffic is dense and significantly slower than the limit, driving just under the speed limit is too fast and can pose risks to other vehicles. Driving too fast or too slow compared to the traffic flow makes it more difficult for other drivers to anticipate or expect swift changes and is thus riskier for all parties involved.

While a GPS sensor on a smartphone can provide an estimated speed of the ego-vehicle, it cannot indicate the speed of other vehicles on the road. Therefore, dashcam videos can be used for calculation of speeds of other vehicles in the traffic.

In the following discussion, traffic flow refers to an average speed of other vehicles detected during the video. An analysis was performed on a subset of videos taken on highways and city streets, with speed limit ranging from 11 m/s (25 mph) to 27 m/s (60 mph). For each video, the position, orientation, and speed of vehicles in the scene are estimated, as explained in §4.5.2. The vehicles that are driving in the same direction and are tracked for at least 3 seconds are considered in the calculation. With at least one vehicle in the scene, information from 129 videos are shown in Figure 5-6. The scatter plot shows the ego-vehicle’s average speed relative to the traffic flow vs. its average speed over the duration of each video. A data point below the horizontal axis \( y = 0 \) indicates that the ego-vehicle’s speed is slower than the traffic flow, while a data point above means the
Figure 5-6: Scatter plots of the ego-vehicle’s speed relative to the traffic flow vs. its speed. Each data point corresponds to a video on where at least one other vehicle driving in the same direction are detected. The horizontal axis represents the ego-vehicle’s speed averaged over the duration of the video. The vertical axis represents its speed compared to the traffic flow, as computed by relative speeds of other vehicles. The marker size represents the number of detected cars and the color represents the speed relative to the speed limit.
opposite. Therefore, a large deviation in either direction is a risk factor that should be taken into account, in addition to the speed relative to the limit itself. Note that a data point with a large marker size indicates a larger number of tracked vehicles, leading to a more reliable estimate of the traffic flow. The number of tracked vehicles range from 1 to 14, with an average of 3.9 vehicles per video.

To gain more insights, this analysis can be improved and extended in multiple ways. First, the number of vehicles detected in the duration of each video is quite small, which may lead to an unreliable estimate of the actual traffic flow. Multiple videos can be used together to provide a better understanding of the environment, especially because driving on a highway usually takes tens of minutes to hours. Additionally, other factors that affect evaluation of driving risks can also be taken into account. For example, the position of the ego-vehicle’s in the lane, i.e. whether the driver is in the left or right lane, dictates how the driver should drive relative to other vehicles. Driving slow in the left lane can be risky to other drivers trying to use the left lane to pass. Lastly, the analysis can be extended to include driving scenarios in slower traffic. However, performing such analysis for city streets is significantly more complex than for highway scenarios. With many traffic lights and stop signs, driving speed can vary by a large magnitude and the traffic flow becomes less clearly defined.
Chapter 6

Conclusion and Further Discussion

In this thesis, I present the system that identifies and analyzes dangerous driving behaviors using monocular dashcam videos and data from smartphone sensors. Extracting useful information from real-world driving videos requires many computer vision tasks, including fisheye rectification, camera calibration, lane detection, and object detection and tracking. The processed information of the ego-vehicle’s position and movement, as well as those of other vehicles in the scene, can then be used to identify risk behaviors, including tailgating and running stop signs, calculate the speed of traffic flow compared to the ego-vehicle’s speed, and classify the cause for hard braking events, such as being inattentive, stopping for a pedestrian, reacting to the next car in the lane.

There are many possible next steps to improve the system. First, the algorithms can be further improved, both in terms of accuracy and speed. In addition to improving the algorithm for each perception task individually, many of the tasks are relevant to each other and can benefit from learning jointly for multiple tasks. For example, video object detection and tracking may be performed in a single shot, and estimation of depth and ego-motion from a video may also include a network for estimating the optical flow [51]. Because the model for each task typically starts with a similar Convolutional Neural Network backbone that encodes an image into features, the problem can be formulated as multi-task learning using a shared backbone and outputting predictions for many tasks [4]. However, while this approach may provide a faster and more accurate result, training such a large network requires large computational resources and datasets and may be
more challenging to reach an optimum.

Second, the datasets for training and testing the system can be expanded. In recent years, the number of large-scale open datasets from research groups and companies around the world has been increasing rapidly, such as nuScenes [10], ApolloScape [34], and BDD100K [67]. Multiple datasets may be used together to train various computer vision tasks. However, dashcam videos collected from users in our dataset also have unique challenges and do not come with a full suite of sensors like most datasets. Therefore, labeling our video dataset in a subset of frames maybe helpful. The papers presenting open datasets also typically offer insights into efficient data annotation, which can be helpful to our data labeling process in the future.

Moreover, additional sources of information may be helpful for improving perception tasks and analyzing behaviors. For example, given the GPS coordinates, map information may provide useful hints, such as the type of roads and areas, the expected number of lanes, and the road topology. With dual-camera dashcams, inward facing cameras can also be used to identify whether the driver is inattentive or drowsy. Therefore, there are endless sources of information that can be leveraged to improve the system.

Finally, more testing data is needed, especially for dangerous driving behaviors and borderline events that may fool the system. Nevertheless, performing dangerous driving in the normal driving scenarios would be unreasonable. Thus, collecting data in a controlled environments, similar to [66, 6], can be beneficial to improving the robustness of the system.
Bibliography


[2] Opencv: Fisheye camera model. URL: https://docs.opencv.org/4.3.0/db/d58/group__calib3d__fisheye.html. (Cited on page 38.)


