UNIVERSITY OF CALIFORNIA SAN DIEGO

Intelligent Human-in-the-Loop Vehicular Automation with Real-Time Vision Models

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Electrical Engineering
(Intelligent Systems, Robotics & Control)

by

Akshay Rangesh

Committee in charge:

Professor Mohan M. Trivedi, Chair
Professor Manmohan Chandraker
Professor Garrison W. Cottrell
Professor Kenneth Kreutz-Delgado
Professor Bhaskar D. Rao

2021

The dissertation of Akshay Rangesh is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2021

DEDICATION

To my mother Gayathri,

for giving me all her time and expecting none in return.

EPIGRAPH

We shape our tools,

thereafter our tools shape us.

—John Culkin & Marshall McLuhan

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

LIST OF ABBREVIATIONS

ACF Aggregate Channel Features

ADAS Advanced Driver Assistance Systems

AN Association Nodes

AOS Average Orientation Similarity

AP Average Precision

BCE Binary Cross Entropy

CA Constant Acceleration

CAM Class Activation Map

CDS Controlled Data Study

CE Cross-Entropy

CNN Convolutional Neural Network

CPU Central Processing Unit

CTRV Constant Turn Rate and Velocity

CV Constant Velocity

CWS Current Window Size

CycleGAN Cycle-Consistent Generative Adversarial Network

DLA Deep Layer Aggregation

DN Association Nodes

DoF Degrees of Freedom

DTW Dynamic Time Warping

EKF Extended Kalman Filter

FB Forward-Backward

FC Fully Connected

FoV Field of View

FN False Negative

FP False Positive

FPCA Functional Principal Component Analysis

FPN Feature Pyramid Network

FRAG Fragmentation

FSA Forced Spatial Attention

GAN Generative Adversarial Network

GAP Global Average Pooling

GMM Gaussian Mixture Model

GNN Graph Neural Network

GPCycleGAN Gaze Preserving Cycle-Consistent Generative Adversarial Network

GPP Ground Plane Polling

GPS Global Positioning System

GPU Graphics Processing Unit

GradCAM Gradient-Weighted Class Activation Map

GRU Gated Recurrent Unit

HAF Horizontal Affinity Fields

HDA Hierarchical Deep Aggregation

HMM Hidden Markov Model

IDA Iterative Deep Aggregation

IDS Identity Switches

ID LSTM Independent Long Short Term Memory

IMM Interacting Multiple Model

IMU Inertial Measurement Unit

IOHMM Input-Output Hidden Markov Model

IoU Intersection over Union

IPM Inverse Perspective Matching

IR Infrared

KF Kalman Filter

LCSS Least Common Subsequence

LED Light-Emitting Diode

LiDAR Light Detection and Ranging

LILO Looking-In and Looking-Out

LSTM Long Short Term Memory

M^{3}OT Multi-Perspective, Multi-Modal, Multi-Object Tracking

MAE Mean Absolute Error

MDP Markov Decision Process

MHT Multi-Hypothesis Tracking

ML Mostly Lost

MOT Multi-Object Tracking

MOTA Multi-Object Tracking Accuracy

MOTP Multi-Object Tracking Precision

MPNN Message Passing Neural Network

MSE Mean Squared Error

MT Mostly Tracked

NCC Normalized Cross-Correlation

NIR Near-Infrared

NMS Non-Maximum Suppression

OS Orientation Score

PCA Principal Component Analysis

QPBO Quadratic Pseudo-Boolean Optimization

RADAR Radio Detection and Ranging

RANSAC Random Sample Consensus

ReLU Rectified Linear Unit

RGB Red-Green-Blue (Color)

RGB-D Red-Green-Blue-Depth (Color and Depth)

RNN Recurrent Neural Network

ROC Receiver Operating Characteristics

RoI Region of Interest

RPN Region Proposal Network

RRC Recurrent Rolling Convolutions

RWS Retained Window Size

SAD Self-Attention Distillation

SAE Society of Automotive Engineering

SGD Stochastic Gradient Descent

SGM Semi-Global Matching

SVM Support Vector Machine

TOR Take-Over Request

TOT Take-Over Time

TN True Negative

TP True Positive

TTC Time to Collision

UKF Unscented Kalman Filter

VAF Vertical Affinity Fields

VGMM Variational Gaussian Mixture Model

VIM Vehicle Interaction Module

ACKNOWLEDGEMENTS

Nothing of significance was ever achieved by an individual acting alone...

Though this dissertation appears to be authored by me on the surface, in reality, it has been a joint endeavor from the beginning. It would be very remiss of me to not take this opportunity to thank those who played a part.

Nothing in life would be achievable for me if not for the love the and support of my family. They have been there for me from the start, and I shall strive to do the same for them in the future. My parents - Gayathri and Rangesh - have sacrificed so much to ensure the best for me, and for this I will forever be grateful. My mother, Gayathri, is the strongest person I know of. She reads, writes, paints, educates, listens, cooks, works, and supports all of us with a smile on her face. She is my idol and role model. I cannot offer her enough thanks but will try nonetheless. My brother, Anirudh, is one of the nicest people I know. He has always stayed in touch and kept my spirits up, even when I bothered not to. I wish him the very best, and look forward to spending more time with him in the coming years. Finally, I would also like to thank the latest addition to our family - my girlfriend Vishakha. Vishakha and I have been friends for many years now, and the more I get to know her, the more I admire her. She has supported me through thick and thin, and has always helped me be better and do better. I cannot wait for the lifetime of hikes, backpacking trips, and board-game nights we have ahead of us.

My advisor, Prof. Mohan Trivedi, is one of the most influential persons in my life. Over the course of my graduate studies, he has been a teacher, mentor, fellow researcher, friend, well-wisher and so much more. His work ethic, enthusiasm for research, and affable nature are qualities that I admire and hope to replicate. He remembers the names of all students enrolled in his courses, takes interest in their lives and well-being, writes countless recommendation letters, and constantly motivates those in need. It is my genuine belief that the best antidote to imposter syndrome is a brief meeting with Prof. Trivedi. I thank him for all of this and more.

I would also like to thank my dissertation committee - Prof. Chandraker, Prof. Cottrell,

Prof. Kreutz-Delgado and Prof. Rao - for being a treasure trove of wisdom and knowledge. Though our interactions were brief (and occasionally tense), they have had an immense impact on the way I think and do research. Additionally, the courses they taught form the bedrock upon which I build all my work.

Research is expensive, and experimental research even more so. For their constant support and backing, I would like to thank our sponsors at the Toyota Collaborative Safety Research Center, the Amazon ML Solutions Lab, and Fujitsu North America. I would like to thank Mr. Pujitha Gunaratne in particular, for being an approachable and receptive collaborator from the very start.

The staff and employees of UC San Diego are some of the most friendly, patient, and helpful people I have had the pleasure of meeting. I thank the staff at the ECE department, Calit2, and the International Students & Programs Office for helping me navigate the complexities of graduate school, research, funding, taxes, and immigration. I cannot imagine how lost I would be without you guidance. I also thank John Minan and Crystal Liu for helping us with timely reimbursements and lab purchases. Finally, I would like to express my appreciation and gratitude to the custodians of the Facilities Management Services who ensured that we returned to a clean work space day after day.

One of the best parts about working at the Laboratory for Intelligent & Safe Automobiles is the people you work with; and I have had the pleasure to work with, and learn from everyone who has been in this lab since 2014. I would like to thank Sujitha, Eshed, Ravi, Kevan, Rakesh, Nachiket, Ross, Larry, Bowen, Ashish, Sean, Frankie, Sourabh, Borhan, Kirill, Grady, Nasha, Ishan, Siddharth, Aida and Hala for being all-around excellent colleagues to work with. Through the lab, I was also able to meet talented researchers like Eduardo, Daniela, Walter, Ole, Kaouther, Jacob, Miklas, Mark and Morten from different parts of the world. Although they were introduced to me through their research, getting to know them in person has only increased my respect and appreciation for them. I wish my colleagues the best in all their future endeavors, and cannot wait

to see the contributions they make to the field in the future.

Living in the US gave me the opportunity to reconnect with my relatives across the country. I thank them for making me feel at home, for giving useful advice, and for showing me new parks, towns and cities. It is always a source of comfort to know that family is nearby.

I thank my childhood friends Sahej, Rohan, Audi, and the fine people of OLPS and Chhadva. For better or for worse, you are partly responsible for the way I am today. I would also like to thank my friends from undergrad - Aman, Alok, Chetan, Greeshma, Julu, Khyati, Pawan, Prasen, Pratyush, Rashmi, Ritesh, Ritika, Sahil, Saurabh, Shruti and Vibu. You have given me a lifetime of memories to cherish, and moments to remember you by. I hope to see you all soon.

Moving to a new country is challenging, especially when you're alone. However, the friends I made at the University and elsewhere made me feel at home overnight. For this and all the amusing times we shared, I would like to thank my friends Ajay, Sanjay, Sid, Anupriya, Amanjit, Pandey ji, Shashank, Suneer, Manavi, Adi, Amitoj, Mrinmayee, Srishty, Kiran, Swati, Nirja, Prit and Raghav. Some old friendships from India were renewed again under fortunate circumstances in the US. It was fun reminiscing old times, and creating new memories with my longtime friends Abhisek, Ikbal, Saurabh, Dusi, Soumar, Sagar and Dhritiman.

I would also like to thank my roommates Pushp, Manish, Pragya, Katherine, Eduardo, Anchal and Serjic for making roommate horror stories seem like exaggerated works of fiction rather than a reality of life. You made living at home comfortable, chill, enjoyable, and full of interesting conversation. I would especially like to thank Dr. Pragya Sidhwani, for forcing me out of isolation, and for somehow managing to embody a Gossip Girl character and a champion Jeopardy contestant at the same time. For this, I wish you a lifetime of Harry Potter reruns.

Publication Acknowledgements:

CHAPTER 2, in full, is a reprint of the material as it appears in the IEEE Transactions on Intelligent Vehicles (2020), by Akshay Rangesh, and Mohan M. Trivedi. The dissertation author was the primary investigator and author of this paper.

CHAPTER 3, in full, is a reprint of the material as it appears in the IEEE Robotics and Automation Letters (2021), by Akshay Rangesh, Hala Abualsaud, Sean Liu, David Lu, Kenny Situ, and Mohan M. Trivedi. The dissertation author was one of the primary investigators and authors of this paper.

CHAPTER 4, in part, is a reprint of the material as it appears in arXiv:2101.04206 (2021), by Akshay Rangesh, Pranav Maheshwari, Mez Gebre, Siddhesh Mhatre, Vahid Ramezani, and Mohan M. Trivedi. The dissertation author was the primary investigator and author of this paper.

CHAPTER 5, in full, is a reprint of the material as it appears in the IEEE Transactions on Intelligent Vehicles (2018), by Akshay Rangesh, Nachiket Deo, and Mohan M. Trivedi. The dissertation author was one of the primary investigators and authors of this paper.

CHAPTER 6, in full, is a reprint of the material as it appears in the IEEE Transactions on Intelligent Vehicles (2021), by Akshay Rangesh, Bowen Zhang, and Mohan M. Trivedi. The dissertation author was one of the primary investigators and authors of this paper.

CHAPTER 7, in full, is a reprint of the material as it appears in the IEEE Conference on Computer Vision and Pattern Recognition - 3D Humans Workshop (2018), by Akshay Rangesh, and Mohan M. Trivedi. The dissertation author was the primary investigator and author of this paper.

CHAPTER 8, in full, is a reprint of the material as it appears in the IEEE Conference on Computer Vision - Workshop on Assistive Computer Vision and Robotics (2019), by Akshay Rangesh, and Mohan M. Trivedi. The dissertation author was the primary investigator and author of this paper.

CHAPTER 9, in part, is a reprint of the material as it appears in the IEEE Conference

on Intelligent Transportation Systems (2021), by Akshay Rangesh, Nachiket Deo, Ross Greer, Pujitha Gunaratne, and Mohan M. Trivedi. The dissertation author was one of the primary investigators and authors of this paper.

APPENDIX A, in full, is a reprint of the material as it appears in the IEEE Transactions on Intelligent Vehicles (2019), by Akshay Rangesh, and Mohan M. Trivedi. The dissertation author was the primary investigator and author of this paper.

VITA

2014	B. Tech. in Electronics and Telecommunication Engineering, National Institute of Technology, Silchar, India
2016	M.S. in Electrical Engineering (Intelligent Systems, Robotics & Control), University of California San Diego
2015-2021	Graduate Student Researcher, University of California San Diego
2021	Ph. D. in Electrical Engineering (Intelligent Systems, Robotics & Control), University of California San Diego

PUBLICATIONS

Akshay Rangesh, Nachiket Deo, Ross Greer, Pujitha Gunaratne and Mohan M. Trivedi, "Autonomous Vehicles that Alert Humans to Take-Over Controls: Modeling with Real-World Data", *IEEE Conference on Intelligent Transportation Systems*, 2021.

Akshay Rangesh, Hala Abualsaud, Sean Liu, David Lu, Kenny Situ and Mohan M. Trivedi, "LaneAF: Robust Multi-Lane Detection with Affinity Fields", *Robotics and Automation Letters*, 2021.

Akshay Rangesh, Pranav Maheshwari, Mez Gebre, Siddhesh Mhatre, Vahid Ramezani and Mohan M. Trivedi, "TrackMPNN: A Message Passing Graph Neural Architecture for Multi-Object Tracking", *arXiv:2101.04206*, 2021.

Akshay Rangesh, Bowen Zhang and Mohan M. Trivedi, "Gaze Preserving CycleGANs for Eyeglass Removal & Persistent Gaze Estimation", *IEEE Transactions on Intelligent Vehicles*, 2021.

Akshay Rangesh, Bowen Zhang and Mohan M. Trivedi, "Driver Gaze Estimation in the Real World: Overcoming the Eyeglass Challenge", *IEEE Intelligent Vehicles Symposium*, 2020.

Akshay Rangesh and Mohan M. Trivedi, "Ground Plane Polling for 6DoF Pose Estimation of Objects on the Road", *IEEE Transactions on Intelligent Vehicles*, 2020.

Akshay Rangesh and Mohan Trivedi, "Forced Spatial Attention for Driver Foot Activity Classification", *IEEE International Conference on Computer Vision - Workshop on Assistive Computer Vision and Robotics*, 2019.

Walter Zimmer, Akshay Rangesh and Mohan Trivedi, "3D BAT: A Semi-Automatic, Web-based 3D Annotation Toolbox for Full-Surround, Multi-Modal Data Streams", *IEEE Intelligent Vehicles Symposium*, 2019.

Akshay Rangesh and Mohan M. Trivedi, "No Blind Spots: Full-Surround Multi-Object Tracking for Autonomous Vehicles using Cameras & LiDARs", *IEEE Transactions on Intelligent Vehicles*, 2019.

Akshay Rangesh, Nachiket Deo, Kevan Yuen, Kirill Pirozhenko, Mohan M. Trivedi, Heishiro Toyoda and Pujitha Gunaratne, "Exploring the Situational Awareness of Humans inside Autonomous Vehicles", *IEEE Conference on Intelligent Transportation Systems*, 2018.

Akshay Rangesh and Mohan M. Trivedi, "HandyNet: A One-stop Solution to Detect, Segment, Localize & Analyze Driver Hands", *IEEE Conference on Computer Vision and Pattern Recognition - 3D HUMANS Workshop*, 2018.

Nachiket Deo, Akshay Rangesh and Mohan M. Trivedi, "How would surround vehicles move? A Unified Framework for Maneuver Classification and Motion Prediction", *IEEE Transactions on Intelligent Vehicles*, 2018.

Sourabh Vora, Akshay Rangesh and Mohan M. Trivedi, "Driver Gaze Zone Estimation using Convolutional Neural Networks: A General Framework and Ablative Analysis", *IEEE Transactions on Intelligent Vehicles*, 2018.

Akshay Rangesh and Mohan M. Trivedi, "When Vehicles See Pedestrians with Phones: A Multi-Cue Framework for Recognizing Phone-based Activities of Pedestrians", *IEEE Transactions on Intelligent Vehicles*, 2018.

Akshay Rangesh, Kevan Yuen, Ravi Satzoda, Rakesh Rajaram, Pujitha Gunaratne, Mohan M. Trivedi, "A Multimodal, Full-Surround Vehicular Testbed for Naturalistic Studies and Benchmarking: Design, Calibration and Deployment", *arXiv:1709.07502*, 2017.

Sourabh Vora, Akshay Rangesh, and Mohan M. Trivedi, "On Generalizing Driver Gaze Zone Estimation using Convolutional Neural Networks", *IEEE Intelligent Vehicles Symposium*, 2017.

Akshay Rangesh, Eshed Ohn-Bar, Kevan Yuen and Mohan M. Trivedi, "Pedestrians and their Phones - Detecting Phone-based Activities of Pedestrians for Autonomous Vehicles", *IEEE International Conference on Intelligent Transportation Systems*, 2016.

Sujitha Martin, Akshay Rangesh, Eshed Ohn-Bar, and Mohan M. Trivedi, "The Rhythms of Head, Eyes, and Hands at Intersections", *IEEE Intelligent Vehicles Symposium*, 2016.

ABSTRACT OF THE DISSERTATION

Intelligent Human-in-the-Loop Vehicular Automation with Real-Time Vision Models

by

Akshay Rangesh

Doctor of Philosophy in Electrical Engineering
(Intelligent Systems, Robotics & Control)

University of California San Diego, 2021

Professor Mohan M. Trivedi, Chair

Modern vehicular automation technology is enabled by complex interactive systems composed of discrete blocks that accomplish specific tasks. These blocks are responsible for perceiving the environment surrounding the vehicle, predicting the future states and intents of other agents in its vicinity, planning the vehicle's path with an eventual goal in mind, and finally, providing control signals and actuations that achieve this goal. While such systems are widely adopted, they fail to consider a key component of the driving process - *the human occupants inside the vehicle*. This dissertation proposes that to ensure a safe, smooth, and enjoyable travel experience, it is essential for the human driver and vehicular automation to be aware of each

other's states and impending failures, i.e., a relationship built on earned trust rather than blind faith.

To codify this requirement, we employ the Looking-In & Looking-Out (LILO) approach, where systems simultaneously model the inside and outside of vehicles. In addition to the set of "looking-out" tasks that are used in conventional automation, we propose a parallel set of "looking-in" tasks that model and analyze the interior of the vehicle. The outputs generated by both these parallel systems can then be integrated to provide useful controls based on a more complete understanding of the circumstances and the environment.

This dissertation presents the models and algorithms we have developed to accomplish the individual tasks that make up the system described above. This includes our research on real-time models for tasks such as object detection, multi-object tracking, and maneuver/trajectory prediction. We also introduce new approaches for the relatively under-explored "looking-in" tasks - which we hope will inspire many such studies in the future. Some of our contributions in this space include real-time models to analyze different aspects of the driver's state, data augmentation and automatic labelling schemes for tasks with limited data, and solutions to other common hindrances. Finally, we provide a concrete example of how the LILO framework can be used in practice - where the driver's state estimated by "looking-in" is used in conjunction with metrics computed by "looking-out" to initiate safer and smoother control transitions.

Part I

Introduction

Chapter 1

Introduction & Preliminaries

1.1 The Road to Complete Automation

With recent developments in machine learning, sensor technology, hardware and software engineering, very few domains have witnessed the extent of growth in innovation, investment, and automation as the field of vehicular transportation. While the automobile landscape has already undergone drastic alterations in recent years, more innovations and fundamental changes to how humans travel are anticipated in the near future. To keep track of this rapidly evolving field, the Society of Automotive Engineers (SAE) defines five levels of driving automation (plus a "no-automation" level 0). These levels of automation provide a broad taxonomy to categorize different automation features and provide a yardstick to measure general progress in the field - from completely manual (level 0) to full driving automation (level 5).

Based on the SAE levels of automation, level 0 systems are limited to providing warnings and momentary assistance, level 1 systems provide steering or brake/acceleration support while level 2 systems provide concurrent support for both, level 3 and level 4 systems can drive the vehicle under limited conditions, and finally, level 5 systems can drive the vehicle under all

conditions. Another key factor that determines the level of autonomy is *what is required of the human driver/occupant*. This is especially evident in the distinction between levels 3 and 4, where humans in level 3 systems are required to be in the loop and take control of the vehicle if the system requests so. Thus, humans are a critical part of the driving process until complete automation is attained.

Although the field of intelligent and autonomous vehicles has made remarkable progress in recent years, it is widely regarded that full driving automation is still years if not decades away. Even if level 4 and level 5 systems are made operable in certain environments and geographical locations, these systems would still be inaccessible to a vast majority of the population. This could be due to a variety of reasons - the lack of necessary road infrastructure, models/algorithms failing to operate reliably under different/harsher conditions, monetary and/or manufacturing constraints, country-specific formal and informal traffic rules, etc. This highlights the importance of intermediate levels of driving automation, as these systems can be deployed with the technology available today and still provide tangible benefits to vehicle occupants. Such a system, if deployed correctly, would not only improve the safety and reliability of the vehicle itself, but also provide an enjoyable and fatigue-free travel experience for the humans inside it.

1.2 Why Monitor Driver/Occupant Behavior?

The intermediate levels of automation imply shared responsibility between the human driver and the autonomous system. For such systems to operate reliably for prolonged periods of time, it is imperative to ensure that both the driver and the system are aware of each other's states and impending failures. This would allow the driver to compensate for the shortcomings of the system by taking control of the vehicle and vice versa. Such events are termed *control transitions*, and are an unavoidable facet of partially automated systems. Thus, for safe and smooth transfer of control, a partially automated vehicle must not only model the environment *outside* the vehicle,

but also model the behavior of the driver/occupants *inside* it too.

Motivations for studying driver/occupant behavior in the context of highly automated vehicles can also be found aplenty in human-factors research. It is widely regarded that as soon as the level of cognitive stimulation falls below a person's own comfortable "set point", the person will seek out alternate/additional sources of information, leading to distraction. This makes the intermediate levels of automation potentially dangerous, causing problems such as inattention, over-trust, skill atrophy, complacency etc. [1]. Elsewhere, the authors in [2] emphasize the *irony of automation*, whereby "the more advanced a control system is, the more crucial may be the contribution of the human operator". They also acknowledge that decades of research has shown that humans are not particularly good at tasks that require vigilance and sustained attention over long periods of time. All above points suggest that a drop in attention is inherent in human behavior. Thus, it is not a matter of *if*, but *when* the driver will resort to non-ideal behavior.

Another noteworthy trend is the misuse and disregard of existing safety features that fail to keep the driver alert and attentive despite the system's best efforts. Recent accidents, near-crashes, and several YouTube videos have provided many examples of driver's fooling the system by attaching simple weights to the steering wheel in order to simulate torque from a human's hands. This turns off the system's audiovisual alert mechanism indefinitely or until it is too late to take control. This has resulted in situations where the driver is asleep/impotent behind the wheel, has moved to the backseat to leave the driver's seat empty, or even worse, has evacuated the car completely. These examples highlight the shortcomings of existing systems that monitor driver behavior, which is further compounded by the inaccurate marketing of such products and the excessive trust of consumers in such vehicles. This also demonstrates the need for a more sophisticated driver monitoring system - one incorporating visual sensors and a holistic driver behavior model.

1.3 Looking-In & Looking-Out Systems

To address the issues presented above, this dissertation adopts the *Looking-In & Looking-Out (LILO)* framework originally described in [3, 4]. In this initial study, the authors intended to correlate the visual contextual information of the vehicle interior with the vehicle exterior, and went on to describe novel sensory systems and algorithms for capturing not only the dynamic surround information of the vehicle, but also the state, intent, and activity patterns of drivers. This framework additionally allows us to simultaneously model the driver, the environment and the vehicle to create systems and features that can reason across all active agents, and act in a manner that results in maximum safety while minimizing annoyances.

The core competencies of most modern autonomous vehicle systems can be broadly categorized into three categories [5], namely - perception, prediction & planning, and control & actuation. In this categorization, perception refers to the ability of an autonomous system to perceive its surroundings and extract relevant knowledge from the environment - including the ego-vehicle's own state in the environment. Prediction involves forecasting (with uncertainty) the future states of all agents in the scene. Planning refers to the process of making purposeful decisions in order to achieve the robot's higher order goals. This typically involves the final destination of the vehicle while optimizing for secondary objectives like travel time, comfort, fuel efficiency, etc. Finally, the control competency refers to the robot's ability to execute the planned actions that have been generated by the higher level processes that precede it.

This categorization, however, only considers the *looking-out* aspects of driving, i.e., the modelling of the environment. We believe that a similar categorization can be made from a *looking-in* perspective as well, where the individual blocks concern themselves with perception and prediction of, and planning for the various states/behaviors of the human driver. This concept is illustrated in Figure 1.1, where we depict the different stages involved in a modern LILO framework, under the traditional categorization of autonomous vehicle systems. While the

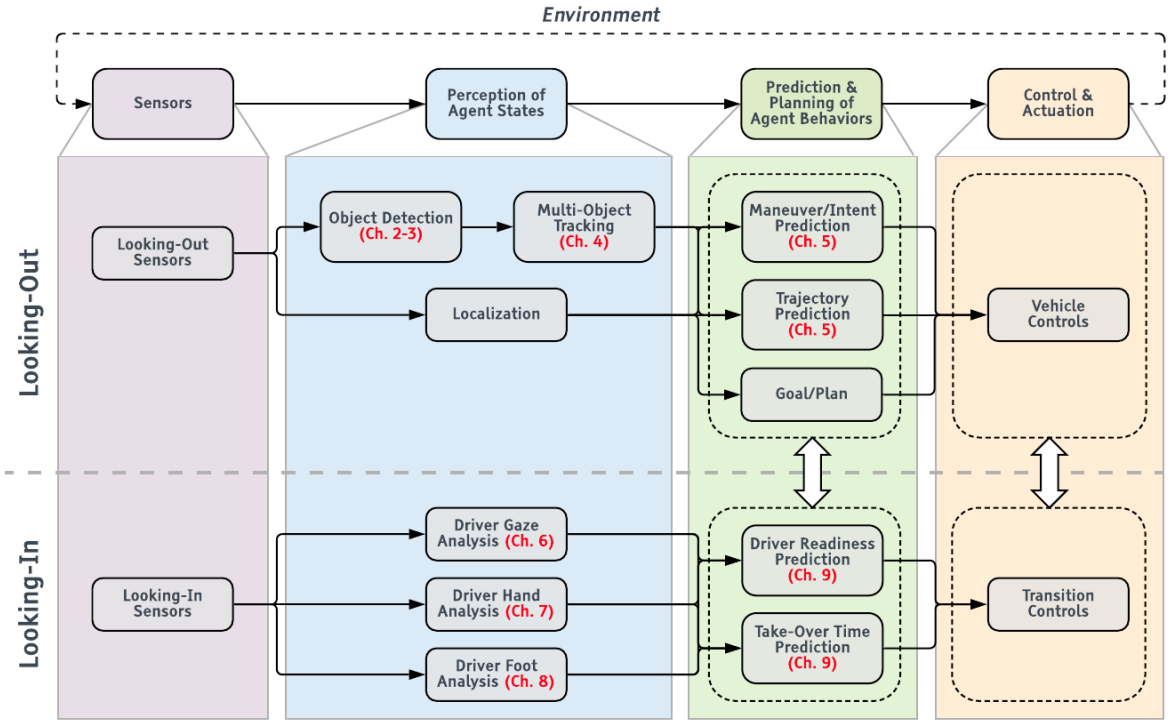


Figure 1.1: **System overview of a modern (semi-)autonomous vehicle:** this dissertation proposes that akin to the conventional *perception-planning-control* categorization of autonomous vehicle systems for "looking-out", a similar categorization for monitoring the driver by "looking-in" is essential for prolonged safety and reliability. The chapters covering each individual block above are indicated in red.

looking-in and looking-out aspects of the system in large parts operate independently, information between the two workflows can still be exchanged - especially at the prediction, planning and control stages. Examples of this include conditioning the future trajectories of surround vehicles on the driver's intent/state, changing the ego-vehicle plan based on driver state/health, or initiating control transitions based on joint reasoning over the state of the human driver and the environment.

1.4 Contributions & Outline

This dissertation operates within the LILO framework presented above, and details our contributions and innovations in the individual blocks that make up the system. The dissertation is

divided into multiple parts based on the nature of the environment being modelled and/or operated in. For example, Chapters 2-5 deal with the modelling of the environment outside the ego-vehicle, and hence belong to the **Looking-Out** part of the dissertation. Similarly, Chapters 6-8 focus on modelling of the environment inside the ego-vehicle, and hence belong to the **Looking-In** part of the dissertation. Finally, Chapter 9 concerns itself with information streams obtained from both within and outside the ego-vehicle, and thus is placed separately in the **Looking-In & Looking-Out** part of the dissertation. For better navigability, we refer the reader to Figure 1.1, where the chapters corresponding to each block are indicated in red. We also provide a brief overview of each chapter below for the reader's benefit.

CHAPTER 2 presents our approach for monocular 3D object detection using ground plane polling. In this approach, we find the best-fit plane for each object in the scene, and using estimated 2D keypoints, object dimensions, are able to completely recover the 6 Degree-of-Freedom (DoF) pose of the object under consideration.

CHAPTER 3 presents our approach to lane detection based on affinity fields. In particular, we first perform a binary segmentation to identify lane pixels in an image, and further estimate two affinity vector fields named the Horizontal Affinity Field (HAF) and Vertical Affinity Field (VAF), which together allow us to cluster the foreground pixels into an unknown number of lanes using a simple row-by-row decoding procedure.

CHAPTER 4 introduces a tracking framework based on undirected graphs that represent the data association problem over multiple timesteps in the tracking-by-detection paradigm. In this framework, both individual detections and associations between pairs of them are represented as nodes in the graph. Furthermore, the graph is dynamic, where only detections (and their associations) within a certain temporal range are processed in a rolling window basis. We also introduce a lightweight message-passing neural network that operates on this dynamic graph to produce multiple object tracks in real-time.

This dissertation also presents a second approach to multi-object tracking for applications

in which multiple sensing modalities are available (see Appendix A). This approach is designed to be modular, and can work with any number of sensors and different sensing modalities. This approach also leverages benefits from different sensors via early fusion, and extends the popular Markov Decision Process framework to accommodate information from 2D and 3D sensors. These changes allow us to track vehicles for long durations, and in some cases for more than a few minutes.

CHAPTER 5 presents a joint approach to identify the maneuver and predict the future trajectory of all agents in the driving scene. This includes an Hidden Markov Model (HMM) based maneuver classifier, a Variational Gaussian Mixture Model (VGMM) based trajectory generator, and a final Vehicle Interaction Module (VIM) that reasons over all future trajectories and outputs high probability trajectories that are non-colliding.

CHAPTER 6 details our efforts to overcome the challenges associated with estimating gaze in the real-world. Specifically, we address two problems encountered by gaze classifiers, namely - harsh/no illumination and eyeglasses. We show that the use of infrared (IR) cameras and input normalization suffice for most illumination conditions, while a suitably trained eyeglass removal network can help alleviate issues arising from eyeglasses. To this end, we propose a Gaze Preserving CycleGAN (GPCycleGAN) that is capable of removing eyeglasses and related artifacts while preserving the gaze direction of the human driver for downstream gaze estimation.

CHAPTER 7 presents our semi-automatic labelling strategy based on chroma-keying that enables us to obtain large amounts of labelled data to analyze the hands of a human driver. Using this labelling scheme, we are able to obtain accurate pixelwise segmentation masks for each hand of the driver, which we then use to train a model than can segment each hand and localize it in 3D space. We additionally identify and classify objects held in each detected hand instance.

CHAPTER 8 presents a simple activity classification model for the human driver's feet. For additional robustness and to prevent overfitting, we introduce a novel forced spatial attention loss during training. This loss encourages the model to focus its attention to relevant regions in

image-space that pertain to the individual activity classes when making a prediction.

CHAPTER 9 follows from the chapters preceding it, and also offers one potential culmination of the LILO system described before. In particular, this chapter introduces one of the largest studies on take-over times and control transitions involving real-world drives with 89 subjects operating a Tesla Model S experimental platform. Using these real-world take-over events and associated labels, we show that a real-time, sequential model for predicting take-over times is achievable. We also offer modelling tweaks and data augmentation schemes to further improve the predictions, and provide qualitative examples to study the proposed TOT model's behavior.

CHAPTER 10 offers some final thoughts and concluding remarks. We also provide brief comments on the state of each field, point out the limitations of current approaches, and highlight some new and exciting advances that attempt to overcome them.

Part II

Looking-Out

Chapter 2

Ground Plane Polling for 6DoF Pose Estimation of Objects on the Road

This chapter introduces an approach to produce accurate 3D detection boxes for objects on the ground using single monocular images. We do so by merging 2D visual cues, 3D object dimensions, and ground plane constraints to produce boxes that are robust against small errors and incorrect predictions. First, we train a single-shot convolutional neural network (CNN) that produces multiple visual and geometric cues of interest: 2D bounding boxes, 2D keypoints of interest, coarse object orientations and object dimensions. Subsets of these cues are then used to *poll* probable ground planes from a pre-computed database of ground planes, to identify the "best fit" plane with highest consensus. Once identified, the "best fit" plane provides enough constraints to successfully construct the desired 3D detection box, without directly predicting the 6DoF pose of the object. The entire ground plane polling (GPP) procedure is constructed as a non-parametrized layer of the CNN that outputs the desired "best fit" plane and the corresponding 3D keypoints, which together define the final 3D bounding box. Doing so allows us to poll thousands of different ground plane configurations without adding considerable overhead, while also creating a single CNN that directly produces the desired output without the need for post processing. We evaluate our method on the 2D detection and orientation estimation benchmark from the challenging KITTI dataset, and provide additional

Code: https://github.com/arangesh/Ground-Plane-Polling

comparisons for 3D metrics of importance. This single-stage, single-pass CNN results in superior localization and orientation estimation compared to more complex and computationally expensive monocular approaches.

2.1 Introduction

Localizing objects in 3D is of extreme importance in autonomous driving and driver safety applications. Traditional and contemporary approaches have mostly relied on range sensors like LiDARs and Radars, or stereo camera pairs to predict the desired 6DoF pose and dimensions of objects of interest. Some of these approaches are demonstrably robust under a variety of conditions, and produce high quality 3D detection boxes despite large occlusions, truncation etc. These approaches benefit from the use of 3D data, either as LiDAR point clouds, range measurements from Radars, depth maps obtained from stereo cameras, or a combination thereof. However, the benefits of 3D sensors are almost always accompanied by certain downsides. These sensors are typically orders of magnitudes more expensive compared to cheap cameras, and are also bulkier and power-hungry. It is therefore desirable to carry out 3D object detection with monocular cameras, if suitable robustness can be achieved. A robust 3D detector could also in turn improve the performance of purely camera based tracking [6], prediction [7–9], and other driver safety systems [10] and tools [11] in general. This however introduces many challenges, most of which stem from the fact that predicting 3D attributes from 2D measurements is an ill-posed problem.

In this study, we overcome this challenge by only considering those cues (visual or otherwise), that can be reliably predicted using monocular camera images alone, while also being generalizable to data captured by cameras with different settings and parameters. With this in mind, we only predict attributes like 2D detection boxes, 2D keypoint locations, coarse local orientations, and dimensions of the object in 3D. State-of-the-art approaches for predicting 2D attributes like detection boxes and keypoints are known to generalize quite well to new datasets

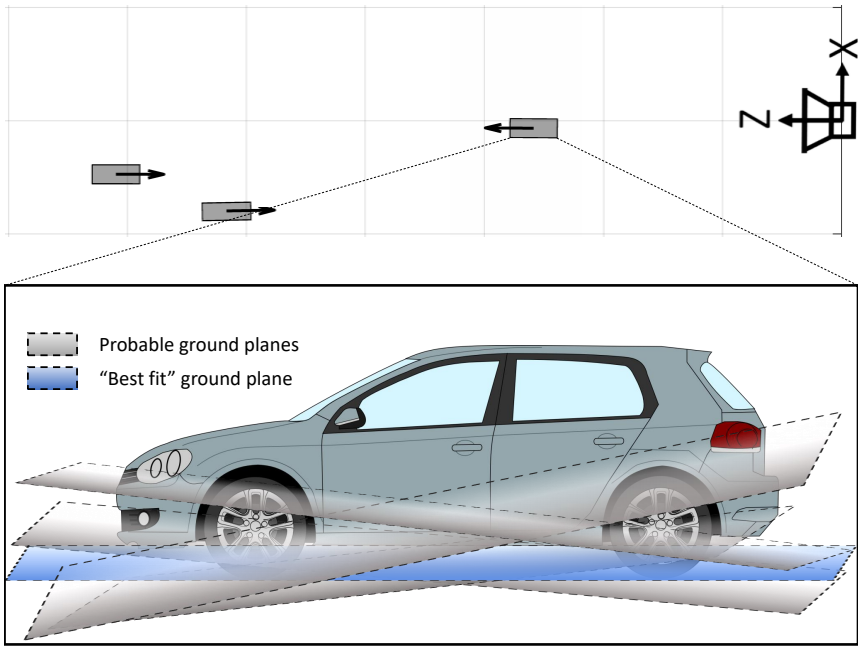


Figure 2.1: Illustration of the ground plane constraint enforced in this study. Each object is assumed to lie on one of many probable ground plane configurations - termed to be the "best fit" plane associated with the object.

and scenarios. On the other hand, coarse local orientations are closely tied to the appearance of an object, and can be reliably predicted as we will show later. Finally, the dimensions of an object in 3D are somewhat tied to the appearance of an object (e.g. cars, vans, trucks etc. look different), but are less prone to large errors because of the low variance in dimensions of an object within a particular object class.

On the other hand, we would like the constraints and assumptions we enforce to result in reasonable estimates, while not being too restrictive. Ground plane constraints are a common choice, where objects of interest are forced to lay on a common plane. This however, might be too restrictive of an assumption, resulting in large errors for objects farther away lying on irregular terrains. In this study, we instead create a database of probable ground planes, and choose the "best fit" plane for each object locally (see Figure 2.1). This is similar to modeling the road terrain with a piecewise planar approximation, thereby loosening the single ground plane

constraint. We also purposefully predict more attributes than needed to estimate a 3D detection box, and use these predictions to form maximal *consensus set* of attributes, in a manner similar to traditional RANSAC approaches. This makes our approach robust to individual errors and outliers in predictions.

Our main contributions in this work can be summarized as follows - 1) We propose a single-shot approach to predict the 6DoF pose and dimensions of objects on the road by predicting 2D attributes of interest in single monocular images. 2)We then combine subsets of these attributes to robustly identify the "best fit" ground plane for each object locally using a novel polling approach, thereby determining the 3D detection box corresponding to each object. 3) Finally, we carry out extensive comparisons and experiments with previous state-of-the-art techniques to illustrate the advantages and limitations of our approach.

2.2 Related Research

Since our approach is based on extending 2D object detectors to infer 6DoF pose, we first give a brief overview of recent 2D detection approaches. 2D detection of objects in single images has long been of interest to the computer vision community. Most recent works rely on convolutional neural networks (CNNs) to produce high quality 2D boxes for a variety of objects under challenging scenarios. 2D object detection is generally approached in two ways - using single shot networks, or by using multi-stage architectures. Single shot (or single stage) detectors like [12–14] directly regress to the offset between predefined anchors in a grid, in a dense manner. Additionally, anchors are scored to determine which boxes to retain and the class of the corresponding object. Multi-stage detectors split these operations into two stages, namely the proposal generation stage, followed by the proposal processing stage [15–17]. During inference, these detectors first propose candidate detection boxes, each of which is then individually processed and refined by a second network that outputs the desired 2D

detection box and object class. Even though the object proposal stage reduces the search space significantly, having to process each proposal individually imposes a significant toll on the runtime. Consequently, two stage methods are generally slower than single shot methods, albeit with better performance.

Classical methods for estimating 3D pose primarily involve identifying distinct keypoints and finding correspondences between different views by matching features. These feature descriptors were designed to be invariant to changes in scale, rotation, illumination and keypoints [18–21]. Other related approaches involve 3D model-based registration [22, 23], and Hausdorff and Chamfer matching for edges and curves [24–26]. Such methods are often fast and work reasonably well in cluttered scenes. However, they are too reliant on texture, high resolution images, and do not take into account high level information about the scene.

With the introduction of new large-scale datasets like [27–29] and overall success of CNNs, many approaches for 3D object detection and 6DoF pose estimation using single monocular images have recently emerged. Although these methods are mostly based on popular 2D detection architectures [13–15, 17], they differ in the ways they incorporate 3D information, the attributes they predict, and the constraints they enforce. In particular, we find that most monocular 3D detection methods can be categorized into the following three groups or a combination thereof - approaches that make use of 3D models, templates or exemplars, approaches that use 3D bounding box proposals, and approaches that enforce geometric constraints. We describe representative works from each category below.

Xiang et al. [30] cluster the set of possible object poses into viewpoint-dependent subcategories using 3D voxel patterns. In their following work [31], these subcategories were used as supervision to train a network to detect and classify the subcategory of each object. Subcategory information is then transferred to obtain the pose of each object. More recently, Chabot et al. [32] create a dataset of 3D shapes and templates, and identify the most similar template for each object that is detected. To do so, they manually annotate vehicle part coordinates, part visibility

and the 3D template corresponding to each object in the training set. At test time, their network predicts all part coordinates, their visibility and the most similar template. A 2D/3D matching algorithm (Perspective-n-Point) then produces the desired object pose. These methods generally provide more information about each object, at the expense of having a more convoluted approach involving a database of shapes, templates, voxel patterns etc., and sometimes requiring more manual annotations.

Archetypal studies that make use of 3D object proposals for monocular detection are presented by Chen et al. in [33, 34]. In these studies, the authors first sample candidate 3D boxes using a ground plane assumption and object size priors. These boxes are then scored by exploiting different cues like semantic and instance classes, context, shape, location etc. Although these methods work well on 2D detection tasks, they fail to localize objects accurately in 3D.

Unlike previous methods, some recent studies have chosen to enforce geometric constraints to obtain the 6DoF pose of objects. In [35], Mousavian et al. only predict the orientation of objects, and use the fact that the perspective projection of a 3D bounding box should fit tightly within its 2D detection window. This constraint, expressed as a linear system of equations, when solved, results in the desired 6DoF pose of the object. In a more straightforward approach, the authors in [36] use a single shot network to predict 2D projections of all 8 corners and the centroid of the 3D bounding box, and use these 2D-3D correspondences to obtain the 6DoF pose of the object by solving the Perspective-n-Point (PnP) problem. These methods, although conceptually simple and straightforward to implement, rely on the accuracy of all predicted entities.

2.3 Network Architecture

2.3.1 Overview

Our primary focus in this study is to propose an approach to real time 3D object detection for the purpose of autonomous driving. With this in mind, our network design is based on

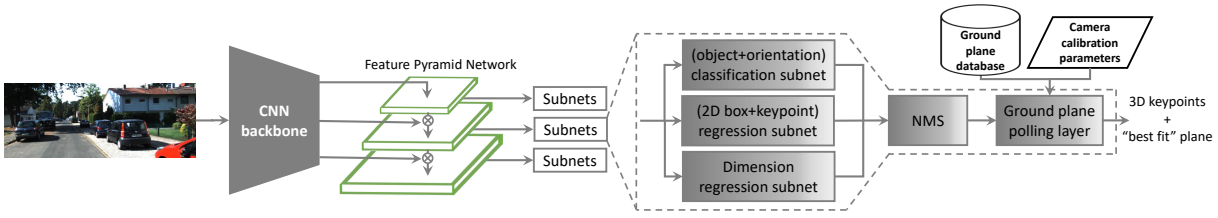


Figure 2.2: Network architecture and our overall approach for predicting 3D bounding boxes from monocular images.

the RetinaNet detector presented in [14]. Although our approach can be adapted to work with any generic object detector based on anchor boxes, we decided to work with the RetinaNet architecture because it matches the speed of other one-stage detectors, while having comparable performance to state-of-the-art two stage detectors. As illustrated in Figure 2.2, we retain the backbone structure based on Feature Pyramid Networks (FPN) [37], and modify and add to the subnetworks that follow. For most of our experiments, we use a ResNet50 backbone [38] with five pyramid levels (P_3 to P_7) computed using convolutional features C_3, C_4 and C_5 from the ResNet architecture as before. However, unlike [14], all pyramid levels have $C = 512$ channels instead of 256 to account for the increase in the number of subnetwork outputs, and the number of channels per output. We describe the purpose and structure of each subnetwork in the subsections that follow. Details of the ground plane polling layer are presented in the following section.

Similar to previous works, we use translation-invariant anchor boxes with areas of 32^2 to 512^2 on pyramid levels P_3 to P_7 respectively. As is common practice, we use anchors at three aspect ratios (1:2, 1:1, 2:1). However, for denser scale coverage and to account for objects that are farther away, we consider 4 different relative scales ($2^{-1/3}, 2^0, 2^{1/3}, 2^{2/3}$) of each anchor aspect ratio, resulting in a total of $A = 12$ anchors per location, per level. These anchors together cover a scale range of $25 - 813$ pixels with respect to the network's input image.

The multi-scale features obtained from each level of the FPN are then processed by three different subnetworks as shown in Figure 2.2. First, the (object + orientation) classification subnetwork outputs the object class and coarse 3D orientation range of each object in the

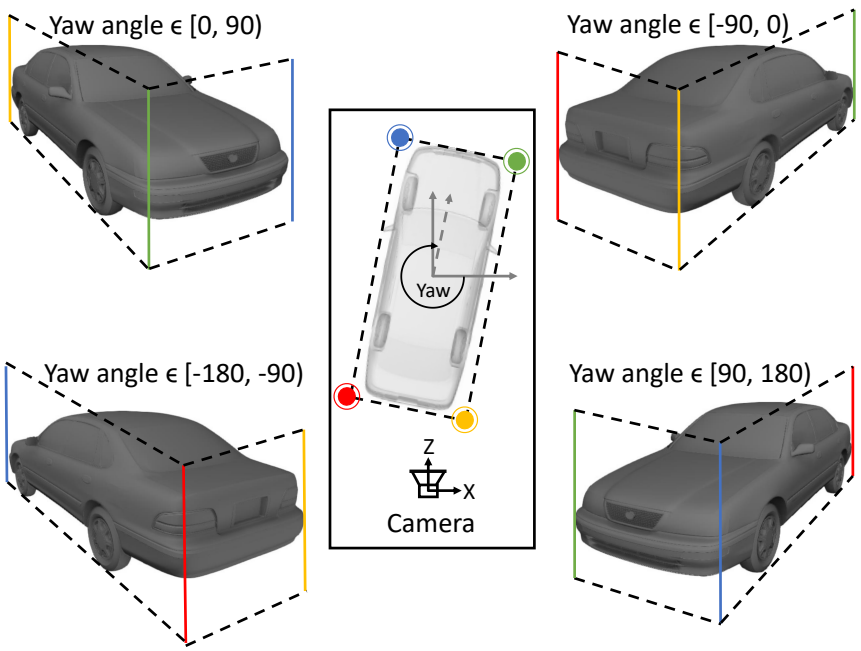


Figure 2.3: Illustration of the 4 different orientation classes based on yaw angles (corners of the figure), and a bird's-eye view of how the yaw angle is calculated (center of the figure). Best viewed in color.

scene. Next, the (2D box + keypoint) regression subnetwork regresses to the desired 2D box and keypoints of interest from designated anchor boxes with sufficient overlap. Finally, the dimension regression subnetwork outputs the dimensions (height, width, length) of the 3D box corresponding to objects at each anchor location. These subnetwork outputs are then passed on to a non-parametrized *ground plane polling* layer. The ground plane polling layer takes in the outputs of each subnetwork, the camera calibration parameters, and a database of probable ground planes, and outputs the desired "best fit" plane, and the corresponding 3D location of each keypoint. We describe the purpose and structure of each subnetwork in the subsections that follow. Details of the ground plane polling layer are presented in the following section.

2.3.2 Subnetwork Architectures

(Object + Orientation) Classification Subnetwork

This subnetwork is similar to the classification head found in 2D detectors, with the notable addition of orientation classes. In particular, we define 4 coarse orientation classes tied to 4 different ranges into which an object's yaw angle in the camera coordinate frame may fall. These orientation classes and the corresponding change in object appearance is depicted in Figure 2.3. In this Figure, each vertical edge of the 3D bounding box is color coded to indicate its relationship to the orientation classes. Each orientation class is further split into two classes depending on the relative locations of certain keypoints of interest with respect to the center of each anchor. This yields a total of 8 orientation classes. We explain the nature of these split classes in our description of the regression subnetwork.

The classification subnetwork architecture is then modified to predict not just the object class, but also the desired orientation class. As can be seen in Figure 2.4, this subnetwork takes in the output from each level of the FPN, applies a series of convolutional operations resulting in an output with $8KA$ channels to account for each of 8 orientation classes, K object classes, and A anchors per location. We use a focal loss [14] to train this subnetwork:

$$L_{class}(\mathbf{p},\mathbf{y}) = -\sum_{k=1}^{K}\sum_{o=1}^{8} \boldsymbol{\alpha}_{k,o}(1-\mathbf{p}_{k,o})^{\gamma}\mathbf{y}_{k,o}\log(\mathbf{p}_{k,o}), \tag{2.1}$$

where $\mathbf{p}$ and $\mathbf{y}$ are the estimated probabilities and one-hot ground truth vector corresponding to positive and negative anchors. Specifically, anchors are positive if they have an intersection-over-union (IoU) greater than 0.5 with the ground truth and are considered negative if their IoU is in $[0, 0.4)$. Other anchors are ignored. γ and $\boldsymbol{\alpha}$ are hyperparameters of the focal loss. While γ is a simple scalar, $\boldsymbol{\alpha}$ represents the weights that account for foreground-background class imbalance and are defined as follows:

$$\boldsymbol{\alpha}_{k,o} = \begin{cases} \alpha, & \text{if } \mathbf{y}_{k,o} = 1 \\ 1-\alpha, & \text{otherwise.} \end{cases} \tag{2.2}$$

The total classification loss is summed across all positive and negative anchors, and is normalized

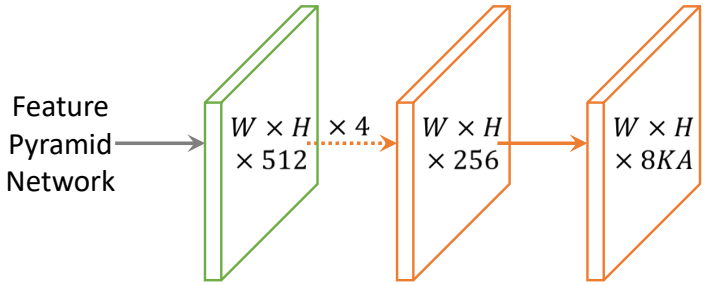


Figure 2.4: **(object + orientation) classification subnetwork:** output channels accounts for 8 orientation classes, per anchor, per object class.

by the total number of positive anchors.

(2D Box + Keypoint) Regression Subnetwork

This subnetwork is used to regress to not only the desired 2D detection box, but also four additional keypoints of interest: $\mathbf{x}_l$, $\mathbf{x}_m$, $\mathbf{x}_r$, and $\mathbf{x}_t$ that denote the left, middle, right and top *visible* corners of the desired 3D bounding box when projected into the image plane. We perform class and orientation agnostic regression for both the 2D boxes and the keypoints. This implies that a given keypoint may represent different corners of the 3D bounding box depending on the orientation class of the object in question.

Figure 2.5 outlines the different regression targets for each positive anchor, and also the edge of the anchor from which each target is regressed. The subnetwork architecture and its outputs are depicted in Figure 2.6.

For the X coordinates x_m and x_t of keypoints $\mathbf{x}_m$ and $\mathbf{x}_t$, we only regress to the absolute values of the target (see Figure 2.6). The corresponding signs of these regression targets for each anchor are accounted for by the split in orientation classes described in the classification subnetwork i.e. for each anchor, the classification network picks one of 8 orientation classes, which both defines the coarse orientation of the object, and the sign of the regression targets for x_t and x_m. This is especially important for degenerate vehicle orientations close to the boundaries of different orientation classes. For example, an aligned vehicle directly in front of the camera could have keypoints $\mathbf{x}_m$ and $\mathbf{x}_t$ close to either the left or right edges of a positive anchor box depending

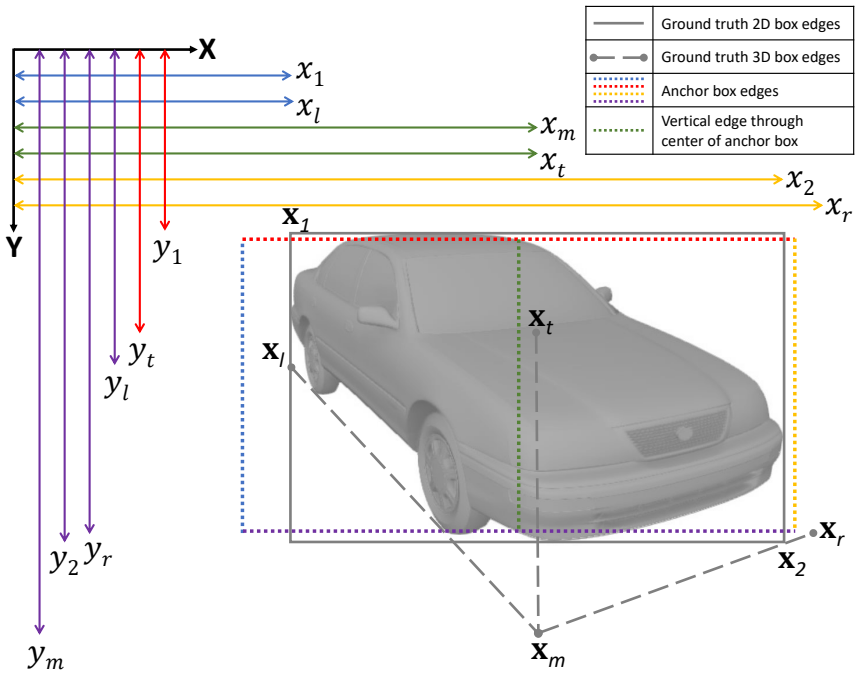


Figure 2.5: Illustration of proposed regression targets for each anchor. Each target is colored based on the anchor edge from which it is regressed. Best viewed in color.

on the orientation class that is chosen.

We use a smooth $L1$ loss for all regression heads. The total loss for each positive anchor in this subnetwork is the sum of all individual losses:

$$\begin{aligned} L_{reg}(\tilde{\mathbf{t}}, \mathbf{t}) = L_{smooth-L1}(\tilde{\mathbf{t}}_{2D}, \mathbf{t}_{2D}) + L_{smooth-L1}(\tilde{\mathbf{t}}_{\mathbf{x}_l}, \mathbf{t}_{\mathbf{x}_l}) \\ + L_{smooth-L1}(\tilde{\mathbf{t}}_{\mathbf{x}_m}, \mathbf{t}_{\mathbf{x}_m}) + L_{smooth-L1}(\tilde{\mathbf{t}}_{\mathbf{x}_r}, \mathbf{t}_{\mathbf{x}_r}) \\ + L_{smooth-L1}(\tilde{\mathbf{t}}_{\mathbf{x}_t}, \mathbf{t}_{\mathbf{x}_t}), \end{aligned} \tag{2.3}$$

where $\tilde{\mathbf{t}}$ and $\mathbf{t}$ are the regression outputs and the corresponding targets respectively. The total loss for this subnetwork is the average loss over all positive anchors.

Dimension Regression Subnetwork

This subnetwork has a similar architecture to the (object + orientation) classification subnetwork, but with fewer channels (128 instead of 256) per convolution. The network directly

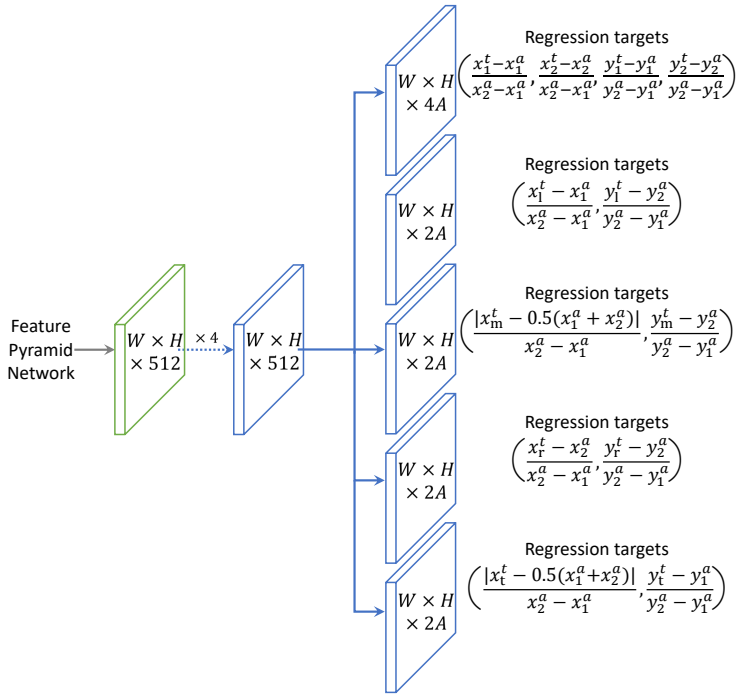


Figure 2.6: **(2D box+keypoint) regression subnetwork:** we use separate heads for the 2D box and each keypoint. All regression targets associated with a target bounding box $(x_1^t, x_2^t, y_1^t, y_2^t)$, its keypoints $\{(x_l^t, y_l^t), (x_m^t, y_m^t), (x_r^t, y_r^t), (x_t^t, y_t^t)\}$, and a positive anchor $(x_1^a, x_2^a, y_1^a, y_2^a)$ are shown for reference.

outputs the three dimensions (height, width, length) of the desired 3D bounding box, but in a class specific manner. By making the predictions class specific, we improve the prediction accuracies just by reducing the variance within each class. The subnetwork architecture is shown in Figure 2.7, resulting in an output with $3KA$ channels, representing the 3 dimensions of the box, for each of K object classes, and each of A anchors per position.

The dimension regression subnetwork is trained using a smooth $L1$ loss:

$$L_{dim}(\tilde{\mathbf{d}}, \mathbf{d}) = L_{smooth-L1}(\tilde{\mathbf{d}}, \mathbf{d}), \tag{2.4}$$

where $\tilde{\mathbf{d}}$ and $\mathbf{d}$ are the regression outputs and the corresponding targets for all three dimensions respectively. The total loss for this subnetwork is average loss over all positive anchors.

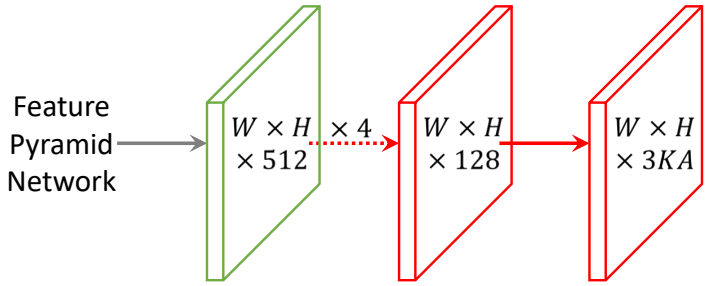


Figure 2.7: **Dimension regression subnetwork:** output channels accounts for three dimensions (height, width, length), per anchor, per object class.

The final loss for the entire network is a weighted sum of the three subnetwork losses:

$$L = L_{class} + \lambda_{reg} L_{reg} + \lambda_{dim} L_{dim}. \tag{2.5}$$

2.4 Ground Plane Identification by Polling

2.4.1 Database Creation

As highlighted in Figure 2.2, the ground plane polling block takes in a database of probable ground planes and outputs the "best fit" plane for each detection. This database of planes can be compiled either using heuristics based on the camera location, or by using 3D sensors like LiDARs to automatically fit planes to 3D data. In this study, we adopt the latter approach. In particular, we make use of the KITTI-15 dataset [39] comprising of 200 RGB images, LiDAR point clouds, and pixel-accurate semantic labels.

Algorithm 2.1 describes the approach to creating a ground plane database using such data. The procedure involves identifying and retaining LiDAR points corresponding to semantic classes of interest, and then iteratively fitting planes to these points using RANSAC. Since we are interested in generating a large number of diverse ground planes, we use a very small inlier threshold ($t = 2cm$) and a very high probability of success ($p = 0.999$). This produces

Algorithm 2.1 Pseudocode for creating a database of ground planes

Inputs: $\{im_i^{seg}$, ▷ ground truth semantic segmentation
$\{(X_i^j, Y_i^j, Z_i^j)\}_{j=1}^{M}$, ▷ LiDAR point cloud
$P_i\}_{i=1}^{N}$, ▷ camera calibration matrix
$semantic\_classes = \{ground, road, sidewalk, parking\}$
▷ semantic classes of interest

Outputs: $\{\pi_k\}_{k=1}^{K}$ ▷ database of ground planes

```
ground_planes ← {}
for i ← 1 to N do
    points ← {}
    for j ← 1 to M do
        (x_i^j, y_i^j) ← project((X_i^j, Y_i^j, Z_i^j), P_i)
                                   ▷ project LiDAR point to image plane
        if im_i^seg(x_i^j, y_i^j) ∈ semantic_classes then
            points ← points ⋃ {(X_i^j, Y_i^j, Z_i^j)}
        end if
    end for
    while |points| ≥ 3 do
        π, inliers ← RANSAC(points)
        ground_planes ← ground_planes ⋃ {π}
        points ← points \ inliers
    end while
end for
return ground_planes
```

approximately 22k ground plane candidates.

2.4.2 Ground Plane Polling

Given a ground plane and 2D keypoints $\mathbf{x}_l$, $\mathbf{x}_m$, and $\mathbf{x}_r$ of an object, it is straightforward to obtain the corresponding 3D keypoints lying on the plane. This is done by backprojecting a ray from the camera center through each keypoint, and finding its point of intersection with the plane. The backprojected ray $\mathbf{r} = (r_1, r_2, r_3, r_4)^T$ for a 2D keypoint $\mathbf{x} = (x, y)^T$ is created using the camera projection matrix,

$$\mathbf{r} = P^{+}[x, y, 1]^T, \tag{2.6}$$

and the corresponding 3D keypoint on the plane $\pi = (a,b,c,d)^T$ is given by

$$\mathbf{X}^{\pi} = s[r_1, r_2, r_3]^T, \tag{2.7}$$

where the scalar s is defined as follows

$$s = \frac{-dr_4}{ar_1 + br_2 + cr_3}. \tag{2.8}$$

This results in the 3D keypoints $\mathbf{X}_l^{\pi}$, $\mathbf{X}_m^{\pi}$, $\mathbf{X}_r^{\pi}$ corresponding to 2D keypoints $\mathbf{x}_l$, $\mathbf{x}_m$, $\mathbf{x}_r$, lying on the plane π.

Unlike the other three keypoints, the desired point $\mathbf{X}_t^{\pi}$ corresponding to the 2D keypoint $\mathbf{x}_t$ does not lie on the plane π, but is rather a point on the line through $\mathbf{X}_m^{\pi}$ along the normal to the plane π. Since the desired 3D keypoint lies on both the backprojected ray $\mathbf{r}$ and the plane normal through $\mathbf{X}_m^{\pi}$, we could ideally calculate it from the intersection of these two lines. In practice, however, these are *skew* lines that do not intersect. We therefore resort to an approximation, where $\mathbf{X}_t^{\pi}$ is designated as the point on the plane normal through $\mathbf{X}_m^{\pi}$ that is closest in distance to the ray $\mathbf{r}$. If $\mathbf{n_r}$ and $\mathbf{n}_{\pi}$ are the unit vectors representing the directions of the ray $\mathbf{r}$ and the plane normal respectively, the desired point can be calculated as follows:

$$\mathbf{X}_t^{\pi} = \mathbf{X}_m^{\pi} - \frac{\mathbf{X}_m^{\pi} \cdot \big(\mathbf{n_r} \times (\mathbf{n}_{\pi} \times \mathbf{n_r})\big)}{\mathbf{n}_{\pi} \cdot \big(\mathbf{n_r} \times (\mathbf{n}_{\pi} \times \mathbf{n_r})\big)} \mathbf{n}_{\pi}. \tag{2.9}$$

To find out how well a ground plane π fits the predicted 2D keypoints and 3D dimensions of an object, we propose an approach based on the construction of triplets comprised of two 3D keypoints, and the estimated length of the line segment joining them. Four 3D keypoints result in a total of ${}^4C_2 = 6$ unique pairs of keypoints, and therefore 6 total triplets $\{(\mathbf{X}_i^{\pi}, \mathbf{X}_{i'}^{\pi}, l_i)\}_{i=1}^{6}$. Figure 2.8 depicts these combinations. Note that the lengths of the line segments are determined using the 3D box dimensions predicted by the network.

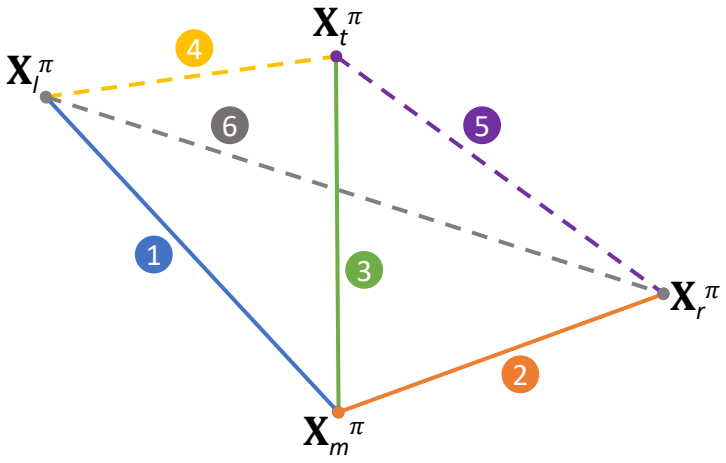


Figure 2.8: Illustration of the 6 unique combinations of 3D keypoints and the line segments joining them.

Each triplet $(\mathbf{X}_i^{\pi}, \mathbf{X}_{i'}^{\pi}, l_i)$ is associated with a residual error defined by:

$$e_i^{\pi} = \left| ||\mathbf{X}_i^{\pi} - \mathbf{X}_{i'}^{\pi}|| - l_i \right|, \tag{2.10}$$

and the total residual error e^{π} for a plane is the sum of the six individual errors. Finally, the "best fit" ground plane for a given object from a database of probable ground planes $\{\pi_k\}_{k=1}^{K}$ is determined to be

$$\pi^* = \underset{\pi_k \in \{\pi_k\}_{k=1}^{K}}{\arg\min} \sum_{i=1}^{6} e_i^{\pi_k}. \tag{2.11}$$

The residual errors tied to the three dimensions of the bounding box ensure that a plane that produces a box of reasonable dimensions is chosen. The other three errors corresponding to the face diagonals of the bounding box promote planes that result in boxes with nearly orthogonal edges. In our experiments, directly enforcing orthogonality or near-orthogonality led to most probable planes being discarded, thereby causing performance degradation.

Knowing the "best fit" plane π^*, we first discard one of the two keypoints $(\mathbf{X}_l^{\pi^*}, \mathbf{X}_r^{\pi^*})$ corresponding to the width of the bounding box. This is easily discerned from the predicted coarse orientation of the object. Doing so allows us to ensure orthogonality between adjacent sides,

while retaining the predicted orientation (yaw). Next, we construct the desired 3D cuboid by utilizing $\mathbf{X}_l^{\pi^*}$ and the retained keypoint, the estimated 3D dimensions and the predicted orientation class. To account for a large database of ground planes, potentially large number of objects per image, and to leverage the GPU for matrix multiplications, we implement the entire polling procedure as a layer in our network. This is referred to as the ground plane polling (GPP) layer.

2.5 Experimental Evaluation

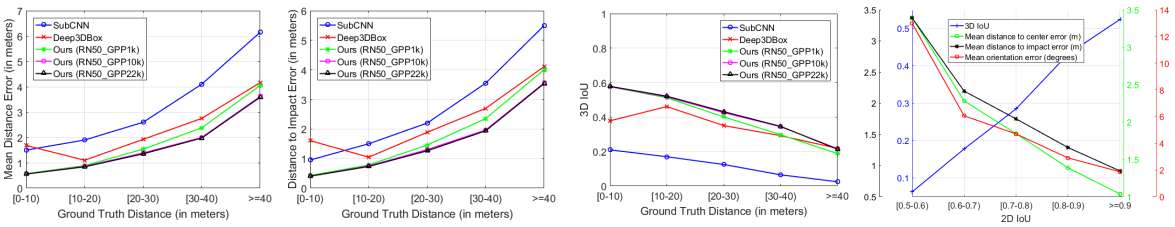
2.5.1 Implementation Details

Training

We use $\gamma = 2$ and $\alpha = 0.25$ for the focal loss presented in Equation 2.1. The backbone of our network is pre-trained on ImageNet1k. The total loss depicted in Equation 2.5 is fairly robust to the choice of hyperparamters λ_{reg} and λ_{dim} provided the network is trained for enough epochs. We set $\lambda_{reg} = \lambda_{dim} = 1$ for simplicity. The entire network is trained using mini-batch gradient descent using an Adam optimizer with learning rate 0.00001, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We use a batch size of two images and train for a total of 70 epochs on a single GPU. We apply random rotations in the image plane, translations, shears, scalings and horizontal flips to input images for robustness against small geometric transformations. Care was taken to apply the same transformations to all keypoints and also change the corresponding orientation class of objects when necessary. Additionally, the brightness, contrast, saturation and hue of input images were randomly perturbed. We noticed that data augmentation greatly helped with the robustness of our keypoint predictions.

Inference

During inference, we add the NMS (non-maximum suppression) and GPP layers to the end of the subnetwork outputs at each level of the FPN. To ensure fast operation, we only decode



(a) Mean error in estimating the 3D box center

(b) Mean error in estimating the closest point of the 3D box

(c) 3D IoU between the predicted and ground truth 3D boxes

(d) 3D metrics as a function of 2D IoU for our approach.

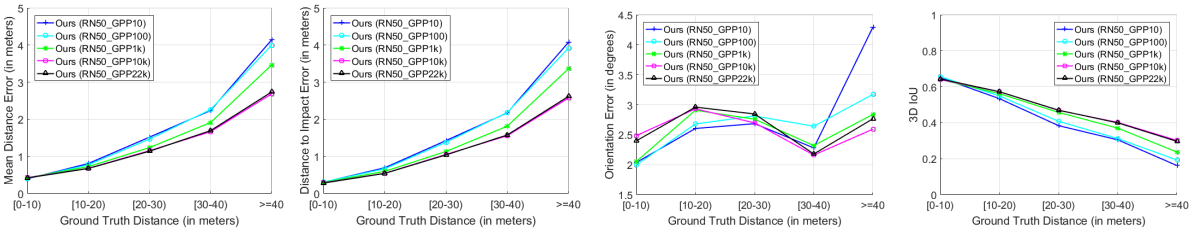
Figure 2.9: Experiments related to 3D metrics of interest for different approaches on KITTI cars using the validation set provided in [31, 35].

box predictions from at most 1k top-scoring predictions per FPN level, after thresholding detector confidence at 0.05. The top predictions from all levels are merged and non-maximum suppression with a threshold of 0.5 is applied to yield the final 2D detections. These detections and their corresponding keypoints, dimensions and orientations are passed on to the GPP layer, which outputs the 3D keypoints and "best fit" ground plane for each detection. The desired 3D bounding box is then constructed using these outputs.

2.5.2 Results

Evaluation of 3D Bounding Box Metrics

The KITTI detection and orientation estimation benchmark [40] for cars only evaluates the partial pose of vehicles. To provide a more complete evaluation and comparison with other monocular methods, we carry out analogous experiments to the ones proposed in [35]. In particular, we plot three metrics of interest as a function of the distance of an object from the camera. The first metric is the average error in estimating the 3D coordinate of the center of objects. The second metric is the average error in estimating the closest point of the 3D bounding box from the camera. This metric is important for driving scenarios where the system needs to avoid hitting obstacles and is closely related to the *time to collision* metric. The last metric is the standard 3D intersection over union (3D IoU) that depends on all factors of the 3D bounding box.



(a) Mean error in estimating the 3D box center

(b) Mean error in estimating the closest point of the 3D box

(c) Mean orientation (yaw) error

(d) 3D IoU between predicted and ground truth 3D boxes

Figure 2.10: Effect of ground plane database size on 3D bounding box metrics for KITTI cars on a common validation set.

In keeping with the original experiment, we factor away the 2D detection performance by considering only those detections that result in an IoU ≥ 0.7 with the ground truth 2D box. This removes the effects of 2D detection performance and enables a side-by-side comparison of how well each method performs on the desired 3D metrics. To make the comparison fair, we use the same *train-val* split provided in [31], where the KITTI training dataset comprising of 7481 images is split into *train* and *val* sets consisting of 3619 and 3799 images respectively. Each method is trained on the *train* set and is evaluated on the *val* set. We compare our method to current (Deep3DBox [35]) and previous (SubCNN [31]) state-of-the-art monocular methods with publicly available results and/or code. We compare three variants of our method: RN50_GPP1k, RN50_GPP10k and RN50_GPP22k, all three of which have a ResNet50 backbone, and differ only by the size of the ground plane database used in the GPP layer (1k, 10k and 22k planes respectively). As Figures 2.9a, 2.9b and 2.9c show, all three variants of our method outperform the other two methods on nearly all data points in the plots despite being an order of magnitude faster. In addition to these comparative results, we also plot the 3D metrics of interest as a function of 2D IoU for our model in Figure 2.9d. As 2D IoU with the ground truth increases, we observe monotonic improvements in all 3D metrics. This implies that a better 2D detector could result in better 3D pose.

Effect of Ground Plane Database Size

To quantify the effect of the number of ground planes used in the database, we provide comparative analysis of different variants of our method using the same 3D metrics as before. Additionally, we also plot the orientation error (in degrees) averaged across all samples as a function of distance to the camera. Crucially, we create a different *train-val* split to ensure that the network does not overfit, leading to a more accurate comparison. The resulting *train* and *val* sets consist of 6373 and 1108 images respectively. Each set is comprised of images from disparate drives with no overlap.

As mentioned in Section 2.4.1, our complete database is made of 22k candidate ground planes. To create databases of smaller sizes, we rank planes based on the number of inliers associated with them (see Algorithm 2.1), and choose planes in a "top-k" fashion. While retaining the same ResNet50 backbone and subnetworks, we provide comparisons between databases of size 10, 100, 1k, 10k and 22k. We refer to these 5 variants as RN50_GPP10, RN50_GPP100, RN50_GPP1k, RN50_GPP10k and RN50_GPP22k respectively.

Figure 2.10 depicts the comparison between these variants on all four metrics. As expected, we see an increase in performance over all metrics as the size of the ground plane database is increased. This effect is exacerbated as the distance of objects from the camera increases. The plots also seem to indicate that doubling the number of planes from 10k to 22k results in minor performance improvement, and even degradation in some cases. This hints at a case of diminishing returns beyond a database with 10k ground planes. With this in mind, we use a database of 10k ground planes as our default choice in other experiments.

KITTI 2D Detection and Orientation Benchmark

The official 3D metric of the KITTI dataset is Average Orientation Similarity (AOS), which is defined in [27] and multiplies the average precision (AP) of the 2D detector with the average cosine distance similarity for the azimuth. Hence, AP is by definition the upper bound of AOS. Our results are summarized in Table 2.1, along with other top entries on the KITTI leaderboard. Since the AP and AOS metrics are heavily influenced by the 2D detection

Table 2.1: **Results on the KITTI benchmark:** Comparison of the Average Orientation Similarity (AOS), Average Precision (AP) and Orientation Score (OS) for cars. Our runtimes are reported using a RTX 2080 GPU.

Method	Runtime	Easy			Moderate			Hard		
	(seconds)	AOS	AP	OS	AOS	AP	OS	AOS	AP	OS
DeepMANTA [32][1]	2.00	96.32%	96.40%	0.9991	89.91%	90.10%	0.9979	80.55%	80.79%	0.9970
Mono3D [34]	4.20	91.01%	92.33%	0.9984	86.62%	88.66%	0.9769	76.84%	78.96%	0.9731
SubCNN [31]	2.00	90.67%	90.81%	0.9984	88.62%	89.04%	0.9952	78.68%	79.27%	0.9925
Deep3DBox [35]	1.50	92.90%	92.98%	0.9991	88.75%	89.04%	0.9967	76.76%	77.17%	**0.9947**
Shift R-CNN [41]	0.25	90.27%	90.56%	0.9968	87.91%	88.90%	0.9889	78.72%	79.86%	0.9857
MonoPSR [42]	0.20	89.88%	90.18%	0.9967	87.83%	88.84%	0.9886	70.48%	71.44%	0.9866
Ours (RN50_GPP10k)	0.21	89.42%	89.61%	0.9979	86.08%	87.02%	0.9892	76.47%	77.62%	0.9852
Ours (RN101_GPP10k)	0.24	89.67%	89.84%	0.9981	86.63%	87.52%	0.9898	77.20%	78.36%	0.9852
Ours (VGG16_GPP10k)	0.19	89.17%	89.26%	0.9990	87.16%	87.56%	0.9954	77.65%	78.29%	0.9918
Ours (VGG19_GPP10k)	0.23	90.35%	90.42%	**0.9992**	87.96%	88.23%	**0.9969**	78.57%	79.00%	0.9946
Ours (VGG19_GPP1k)	0.11	90.12%	90.22%	0.9989	87.69%	88.06%	0.9958	78.41%	78.95%	0.9932
Ours (RN50_FAST)	**0.05**	88.86%	89.13%	0.9970	84.53%	85.67%	0.9867	75.18%	76.61%	0.9813
Ours (VGG19_FAST)	0.07	89.87%	90.08%	0.9977	87.01%	87.59%	0.9934	77.59%	78.31%	0.9908

[1] Uses additional keypoint labels and 3D CAD models not available to other methods.

performance, we additionally list the ratio of AOS over AP for each method as done in previous works. This ratio is representative of how each method performs only on orientation estimation, while factoring out the 2D detector performance. This score is referred to as the Orientation Score (OS), which represents the error $(1 + cos(\Delta\theta))/2$ averaged across all examples.

A cursory glance at Table 2.1 indicates that our model is mostly within a few precision points of other methods in terms on AP and AOS. This is expected given that our method is the only single-stage approach that does not require object proposals of any kind, thereby resulting in a speedup not possible with other multi-stage methods. More importantly, our best performing model (VGG19_GPP10k) only falls behind DeepMANTA [32] by a small amount on the OS score, while beating out or remaining at par with other methods. We manage to do so at a fraction of the computational cost, and without requiring additional annotations associated with keypoints, part labels, and part visibility as needed in DeepMANTA. Additionally, our method is the only one that does not rely on computing additional features such as disparity, semantic segmentation, instance segmentation, point clouds etc., and does not need multi-stage processing as in [32, 34, 35, 41, 42].

Effect of Different Backbones

To observe the effect of different backbone sizes and architectures on the final result, we

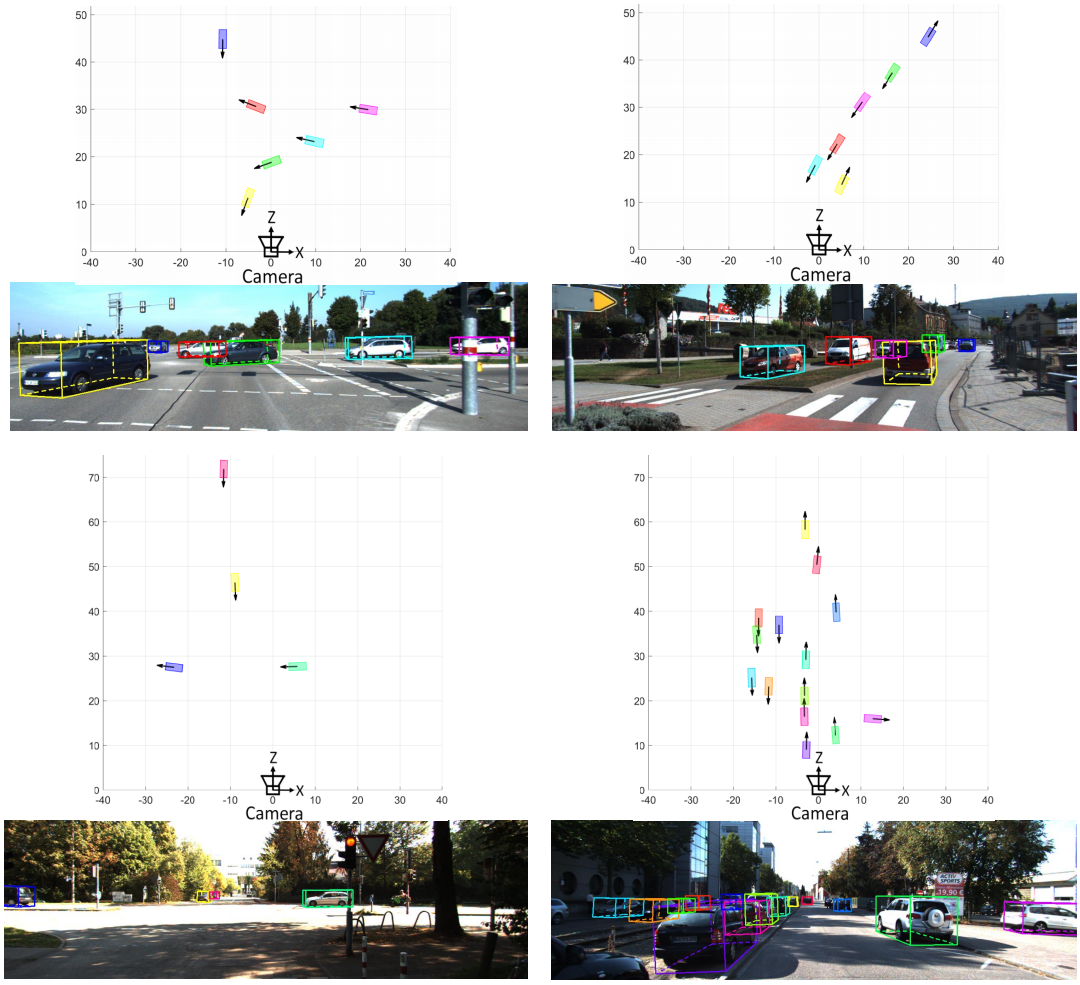


Figure 2.11: **Qualitative results on the KITTI test set (VGG19_GPP10k):** We show the bird's eye view of the predicted boxes (top), and their corresponding projections onto the input image (bottom) for four different scenes.

provide results for four popular backbone choices on the KITTI benchmark in Table 2.1. We refer to these variants as RN50_GPP10k, RN101_GPP10k, VGG16_GPP10k and VGG19_GPP10k, each named after the corresponding backbone architecture used. The FPN for the RN101 (ResNet101) backbone [38] is constructed using the convolutional features C_3, C_4 and C_5, and the FPN for the two VGG architectures [43] are constructed using the features from the maxpooling layers P_3, P_4, and P_5 that follow the convolutional blocks C_3, C_4, and C_5 respectively. Surprisingly, the smaller VGG backbones yield better results, especially for orientation-related metrics. The

larger VGG19 backbone even results in superior AP across the board, and is therefore our best performing model. The RN101 backbone leads to a meager improvement in comparison to the smaller RN50 backbone, implying that adding more layers does not necessarily justify the return. We believe that the superior performance of the VGG variants is most likely explained by their use of smaller convolutional kernels, which preserves smaller details, resulting in better keypoint predictions. Consequently, better keypoint predictions result in better orientation estimates, and hence better orientation scores.

In addition to the backbones listed above, we also train a significantly smaller networks by halving the number of channels per layer in each subnetwork, and per pyramid level. Using a ResNet50 backbone and a database of 1k ground planes, the RN50_FAST network runs at 20 fps with only a small drop in overall performance (see Table 2.1). VGG19_FAST is constructed in the same manner as our RN50_FAST variant, with the only difference being the backbone architecture. Similar to our full size models, the VGG19_FAST variant outperforms the RN50_FAST variant while incurring a small overhead. More importantly, this smaller model is on par or comparable in performance to full size variants RN50_GPP10k and RN101_GPP10k, while being considerably smaller and faster. These experiments further indicate the advantage of VGG backbones over ResNet backbones for the task at hand.

Qualitative Results on the KITTI Test Set

In addition to the quantitative results presented, we also show some qualitative results of our method on the KITTI test set in Figures 2.11, 2.12, and 2.13. All figures contain both the bird's eye view of all predicted boxes, and their corresponding projections into the image.

Figure 2.11 depicts exemplar results of our best performing model (VGG19_GPP10k) on the KITTI test set. Figure 2.12 shows comparative results of the VGG19_GPP10k variant with corresponding ground truth boxes on a validation set. Results indicate that our model produces boxes that largely agree with the ground truth in terms of location and orientation. Finally,

Figure 2.13 compares the output of three different variants of our model: VGG19_GPP10k, VGG19_GPP1k, and VGG19_FAST. The first two variants tend to produce similar results closer to the camera, while being inconsistent farther away from the camera. This observation concurs with our experiment on the effect of ground plane database size.

2.6 Chapter Summary

In this chapter, we introduce an approach to monocular 3D object detection by leveraging ground planes. This Ground Plane Polling (GPP) method works by merging 2D attributes like keypoints and coarse orientations with 3D information from probable ground plane configurations. By doing so, we also ensure that our network only predicts those entities that are known to generalize well across different conditions and datasets. Adapting to a new dataset would only involve reassessing the database of 3D ground planes. Additionally, our method produces a redundant set of cues and relies on identifying a suitable consensus set within, thereby resulting in robustness to individual errors and outliers. We have shown that our GPP approach outperforms other popular monocular approaches in terms of localization and orientation estimation, while remaining comparable to other methods in 2D detection performance, albeit with a significantly reduced inference time. Future work entails adopting our approach to recent two-stage detectors, and also conducting experiments to analyze its robustness to prediction errors and generalizability to more contemporary datasets like [44, 45].

2.7 Acknowledgements

Chapter 2, in full, is a reprint of the material as it appears in the IEEE Transactions on Intelligent Vehicles (2020), by Akshay Rangesh, and Mohan M. Trivedi. The dissertation author was the primary investigator and author of this paper.

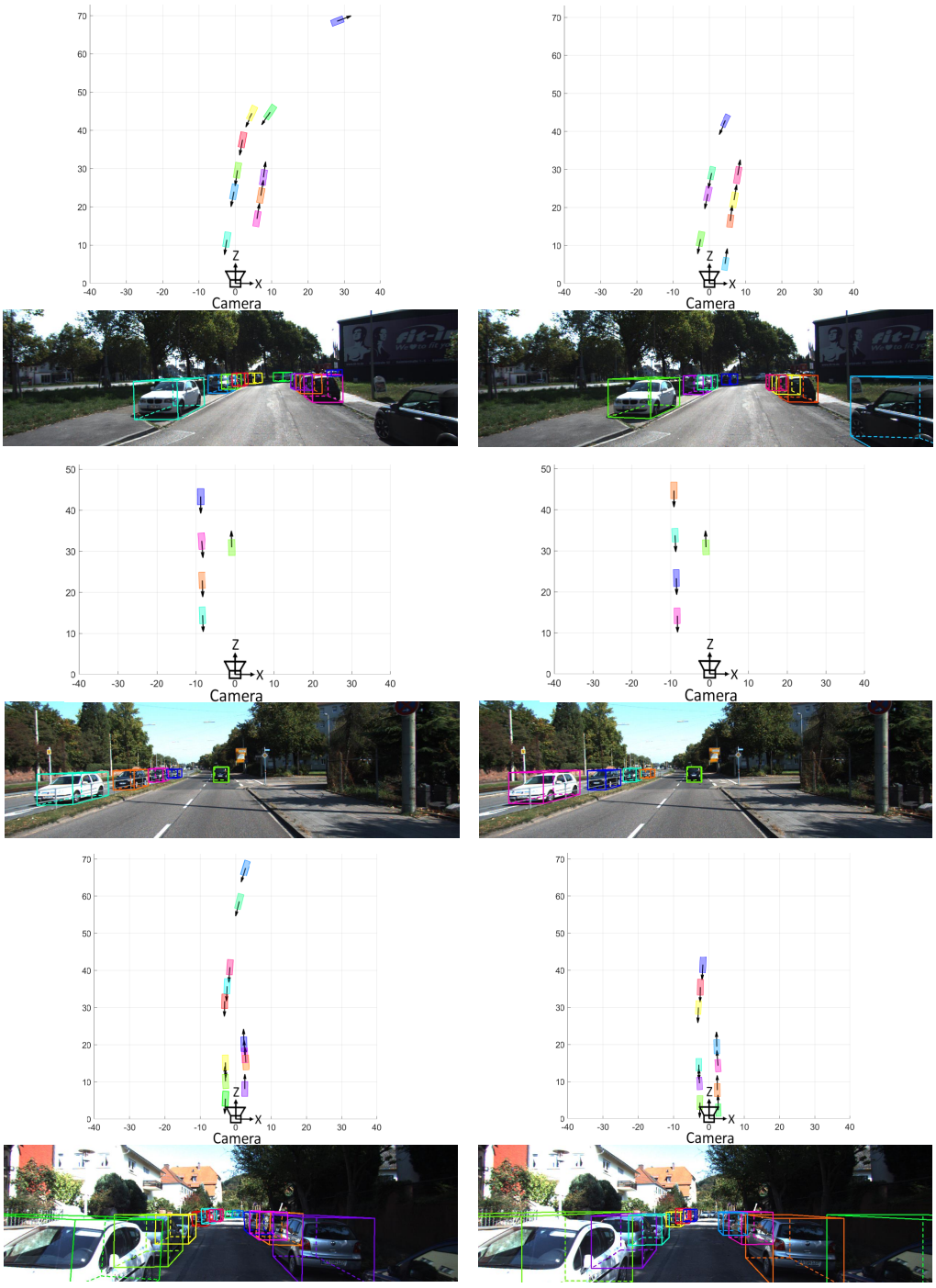


Figure 2.12: **Qualitative results on the validation set (VGG19_GPP10k):** We show the bird's eye view of the boxes (top), and their corresponding projections onto the input image (bottom) for both our model's predictions (left column) and the ground truth (right column).

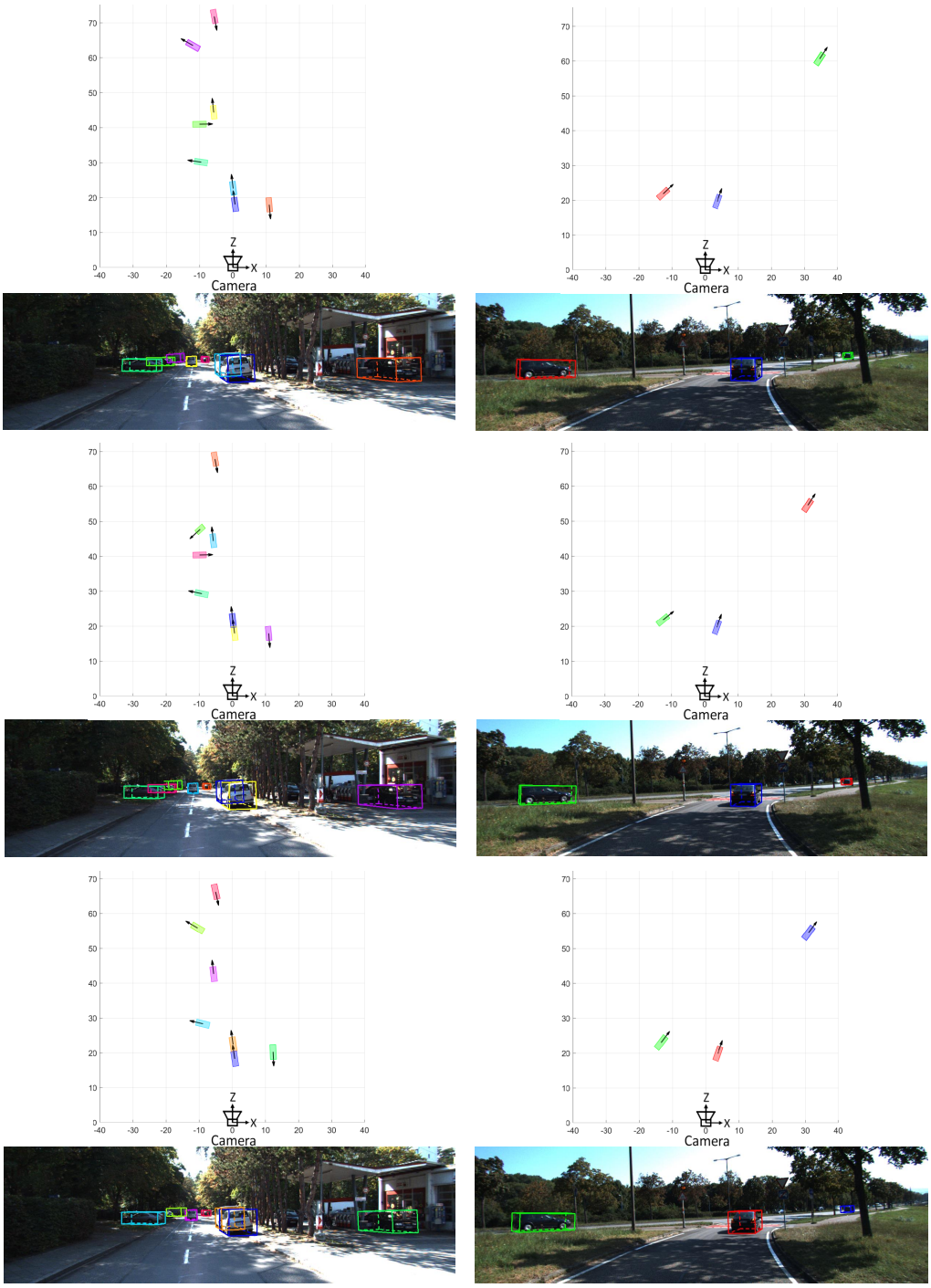


Figure 2.13: **Qualitative results on the KITTI test set:** We show the bird's eye view of the predicted boxes (top), and their corresponding projections onto the input image (bottom) for three variants of our model: VGG19_GPP10k (top row), VGG19_GPP1k (middle row), and VGG19_FAST (bottom row).

Chapter 3

LaneAF: Robust Multi-Lane Detection with Affinity Fields

This chapter presents an approach to lane detection involving the prediction of binary segmentation masks and per-pixel affinity fields. These affinity fields, along with the binary masks, can then be used to cluster lane pixels horizontally and vertically into corresponding lane instances in a post-processing step. This clustering is achieved through a simple row-by-row decoding process with little overhead; such an approach allows LaneAF to detect a variable number of lanes without assuming a fixed or maximum number of lanes. Moreover, this form of clustering is more interpretable in comparison to previous visual clustering approaches, and can be analyzed to identify and correct sources of error. Qualitative and quantitative results obtained on popular lane detection datasets demonstrate the model's ability to detect and cluster lanes effectively and robustly. Our proposed approach sets a new state-of-the-art on the challenging CULane dataset and the recently introduced Unsupervised LLAMAS dataset.

Code: https://github.com/sel118/LaneAF

3.1 Introduction

Lane detection is the process of automatically perceiving the shape and position of marked lanes and is a crucial component of autonomous driving systems, directly influencing the guidance and steering of vehicles while also aiding the interaction between numerous agents on the road. As the number of drivers on the roads has increased, autonomous driving systems have received considerable attention in the automotive and tech industries as well as in academia [46]. According to the Insurance Institute for Highway Safety (IIHS), in the US alone, car accidents claimed 36,560 lives in 2018, underscoring the importance of any technology that can help prevent crashes.

Since roads commonly have different types of lane lines (solid white, broken white, solid yellow, etc.), each of which have specific implications with regards to how vehicles may interact with them, automated lane detection systems can also help alert drivers when there are changes in lane topology on the road. Furthermore, there are several factors that make lane detection a challenging task. Firstly, there is a wide variety of road infrastructure in use around the world. Additionally, the lane detection system must be able to identify instances where lanes are ending, merging, and splitting. Finally, the lane detection system must possess the ability to discern worn or unclear lane markings. Precise detection of lanes also enable more robust trajectory prediction of surrounding vehicles; as discussed in [7], this is critical for successful path planning in autonomous driving. Therefore, while lane detection is a significant and complex task, it is a key factor in developing any autonomous vehicle system.

While binary classification is used for the detection of lanes in our approach, a limitation of this type of classification is that it produces a single-channel output, which does not allow for the identification of separate lane entities. To dissociate different lane instances, we propose a novel clustering scheme based on affinity fields (see Figure 3.1). Affinity fields were originally introduced in [47] for the purpose of multi-person 2D pose estimation, and are comprised of unit

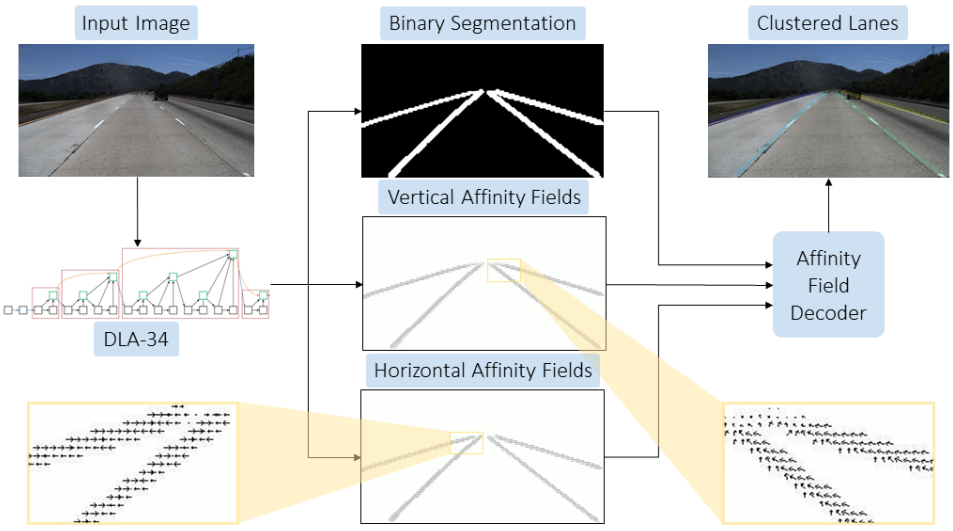


Figure 3.1: In our approach, we propose to train a model that outputs binary segmentation masks and affinity fields, which can then be decoded together to produce multiple lane instances. This is opposed to the standard approach to (anchor-free) lane detection that treats each lane as a separate class and trains a model to perform multi-class segmentation.

vectors that encode location and orientation. This technique was also used for the detection of hands inside a vehicle, as demonstrated in [48]. In this work, we have defined two types of affinity fields, the horizontal affinity field (HAF) and vertical affinity field (VAF). It is these affinity fields that enable unique lane instances to be identified and segmented. Since these affinity fields are present wherever there are foreground lane pixels, they are not bound to a pre-determined number of lanes. The model is therefore agnostic to the number of lanes present on the road.

The main contributions of this work are as follows:

1. We show that using an off-the-shelf convolutional neural network (CNN) backbone [49] that intrinsically aggregates and refines multi-scale features can result in superior performance when compared to other bespoke architectures and losses previously proposed for lane detection.

2. We propose affinity fields that are suitable for clustering and associating pixels belonging to amorphous entities like lanes.

3. We detail the procedure and losses to train models that predict binary segmentation masks and affinity fields for the purpose of lane instance segmentation.

4. We introduce efficient methods for generating and decoding such affinity fields into an unknown number of clustered lane instances.

3.2 Related Research

Lane detection has traditionally been tackled by feature-based approaches [50] which then evolved to model-based approaches to detect lane boundaries. However, these are not practical in real world scenarios since they require ideal road scenes to work effectively. Currently, data-driven approaches are commonly used to detect both lane boundaries as well as lane regions. While several shortcomings of the traditional lane detection methods (i.e. lane segmentation via hand-crafted features) have been resolved with more robust methods in recent years, there is still room for improvement. In more recent times, deep learning and large-scale datasets have provided solutions to many of these issues. However, lane detection in unconstrained environments and complex scenarios remain a challenge.

Lane detection nowadays is typically modelled as a semantic segmentation problem to extract features using deep learning methods. New approaches tackle lane detection as a multi-class segmentation problem, where each lane forms a separate class. Some of these approaches include: [51], [52], [53], [54], [55], and [56]. In [53], the authors combine a recurrent neural network (RNN) with a CNN for lane prediction and detection. The use of an embedding loss was introduced in [54] which uses generative adversarial networks (GANs) to better preserve the structure of lanes and to mitigate the problem of complex post-processing for the output of semantic segmentation; 96% accuracy on the TuSimple dataset was obtained. In [56], a sequential prediction network has been used to avoid heuristic-based clustering post-processing. Another network architecture was presented in [57] with two elements: a deep network which

generates weighted pixel coordinates in addition to a differentiable weighted least-squares fitting module. In [58], the authors introduced Self Attention Distillation (SAD) loss to avoid models that propagate data sequentially and to decrease inference time. However, the fully connected layer that the SAD model employs is computationally expensive and cannot adapt to any number of lanes.

Other lane detection approaches choose to first perform binary segmentation of all lanes, followed by a clustering stage to separate each individual lane instance as in [59], [60], and [61]. Instance segmentation is usually approached with the use of complex pipelines; however, many powerful approaches and research were put to come up with better performance techniques including the approach presented in [62], where they used an end-to-end convolutional neural network to tackle the problem that was inspired by the classical watershed transform. Another method toward instance segmentation was based on using a fully convolutional network to predict semantic labels along with depth and an instance-based encoding. This was implemented by using each pixel's direction toward its corresponding instance center; with the help of low-level computer vision techniques, impressive scene understanding by predicting pixel-wise depth, semantics, and instance-level direction cues was achieved [63]. Lane detection is posed as an instance segmentation problem in [59] so that each lane can be detected in an end-to-end manner, adapting to changing numbers of lanes on the road. In [60], a combination of instance segmentation and classification was used as an end-to-end deep learning real-time method to avoid reliance on two-step detection networks. Although recent methods of lane detection show high accuracy when applied to the popular published datasets, some drawbacks of these current methods are that they are not robust when encountering occlusion and that they require a fixed number of lanes in a scene; thus, they cannot work for a random number of lanes present on the road. Acknowledging this problem in [61], the authors use a key points estimation approach to allow for lane detection of an arbitrary numbers of lanes regardless of orientation.

More recently, some approaches have modelled lane detection as an anchor-based object

detection problem such as [64], [65], [66], [67], and [50]. In [67], a spatio-temporal deep learning method was proposed to mitigate the errors that can occur when experiencing harsh weather or other complex problems in the road, jeopardizing the accuracy of detecting a lane in the scene. Meanwhile, in [64], lane markers were tracked temporally. Additionally, [66] presents an anchor-based single-stage deep lane detection model using anchors for feature pooling. In [65], the authors developed 3D-LaneNet, a network that predicts the 3D layout of lanes using a single image. A combination of LiDAR and camera sensors were used in [68] for their network to obtain accurate lane detection in 3D space directly.

3.3 Methodology

Our proposed methodology involves a feed-forward CNN that is trained to predict binary lane segmentation masks and per-pixel affinity fields. More specifically, the model is trained to predict two affinity fields, which we call the horizontal affinity field (HAF) and vertical affinity field (VAF), respectively. Affinity fields can be thought of as vector fields that map any 2D location on the image plane to a unit vector in 2D. A unit vector in the VAF encodes the direction in which the next set of lane pixels above it is located. On the other hand, a unit vector in the HAF points toward the center of the lane in the current row, thereby allowing us to cluster lanes of arbitrary widths. These two affinity fields, in conjunction with the predicted binary segmentation, can then be used to cluster foreground pixels into lanes as a post-processing step. In the next few subsections, we discuss each individual block in our proposed approach.

3.3.1 Network Backbone

Recent lane detection approaches have made use of a variety of backbone architectures, but most popular among them are usually the ResNet family of architectures [69], ENet [51], and ERFNet [70]. Although these architectures have proven benefits across a variety of tasks, we

Algorithm 3.1 Creating affinity fields from ground truth data

Inputs:
$SEG(H \times W)$: ground truth segmentation
l_{max}: maximum number of lanes

$HAF, VAF \leftarrow zeros(H, W, 2)$ ▷ initialize affinity fields
for $l \leftarrow 1$ **to** L **do** ▷ go through each lane
  $prev\_cols \leftarrow nonzero(SEG[H, :] == l)$ ▷ initialize
  /* *row-by-row, from bottom to top* */
  for $y \leftarrow H - 1$ **to** 1 **do**
    $cols \leftarrow find(SEG[row, :] == l)$ ▷ find lane pixels
    /* *horizontal affinity field* */
    for x **in** $cols$ **do**
      $HAF[y, x] \leftarrow \vec{H}_{gt}(x, y)$ ▷ Eq. 3.1
    end for
    /* *vertical affinity field* */
    for x **in** $prev\_cols$ **do**
      $HAF[y+1, x] \leftarrow \vec{V}_{gt}(x, y+1)$ ▷ Eq. 3.2
    end for
    $prev\_cols \leftarrow cols$
  end for
end for
return HAF, VAF

believe that more recent developments in the field can be leveraged for lane detection. To this end, we make use of the DLA-34 backbone presented in [49].

The DLA family of models make use of deep layer aggregation, which unifies semantic and spatial fusion for better localization and semantic interpretation. In particular, this architecture extends densely connected networks [71] and feature pyramid networks with hierarchical and iterative skip connections that deepen the representation and refine resolution. They employ two forms of aggregation: iterative deep aggregation (IDA), focusing on fusing resolutions and scales, and hierarchical deep aggregation (HDA), focusing on merging features from all modules and channels. These architectures also incorporate deformable convolution operations [72] that can adapt the spatial sampling grid for convolutions based on their inputs. We believe these are desirable properties for the tasks of lane detection and instance segmentation.

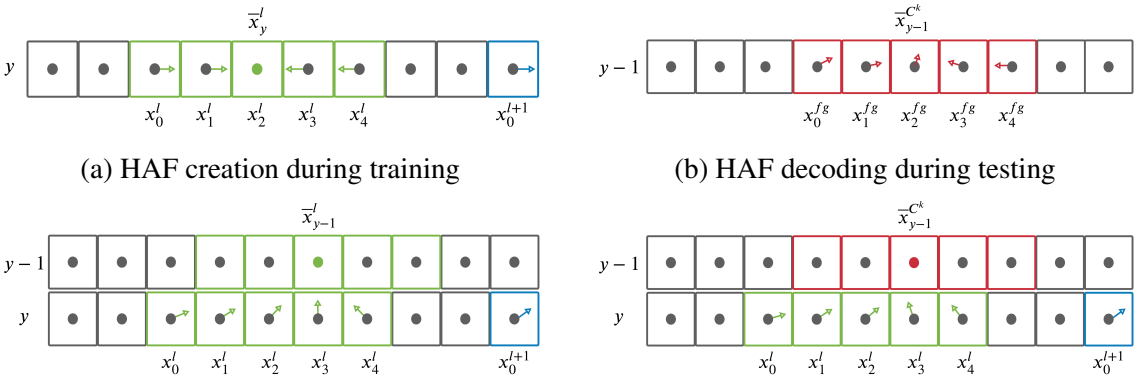
Algorithm 3.2 Decoding predicted affinity fields into lanes

Inputs:
$BW(H \times W)$: binary segmentation mask
$HAF(H \times W \times 2)$: horizontal affinity field
$VAF(H \times W \times 2)$: vertical affinity field
τ: clustering threshold

$SEG \leftarrow zeros(H, W)$ ▷ initialize segmentation output
$lane\_end\_points \leftarrow []$
▷ keeps track of the latest points added to each lane
$L \leftarrow 0$ ▷ initialize number of lanes to 0
/ row-by-row, from bottom to top */*
for $y \leftarrow H$ **to** 1 **do**
  $cols \leftarrow find(BW[row, :] > 0)$ ▷ find foreground pixels
  / cluster horizontally */*
  $clusters \leftarrow []$
  for x **in** $cols$ **do**
    $clusters.update(c^*_{haf}(x, y))$ ▷ Eq. 3.3
  end for
  / assign clusters to existing lanes */*
  for $l \leftarrow 1$ **to** L **do**
    if $d^*(l) <= \tau$ **then** ▷ error less than threshold (Eq. 3.5)
      $lane\_end\_points[l] \leftarrow c^*_{vaf}(l)$ ▷ Eq. 3.4,Eq. 3.6
      ▷ update latest points added to lane
      for x **in** c^*_{vaf}(l) **do**
        $SEG[y, x] \leftarrow l$ ▷ assign cluster to lane l
      end for
    end if
  end for
  / spawn new lanes with unassigned clusters */*
  for $cluster$ **in** $clusters$ **do**
    if $cluster$ **is not assigned then**
      $L \leftarrow L + 1$
      $lane\_end\_points[L] \leftarrow cluster$
    end if
  end for
end for
return SEG

3.3.2 Affinity Fields

In addition to binary lane segmentation masks, our model is trained to predict horizontal and vertical affinity fields (HAFs and VAFs respectively). For any given image, the HAF and VAF



(a) HAF creation during training

(b) HAF decoding during testing

(c) VAF creation during training

(d) VAF decoding during testing

Figure 3.2: Illustrations of HAF and VAF creation and decoding processes during training and testing respectively.

can be thought of as vector fields $\vec{H}(\cdot,\cdot)$ and $\vec{V}(\cdot,\cdot)$, that assign a unit vector to each (x,y) location in the image. As we alluded to earlier, the HAF enables us to cluster lane pixels horizontally and the VAF vertically. With the predicted affinity fields and binary mask, clustering lane pixels is achieved through a simple row-by-row decoding process from bottom to top. The rest of this subsection provides details on how to create such affinity fields using the ground truth and how to use the predicted affinity fields to decode individual lanes.

Creating HAFs and VAFs

Affinity fields are created using ground truth segmentation masks on the fly as detailed in Algorithm 3.1. This proceeds row-by-row, from bottom to top.

For any row y in the image, the HAF vectors are computed for each lane point (x^l_i, y) using the ground truth vector field mapping $\vec{H}_{gt}(\cdot,\cdot)$ as follows:

$$\begin{aligned}\vec{H}_{gt}(x^l_i, y) &= \left(\frac{\bar{x}^l_y - x^l_i}{|\bar{x}^l_y - x^l_i|}, \frac{y-y}{|y-y|}\right)^\intercal \\ &= \left(\frac{\bar{x}^l_y - x^l_i}{|\bar{x}^l_y - x^l_i|}, 0\right)^\intercal,\end{aligned} \tag{3.1}$$

where $\bar{x}^l_y$ is the mean x-coordinate of all points belonging to lane l in row y. This process is

illustrated in Figure 3.2a, where pixels in green and blue represent points belonging to lanes l and $l+1$ respectively.

Similarly, the VAF vectors are computed for each lane point (x_i^l, y) in row y using the ground truth vector field mapping $\vec{V}_{gt}(\cdot,\cdot)$ as follows:

$$\begin{aligned}\vec{V}_{gt}(x_i^l, y) &= \left(\frac{\bar{x}_{y-1}^l - x_i^l}{|\bar{x}_{y-1}^l - x_i^l|}, \frac{y-1-y}{|y-1-y|}\right)^\intercal \\ &= \left(\frac{\bar{x}_{y-1}^l - x_i^l}{|\bar{x}_{y-1}^l - x_i^l|}, -1\right)^\intercal,\end{aligned} \tag{3.2}$$

where $\bar{x}_{y-1}^l$ is the mean x-coordinate of all points belonging to lane l in row $y-1$. This process is illustrated in Figure 3.2c, where pixels in green represent points belonging to lanes l. Note that unlike the HAF, unit vectors in the VAF point to the mean location of the lane in the previous row.

Decoding HAFs and VAFs

After a model is trained to predict the HAFs and VAFs detailed above, a decoding procedure is carried out to cluster foreground pixels into lanes during testing. This procedure is presented in Algorithm 3.2, and similarly operates row-by-row, from bottom to top.

Assuming $\vec{H}_{pred}(\cdot,\cdot)$ is the vector field corresponding to the predicted HAF, foreground pixels in a row $y-1$ are first assigned to clusters based on the following rule:

$$c_{haf}^*(x_i^{fg}, y-1) = \begin{cases} C^{k+1} & \text{if} \quad \vec{H}_{pred}(x_{i-1}^{fg}, y-1)_0 \leq 0 \\ & \quad \wedge \vec{H}_{pred}(x_i^{fg}, y-1)_0 > 0, \\ C^k & \text{otherwise,} \end{cases} \tag{3.3}$$

where $c_{haf}^*(x_i^{fg}, y-1)$ denotes the optimal cluster assignment for a foreground pixel $(x_i^{fg}, y-1)$; C^k and C^{k+1} denote two different clusters indexed by k and $k+1$ respectively. This assignment is illustrated in Figure 3.2b, where pixels in red are assigned the same cluster.

Next, these horizontal clusters are assigned to existing lanes indexed by l using the vector

field $\vec{V}_{pred}(\cdot,\cdot)$ corresponding to the VAF as follows:

$$c^*_{vaf}(l) = \arg\min_{C^k} d^{C^k}(l), \tag{3.4}$$

where

$$d^*(l) = \min_{C^k} d^{C^k}(l). \tag{3.5}$$

Here, $d^{C^k}(l)$ denotes the error of associating cluster C^k to an existing lane l:

$$\begin{aligned} d^{C^k}(l) = & \frac{1}{|C^k|} \sum_{i=0}^{|C^k|-1} \Big|\Big| (\bar{x}^{C^k}, y-1)^\intercal - (x^l_i, y)^\intercal \\ & - \vec{V}_{pred}(x^l_i, y) \cdot ||(\bar{x}^{C^k}, y-1)^\intercal - (x^l_i, y)^\intercal|| \Big|\Big|. \end{aligned} \tag{3.6}$$

We illustrate this process in Figure 3.2d, where the cluster in red is assigned to the existing lane in green. By repeating the above steps row-by-row starting from the bottom and working to the top, we are able to assign every foreground pixel to their respective lanes.

3.3.3 Losses

To train the proposed model, we use a separate loss at each prediction head. For our binary segmentation branch, we used weighted binary cross-entropy loss, a standard loss for imbalanced binary segmentation tasks. The raw logits produced by the model are first passed through a sigmoid activation for normalization. The loss is then calculated as:

$$L_{BCE} = -\frac{1}{N} \sum_i \Big[w \cdot t_i \cdot log(o_i) + (1-t_i) \cdot log(1-o_i) \Big], \tag{3.7}$$

where t_i is the target value for the pixel i and o_i is the sigmoid output. Since this is an unbalanced segmentation task, a weight w was used to increase penalization for foreground pixels. To further account for the imbalanced dataset, an additional intersection over union loss was used for the

segmentation branch:

$$L_{IoU} = \frac{1}{N}\sum_{i}\left[1 - \frac{t_i \cdot o_i}{t_i + o_i - t_i \cdot o_i}\right]. \tag{3.8}$$

For the affinity field branches of the model, a simple $L1$ regression loss was applied only to the foreground locations of both the vertical and horizontal affinity fields:

$$L_{AF} = \frac{1}{N_{fg}}\sum_{i}\left[|t_i^{haf} - o_i^{haf}| + |t_i^{vaf} - o_i^{vaf}|\right]. \tag{3.9}$$

The total loss applied to the model is a simple summation of the individual losses:

$$L_{total} = L_{BCE} + L_{IoU} + L_{AF}. \tag{3.10}$$

3.4 Experimental Evaluation

3.4.1 Implementation Details

Our backbone architecture (DLA-34) is a fully convolutional network that does not retain the original resolution, but rather downsizes the outputs by a factor of 4; thus, we re-scaled the input images to one-half their original resolution during run-time and reshaped the ground truth affinity fields and segmentation masks to one-eighth the original resolution (accounting for the model's downsizing factor). This has the added benefit of making our decoding process faster since we now process only an eighth of the original rows. The decoding time typically depends on the number of lanes, the quality of the outputs produced by the model, and the output size. On average, it takes about 15-20ms on a modern CPU without any code optimizations. However, since this an entirely CPU-based operation, it should not affect the overall latency of the approach. We also make use of random rotations, crops, scales and horizontal flips during training.

We use the Adam optimizer as our solver with a learning rate of 0.0001, weight decay of 0.001, and train for a total of 40 epochs. We also employ a scheduler that reduces the learning rate

Table 3.1: Attributes of popular lane detection datasets.

Dataset	TuSimple	CULane	LLAMAS
# Frames	6,408	133,325	100,042
Train	3,268	88,880	58,269
Validation	358	9,675	20,844
Test	2,782	34,680	20,929
Resolution	1280×720	1640×590	1280×717
Road Type	highway	urban, rural, highway	highway

by a factor of 5 every 10 epochs. The weight w for the loss in Eq. 3.7 was set to 9.6 because there are approximately 9.6 times as many background pixels than there are foreground (lane) pixels in most public datasets. To avoid overfitting, early stopping was implemented by retaining the model parameters that best performed on the validation set. Using a single GTX Titan X Maxwell GPU, training our model on the CULane dataset until convergence (about 25-30 epochs) takes 2-3 days. Significant speedup can be obtained by using more modern GPUs and by employing multiple GPUs when available.

3.4.2 Datasets

To train and benchmark our proposed approach, we make use of the popular TuSimple, CULane [55], and LLAMAS [73] datasets. TuSimple features good and fair weather conditions in various daytime lighting and traffic conditions, employing highways with up to five lanes. Meanwhile, CULane contains significantly more data and also divides test images into nine categories that contain more complex scenarios, including images with challenging lighting conditions. Finally, the LLAMAS dataset is a newer dataset with a sizeable amount of images all obtained using highway recordings and generated from an automated labeling pipeline. A summary of all datasets is compiled in Table 3.1.

3.4.3 Metrics

We use the same evaluation metrics used in past literature to make a representative comparison between our approach and prior work. This consists of the official metric of the TuSimple dataset (accuracy), the false positive (FP) rate, and the false negative (FN) rate. The TuSimple accuracy is calculated as:

$$Accuracy = \frac{N_{pred}}{N_{gt}} \tag{3.11}$$

where N_{pred} is the number of lane points that have been correctly predicted and N_{gt} is the number of ground-truth lane points.

Additionally, we report the $F1$ measure, which is based on the intersection over union (IoU) and is the only metric for CULane. This is calculated as in [55]:

$$F1 = 2 \cdot \left(\frac{precision \cdot recall}{precision + recall} \right) \tag{3.12}$$

where *precision* is defined as $\frac{TP}{TP+FP}$, *recall* is defined as $\frac{TP}{TP+FN}$, TP is the number of lane points that have been correctly predicted, FP is the number of false positives, and FN is the number of false negatives. This same $F1$ measure is also used for the lane approximations benchmark of the LLAMAS dataset.

3.4.4 Ablation Experiments

In this subsection, we conduct a series of ablation experiments to validate our design choices. All ablation studies were conducted on the TuSimple validation set and can be seen in Table 3.2. The first row contains the results of the standard LaneAF model, which we denote as the baseline model B. First, we train variants without the IoU loss (B w/o IoU) and the weighted binary cross-entropy loss (B w/o wBCE). Removing these losses decreased accuracy

Table 3.2: LaneAF ablation experiments on the TuSimple validation set.

Model type	F1 (%)	Acc (%)	FP	FN
B[1]	95.31	94.62	0.0435	0.0500
B w/o IoU[2]	95.17	94.31	0.0456	0.0507
B w/o wBCE[3]	93.75	94.19	0.0598	0.0649
B w/o RT[4]	93.56	94.29	0.0614	0.0670
B (DS-2)[5]	94.76	94.22	0.0484	0.0559
B (DS-8)	93.94	92.73	0.0549	0.0656
B (HC-128)[6]	94.80	94.56	0.0503	0.0535
DLA-34 multi-class[7]	88.86	92.59	0.1115	0.1114

[1] B: baseline model with down-sampling factor 4 and 256 channels in output head [2] IoU: IoU loss
[3] wBCE: weighted BCE loss [4] RT: random transformations during training [5] DS-x: down-sampling factor for outputs [6] HC-x: number of channels in output head
[7] DLA-34 multi-class: DLA-34 model trained for lane detection via multi-class segmentation

quite drastically while increasing the false positive and false negative rate. In fact, without the weighted binary cross-entropy loss, the F1 score in particular dropped significantly. The same is observed for the baseline model without random transformations during training (B w/o RT), as depicted in the fourth row.

With regards to the down-sampling factor of the outputs, it is clear that the baseline model's factor of 4 achieved the best results; decreasing it to 2 (B (DS-2)) increased runtime and worsened accuracy and F1 slightly, while increasing it to 8 (B (DS-8)) had the most damaging effect on accuracy out of all modifications. We also trained a variant with 128 channels in the output head (B (HC-128)) compared to the original 256, and while this change had the smallest impact with respect to accuracy, it is evident that the baseline's 256 channels yields superior results. Finally, to validate the benefits of our clustering approach over standard multi-class segmentation, we trained a DLA-34 model to directly perform multi-class segmentation of all lanes (DLA-34 multi-class). This model obtained the worst F1 and accuracy scores out of all variants. This result clearly illustrates the effectiveness of binary segmentation followed by a

Table 3.3: LaneAF results on the TuSimple benchmark.

Method	F1 (%)	Acc (%)	FP	FN	MACs (G)
ResNet-18 [69]	87.87	92.69	0.0948	0.0822	-
PolyLaneNet [74]	90.62	93.36	0.0942	0.0933	**1.7**
Cascaded-CNN [60]	90.82	95.24	0.1197	0.0620	-
LaneNet [59]	94.80	96.38	0.0780	0.0244	-
ENet-SAD [58]	95.92	96.64	0.0602	0.0205	-
SCNN [55]	96.53	96.53	0.0617	**0.0180**	-
ResNet-34 [69]	96.77	92.84	0.0918	0.0796	-
PINet [61]	**97.20**	**96.75**	0.0310	0.0250	-
LaneATT(ResNet-18) [66]	96.71	95.57	0.0356	0.0301	9.3
LaneATT (ResNet-34) [66]	96.77	95.63	0.0353	0.0292	18.0
LaneATT (ResNet-122) [66]	96.06	96.10	0.0564	0.0217	70.5
LaneAF (DLA-34)	96.49	95.62	**0.0280**	0.0418	22.2

separate affinity field-based clustering approach.

3.4.5 Results

Performance results from LaneAF on the TuSimple benchmark are shown in Table 3.3. It can be seen that our false positive rate sets a new standard (0.0280) among the current state-of-the-art. This demonstrates that our model does not incorrectly detect a lane pixel as often as other networks and that LaneAF's multi-branch approach leads to confident lane pixel predictions. While we obtain superior accuracy to other backbone architectures such as ResNet-18 and -34 [69], our approach falls slightly short of current state-of-the-art models such as PINet [61], ENet-SAD [58], and SCNN [55]. However, our false negative rate is only marginally higher, signifying that the incorrectly classified lane pixels are most likely at the very ends of the lanes. Additionally, six training runs were conducted on this dataset with different random seeds, producing a standard deviation of 0.12 on the accuracy metric. From the consistency of these results, we can see that our proposed method is robust.

Table 3.4 displays the state-of-the-art results of our model on the CULane benchmark. With this significantly larger and more complex dataset, we can see that LaneAF's performance improves greatly with respect to other models and demonstrates our network's ability to generalize. LaneAF (with DLA-34) outperforms the current state-of-the-art with an F1 score of 77.41%,

Table 3.4: LaneAF results on the CULane benchmark.

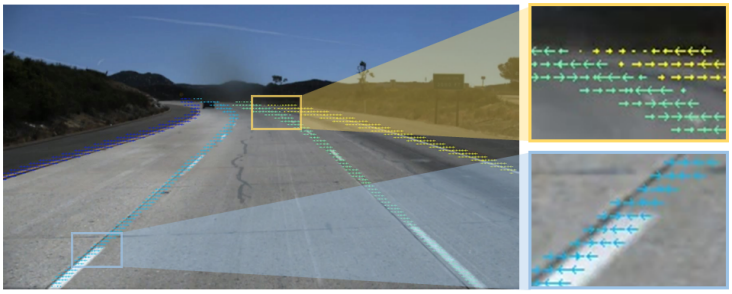
Method	Total	Normal	Crowded	Dazzle	Shadow	No line	Arrow	Curve	Cross	Night	MACs (G)
ResNet-18 [69]	68.40	87.70	66.00	58.40	62.80	40.20	81.00	57.90	1743	62.10	-
ENet-SAD [58]	70.80	90.10	68.80	60.20	65.90	41.60	84.00	65.70	1998	66.00	-
SCNN [55]	71.60	90.60	69.70	58.50	66.90	43.40	84.10	64.40	1990	66.10	-
ResNet-34 [69]	72.30	90.70	70.20	59.50	69.30	44.40	85.70	69.50	2037	66.70	-
ERFNet-Intra-KD [75]	72.40	-	-	-	-	-	-	-	-	-	-
CurveLanes-NAS-M [76]	73.50	90.20	70.50	65.90	69.30	48.80	85.70	67.50	2359	68.20	33.7
SIM-CycleGAN [77]	73.90	91.80	71.80	66.40	76.20	46.10	87.80	67.10	2346	69.40	-
ERFNet-E2E [78]	74.00	91.00	73.10	64.50	74.10	46.60	85.80	71.90	2022	67.90	-
PINet [61]	74.40	90.30	72.30	66.30	68.40	49.80	83.70	65.60	1427	67.70	-
LaneATT (ResNet-18) [66]	75.09	91.11	72.96	65.72	70.91	48.35	85.49	63.37	**1170**	68.95	9.3
RESA-50 [79]	75.30	92.10	73.10	69.20	72.80	47.70	88.30	70.30	1503	69.9	-
LaneATT (ResNet-34) [66]	76.68	**92.14**	75.03	66.47	78.15	49.39	**88.38**	67.72	1330	70.72	18.0
LaneATT (ResNet-122) [66]	77.02	91.74	**76.16**	69.47	76.31	50.46	86.29	64.05	1264	70.81	70.5
LaneAF (ENet)	74.24	90.12	72.19	68.70	76.34	49.13	85.13	64.40	1934	68.67	**2.2**
LaneAF (ERFNet)	75.63	91.10	73.32	69.71	75.81	50.62	86.86	65.02	1844	70.90	22.2
LaneAF (DLA-34)	**77.41**	91.80	75.61	**71.78**	**79.12**	**51.38**	86.88	**72.70**	1360	**73.03**	23.6

Table 3.5: LaneAF results on the LLAMAS benchmark.

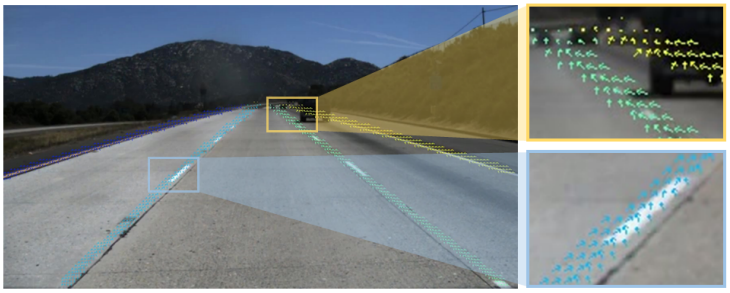
Method	F1 (%)	Prec (%)	Recall (%)
PolyLaneNet [74]	88.40	88.87	87.93
LaneATT(ResNet-18) [66]	93.46	**96.92**	90.24
LaneATT (ResNet-34) [66]	93.74	96.79	90.88
LaneATT (ResNet-122) [66]	93.54	96.82	90.47
LaneAF (DLA-34)	**96.07**	96.91	**95.26**

surpassing models of similar size and even LaneATT [66] with its largest backbone, ResNet-122. Moreover, LaneAF sets a new benchmark in a majority of categories, including difficult ones such as Dazzle, Shadow, No line, Curve, and Night, exhibiting our model's high adaptability to curving roads and challenging lighting conditions.

For the CULane dataset, we additionally trained LaneAF models with ENet [51] and ERFNet [70] backbones, where we forego the final few upsampling/transposed convolution layers to ensure a downsampling factor of 4 (same as the DLA-34 variant). This allows us to make direct comparisons between our approach and other approaches using the same backbone architecture. For instance, LaneAF with the ENet backbone outperforms ENet-SAD [58] by over 3% with respect to the F1 score. When using ERFNet as the backbone network, LaneAF's F1 score eclipses other ERFNet-based models such as ERFNet-E2E [78] and ERFNet-Intra-KD [75] by 1.63% and 3.23%, respectively. These comparisons confirm that the performance gains of



(a) Predicted horizontal affinity field (HAF)



(b) Predicted vertical affinity field (VAF)

Figure 3.3: Example outputs produced by LaneAF with color coded affinity fields; each color represents a unique lane instance based on affinity field decoding. Of note is the successful discrimination of lane instances even as the lanes converge.

LaneAF are achieved through a combination of the DLA-34 backbone and our proposed affinity fields based clustering.

Furthermore, LaneAF again achieves state-of-the-art performance on the LLAMAS dataset with an F1 score of 96.07%. This surpasses LaneATT's [66] best model by over 2%, as shown in Table 3.5. This gap in performance is due to LaneAF's high Recall score, which indicates that the model is more adept at retrieving true lane pixels.

Qualitative examples of the predicted affinity fields from our approach are depicted in Figures 3.3a and 3.3b. The clustered outputs shown here were created using the affinity field decoder, outlined in Algorithm 3.2. In Figure 3.3a, the HAF vectors point towards the center of their respective lane lines for each row of the output image. Lane clusters are still successfully separated despite being closely located for numerous rows, demonstrated in yellow box of Figure 3.3a. Likewise, in Figure 3.3b, the VAF vectors point along the lane towards the mean location of

the next row's lane pixels. This is visualized in the yellow box of Figure 3.3b, where for each unique lane instance, the unit vector points towards the next row's mean lane pixel location. For both Figures 3.3a and 3.3b, the blue boxes clearly display how the HAF and VAF are implemented for a single detected lane instance.

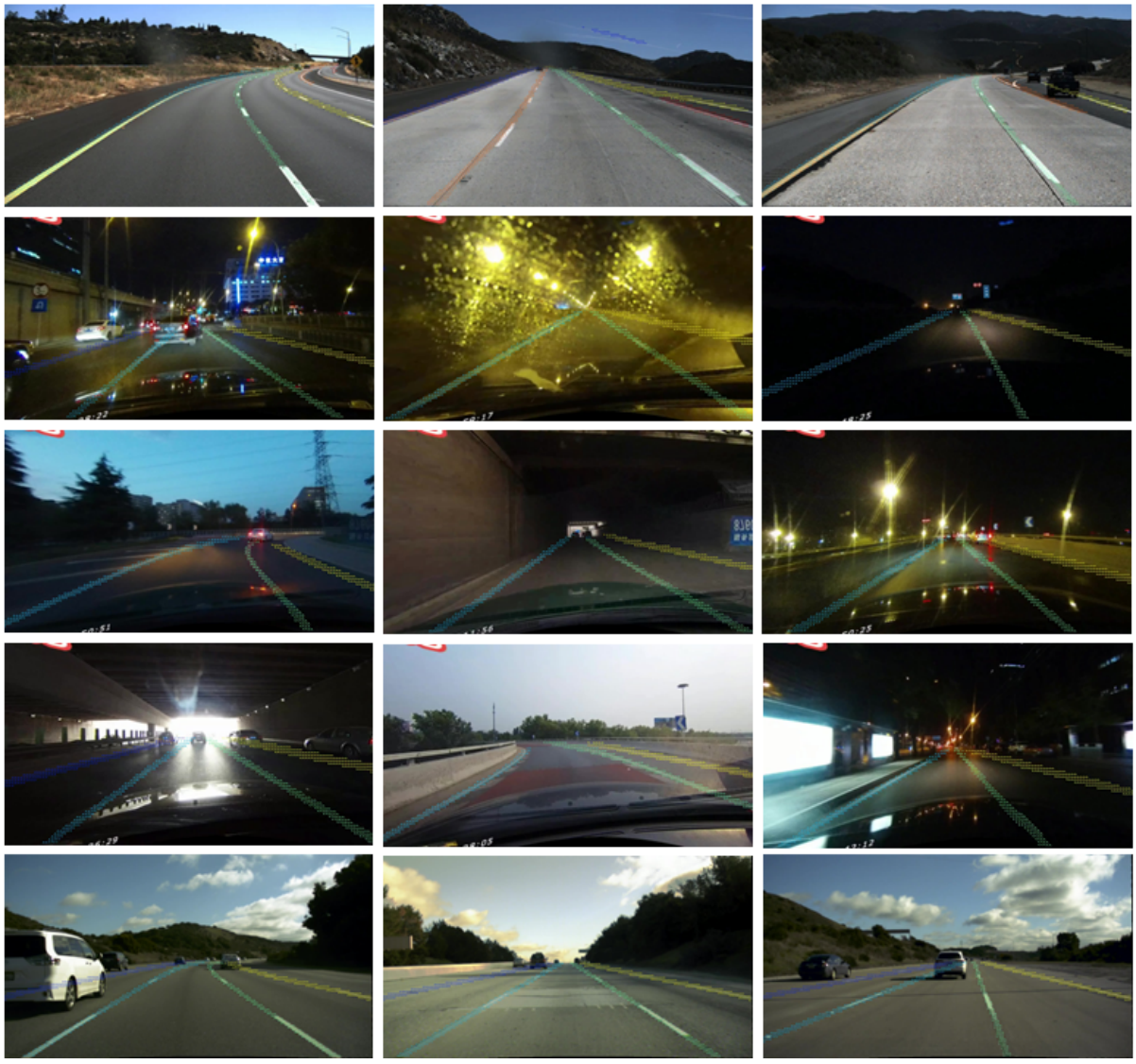


Figure 3.4: LaneAF qualitative results on TuSimple (row 1), CULane (rows 2-4), and LLAMAS (row 5).

Another key point to note is the accuracy of the model at lane points that are farther away from the camera. Since the ground truth segmentation masks for each lane are of approximately

the same thickness in the image plane from top to bottom, the model is trained to predict thick foreground masks for lane points even if they are farther away. This results in little to no degradation for lane points that are far away. However, at the horizon, some clusters will be occasionally assigned to non-optimal lanes due to the close proximity of the lane lines.

In Figure 3.4, we show additional qualitative results from the TuSimple dataset (row 1), the CULane dataset (rows 2-4), and the LLAMAS dataset (row 4). The TuSimple examples demonstrate LaneAF's high performance on curved highways and on lanes that are merging and splitting due to entrances and exits, highlighting our model's flexibility to the number of lanes present on a given road. Also notable in the middle image of the first row is the false detection of a lane line due to an airplane contrail. The displayed results from CULane include challenging scenarios that illustrate LaneAF's robustness on curved roads and in very poor lighting conditions. This diverse set of examples exhibit characteristics of the Dazzle, Shadow, Curve, and Night categories of the CULane dataset. Finally, the LLAMAS samples show excellent performance on additional highway scenes similar to the TuSimple examples.

3.5 Chapter Summary

In this chapter, we proposed a novel approach to lane detection and instance segmentation through the use of binary segmentation masks and per-pixel affinity fields. The horizontal and vertical affinity fields, along with the predicted binary masks were demonstrated to successfully cluster lane pixels into unique lane instances in a post-processing step. This is accomplished using a simple row-by-row decoding process with little overhead, and enables LaneAF to detect a variable number of lanes of arbitrary width without assuming a fixed or maximum number of lanes. This form of clustering is also more interpretable in comparison to previous visual approaches since it can be analyzed to easily identify and correct sources of error. The ablation study conducted also validated the effectiveness of the approach over standard multi-class segmentation.

Our proposed method achieves the lowest reported false positive rate (0.0280) on the TuSimple benchmark; on the larger and more comprehensive CULane dataset, LaneAF sets a new state-of-the-art result with a total F1 score of 77.41%, surpassing much deeper and more complex models. LaneAF also achieves a state-of-the-art F1 score on the LLAMAS benchmark by a significant margin (+2%), underscoring its robust performance.

3.6 Acknowledgements

Chapter 3, in full, is a reprint of the material as it appears in the IEEE Robotics and Automation Letters (2021), by Akshay Rangesh, Hala Abualsaud, Sean Liu, David Lu, Kenny Situ, and Mohan M. Trivedi. The dissertation author was one of the primary investigators and authors of this paper.

Chapter 4

TrackMPNN: A Message Passing Graph Neural Architecture for Multi-Object Tracking

This study follows many classical approaches to multi-object tracking (MOT) that model the problem using dynamic graphical data structures, and adapts this formulation to make it amenable to modern neural networks. Our main contributions in this work are the creation of a framework based on dynamic undirected graphs that represent the data association problem over multiple timesteps, and a message passing graph neural network (MPNN) that operates on these graphs to produce the desired likelihood for every association therein. We also provide solutions and propositions for the computational problems that need to be addressed to create a memory-efficient, real-time, online algorithm that can reason over multiple timesteps, correct previous mistakes, update beliefs, and handle missed/false detections. To demonstrate the efficacy of our approach, we only use the 2D box location and object category ID to construct the descriptor for each object instance. Despite this, our model performs on par with state-of-the-art approaches that make use of additional sensors, as well as multiple hand-crafted and/or learned features. This illustrates that given the right problem formulation and model design, raw bounding boxes (and

Code: https://github.com/arangesh/TrackMPNN

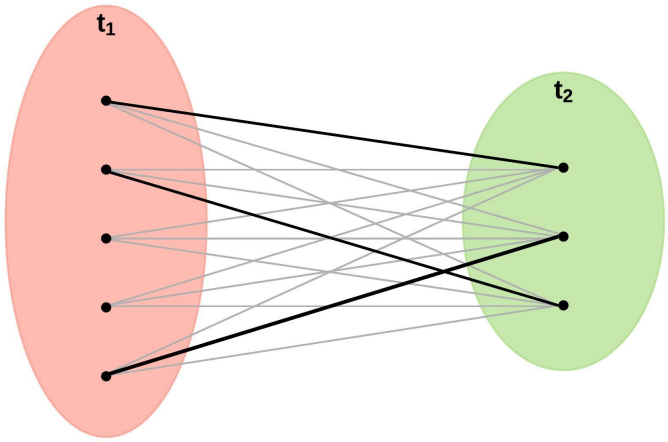


Figure 4.1: Most tracking-by-detection approaches are reduced to successive bipartite matching problems between two disjoint sets, namely - sets of detections from two (consecutive) timesteps as shown in the illustration above. The goal of the multi-object tracker is to propose correct associations between the two sets (bold lines), and suppress spurious ones (gray lines).

their kinematics) from any off-the-shelf detector are sufficient to achieve competitive tracking results on challenging MOT benchmarks.

4.1 Introduction

Object tracking is a crucial part of many complex autonomous systems and finds application in a variety of domains that require tracking of humans and/or objects. Tracking introduces memory and persistence into a system via association, as opposed to standalone measurements and detections in time. This enables systems to monitor object or agent behavior over a period of time, thus capturing historical context which can then be used to better assess current state or more accurately predict future states. Examples of such usage from different domains include - gauging human/pedestrian intent and attention based on history of movement [80] [81] [82], surveillance of crowds scenes to perceive crowd behavior [83], predicting future trajectories of vehicles based on past tracks [84] [7], tracking compact entities like cells and molecules to better understand biological processes [85] etc.

Tracking methods are numerous and varied in their approach, and can broadly be categorized based on the number of objects being tracked i.e. single or multi-object tracking (MOT) approaches. MOT approaches go beyond feature and appearance tracking and attempt to solve successive data association problems to resolve conflicts between multiple competing observations and tracks. Thus, the problem of MOT is also a problem of data association - where multiple observations are to be assigned to multiple active object tracks to optimize some global criterion of choice. MOT approaches can further be classified based on their mode of operation i.e. online, offline (batch), or near-online. Online approaches provide tracks in real-time and continuously, and thus find use in real-time systems like autonomous cars. On the other hand, offline approaches benefit from viewing an entire sequence (i.e. conditioning on all available data) before estimating object tracks. This makes them relatively more robust than their online counterparts. However, these approaches are only suited to applications where post hoc operation is not a hindrance, e.g. surveillance.

Modern MOT techniques typically work within the tracking-by-detection paradigm, where trackers function by stitching together individual detections across time (see Figure 4.1). These detections can either be obtained from a separate upstream object detector that operates independently, or by integrating the detector and tracker into one cohesive unit. Irrespective of the particularities of the approach, multi-object trackers aim to robustly track multiple objects through appearance changes, missed/false detections, temporary occlusions, birth of new tracks, death of existing tracks, and other such inhibitors. They do this by engineering better feature descriptors for each object, improving the similarity or cost function used during data association, or by better optimizing the underlying objective.

In this study, we propose a novel MOT framework based on dynamic, undirected, bipartite graphs for contemporary neural networks. Our approach is partly motivated by the probabilistic graph representation of classical multi-target tracking methods like Multi-Hypothesis Tracking (MHT) [86], with a focus on compatibility with modern graph neural networks (GNNs). Our

approach is online, capable of tracking multiple objects, reasoning over multiple timesteps, and holding multiple hypotheses at any given time. In addition to this, our proposed approach can run real-time on almost any modern GPU without exceeding memory and compute limits.

4.2 Related Research

Modern MOT techniques have different approaches to address the underlying problem of data association. While most methods utilize some form of tracking-by-detection, some create a cohesive framework where the detector and tracker work in tandem [87] [88] [89]. This usually leads to a symbiotic relationship, where detector and tracker benefit one another. In similar fashion, some studies have explored the benefits of incorporating other related tasks in their MOT framework, including segmentation [90] [91] and reconstruction [92]. Some others tweak classical approaches by making them more tractable [93], or by developing better feature representations [94] - thereby making them amenable for deployment in dense, cluttered scenes. Other works focus solely on feature engineering and tuning of cost functions [95, 96], and assume detections from a trained off-the-shelf detector. Finally, with the introduction of large scale datasets for autonomous driving, many modern approaches tend to make use of 3D sensors like LiDARs, either by itself [97], or via multi-modal fusion with other imaging sensors like cameras [98]. This helps these methods resolve temporary occlusion, truncation, and other such issues arising from 3D ambiguity.

In this study, we introduce a MOT framework suitable for Graph Neural Networks (GNNs). GNNs have found a wide range of applications in recent years; see [99] and references therein. GNNs can be viewed as a generalization of convolutional neural networks(CNNs) [100]. Multi-layer CNNs can extract multi-scale features from spatio-temporal data and generate representations which can solve a variety of machine learning tasks. However, CNNs are defined by kernels that operate only on regular multi-dimensional grids. GNNs generalize CNNs kernel

operation to general graphs (of which CNNs on regular grids are special cases). More importantly for us, one can interpret the kernel operations in this graphical context as generating an output at each node using the values of neighboring nodes and the value of the node itself as inputs. This gives rise to a Message Passing Neural Network (MPNN) architecture [101]. As we shall see in the following sections, our architecture is essentially an MPNN with communicating memory units whose graphical structure evolves in time. This kind of architecture, a GNN with communicating memory units, has been applied to the modeling of spatially interacting mobile agents in [102]. GNNs in general have recently found use in multi-object tracking [103–105]; these models however, do not have the ability to pass messages to other parts in graph, and simply use the graph structure to infer similarity scores between consecutive sets of detections. We know of only one other approach which applies GNNs with message passing to the multi-object tracking problem [106]. Unlike [106] where the model is run on batches of detections in an offline manner (with score averaging), the proposed approach involves dynamic graphs which are processed in a rolling-window scheme - making it suitable for online, real-time applications. Moreover, our message passing operations are specifically crafted with rapidly evolving graph structures in mind.

The proposed approach is also partly motivated by the probabilistic graph representation of classical multi-target tracking methods, for example hypothesis based and track-tree based MHT (Multi-Hypothesis Tracking) [86] [107]. In the these methods, the association of observations to targets/tracks is resolved by generating branches emanating from existing global hypotheses to account for new observations, then propagating the posterior probability of resulting global hypotheses, recursively, along those branches. As in the earlier hypothesis based classical approaches, we do not generate explicit track-trees representing a set of underlying targets and we limit the size of the active section of the graph determining the associations to prevent exponential growth of compute. We also make no explicit assumptions about the underlying Bayesian distributions. The network learns from the ground truth tracking data, how best to

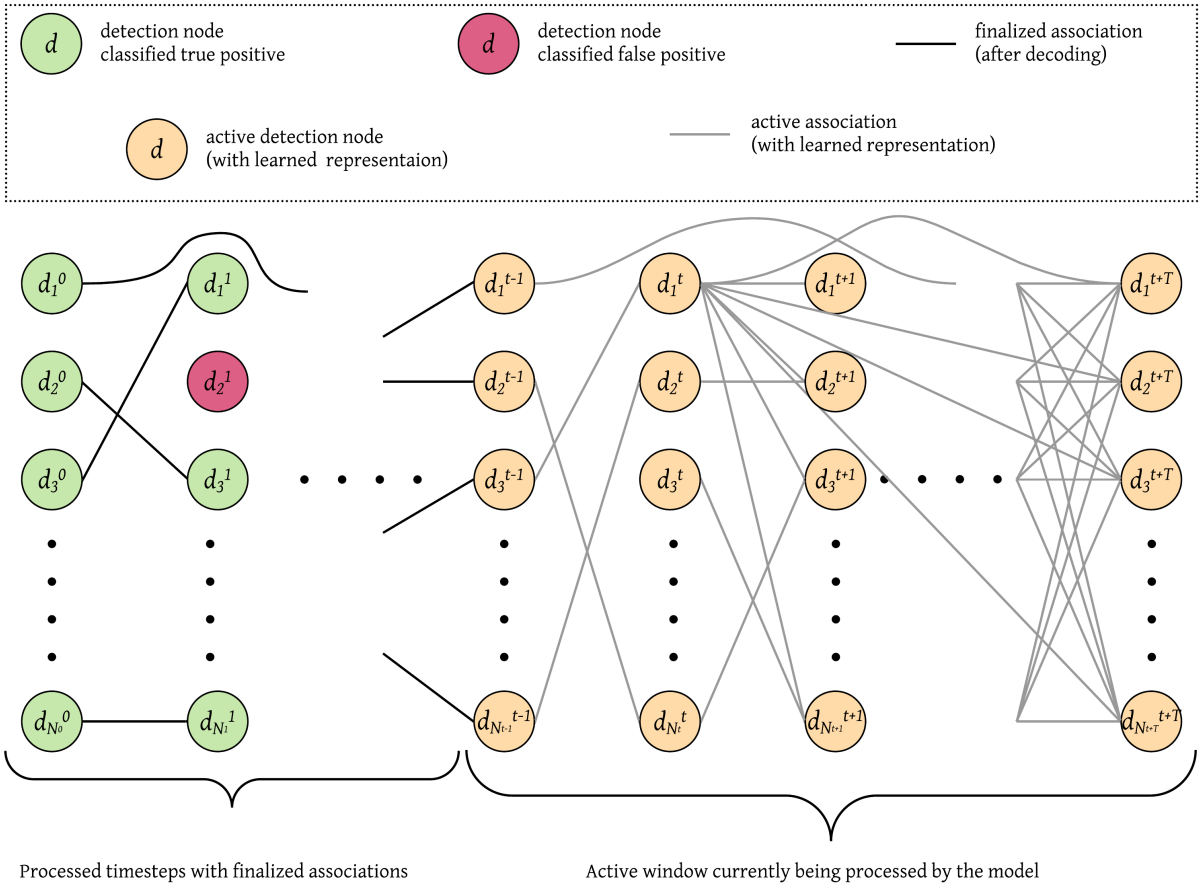


Figure 4.2: Our proposed MOT framework is based on dynamic undirected graphs that depict the data association problem over multiple timesteps. In the Figure above, d_i^j represents a detection node indexed by i, at timestep j.

generate the tracks during inference. Also, we do not enumerate the set of consistent tracks. The graph represents the superposition of all such tracks which can be decoded to produce the optimal one.

4.3 Proposed Tracking Framework

This approach works based on the following idea: as data association in multi-object tracking (MOT) is conventionally modelled as a bipartite graph over two consecutive timesteps, could we simply learn and infer directly on such graph structures using GNNs? This graph

structure could also be expanded to cover multiple consecutive timesteps so as to incorporate non-local information and infer distant associations.

Our approach formulates the MOT problem as inference on an undirected graph where each detection is represented as a node in the graph, and potential associations between different detections are represented by edges connecting them. The graph is dynamically updated at every new timestep, where new detections (and associations) from the latest timestep are added, and old, inactive detections (and associations) are removed. An overview of this setup is illustrated in Figure 4.2. The proposed model (described in Section 4.4) works by operating on this dynamic graph in a rolling window basis (from left to right as presented in Figure 4.2). The size of the rolling window is a design choice to be made based on the desired performance, available memory and compute requirements. This concept of a rolling window is similar to the one used in [108].

Although not explicitly drawn in Figure 4.2 in order to reduce clutter, each edge joining two detection nodes is also treated as a node of the graph in actuality. This helps us endow a vector representation to each potential association between two detections. Thus, this dynamic graph structure is bipartite, with *detection nodes* and *association nodes* forming the two disjoint and independent sets. Detection nodes represent object detections in a sequence, and are initialized with their corresponding feature descriptors. Association nodes represent potential pairwise associations between two detections from different timesteps, and are initialized with zero vectors. These initializations are then transformed and updated by the model to create hidden representations of every node in the graph with each additional timestep.

4.3.1 Training & Inference

Before we describe the model architecture and optimization in detail (Section 4.4), we provide a macro view of the training and inference procedure that we adopt. We make use of the following high level functions pertaining to the graph structure and the model during both training and inference:

Algorithm 4.1 Pseudocode for one training iteration

Input: *feats, labels*
$net \leftarrow TrackMPNN()$ ▷ initialize TrackMPNN model
$G \leftarrow initialize\_graph(feats)$ ▷ initialize graph with detection features and pairwise associations from first two timesteps
$total\_loss \leftarrow 0$ ▷ initialize loss to 0
$(probabilities, loss) \leftarrow net.forward(G, labels)$ ▷ forward pass
$total\_loss \leftarrow total\_loss + loss$ ▷ add to total loss
for $t \leftarrow 2$ **to** t_{end} **do**
  $G \leftarrow update\_graph(G, feats, t)$ ▷ add new nodes and edges to graph; remove earliest nodes and edges
  $(probabilities, loss) \leftarrow net.forward(G, labels)$ ▷ forward pass
  $total\_loss \leftarrow total\_loss + loss$ ▷ add to total loss
  if condition **then**
    $G \leftarrow prune\_graph(G, probabilities)$ ▷ remove low probability nodes to limit memory footprint
  end if
end for
$net \leftarrow net.backward(total\_loss)$ ▷ backward pass to train model
return *net*

- *initialize_graph()*: This creates an initial bipartite graph with detection nodes from two consecutive timesteps, with an association node between every detection pair.

- *update_graph()*: This function is called at the start of every new timestep, to add new (detection and association) nodes and edges to end of the currently active graph. Note that association nodes are only added between the newly introduced detection nodes and unpaired detection nodes from previous timesteps. This function also removes the oldest set of nodes and edges from the currently active part of the graph. It essentially moves the rolling window one step forward.

- *prune_graph()*: This function removes low probability edges and nodes from the currently active part of the graph using a user specified confidence threshold. This function can be called whenever memory/compute requirements exceed what is permissible, or to prevent an explosion of nodes.

- *decode_graph()*: This function is called to decode the model-produced output probabilities

Algorithm 4.2 Pseudocode for inference on a MOT sequence

Input: *feats*
$tracks \leftarrow \{\}$ ▷ initialize tracks to empty
$net \leftarrow TrackMPNN(trained\_weights)$ ▷ initialize TrackMPNN model with trained weights
$G \leftarrow initialize\_graph(feats)$ ▷ initialize graph with detection features and pairwise associations from first two timesteps
$probabilities \leftarrow net(G)$ ▷ forward pass to get probabilities
$tracks \leftarrow tracks \bigcup decode\_graph(G, probabilities)$ ▷ decode model probabilities to produce tracks for desired window
for $t \leftarrow 2$ **to** t_{end} **do**
    $G \leftarrow update\_graph(G, feats, t)$ ▷ add new nodes and edges to graph; remove earliest nodes and edges
    $probabilities \leftarrow net(G)$ ▷ forward pass to get probabilities
    $tracks \leftarrow tracks \bigcup decode\_graph(G, probabilities)$ ▷ decode model probabilities to produce tracks for desired window
    if condition **then**
        $G \leftarrow prune\_graph(G, probabilities)$ ▷ remove low probability nodes to limit memory footprint
    end if
end for
return *tracks*

for every node in the graph into corresponding object tracks. This can either be done in a greedy manner (by following the highest probability path from left to right) or by using the Hungarian algorithm (on consecutive timesteps from left to right).

- *TrackMPNN()*: This initializes an instance of the proposed model (described in Section 4.4).
- *TrackMPNN.forward()*: This carries out one forward pass of the data through the model.
- *TrackMPNN.backward()*: This carries out one backward pass through the model to produce gradients with respect to the losses.

The training and inference pseudocode that make use of these functions are presented in Algorithms 4.1 and 4.2. As described earlier, the inference procedure operates in a rolling window manner, from past (left) to future (right). Thus, we operate on the entire tracking sequence in a continuous manner during inference. However, this is not possible during training, where gradients need to be stored for all detection and association nodes encountered in the dynamic

graph. To make the training process computationally tractable, we split each tracking sequence into smaller contiguous chunks, and process each such mini-sequence as an individual sample. As depicted in Algorithm 4.1, we also accumulate losses over the sequence length and backpropagate only after the entire sequence has been processed.

4.4 Model Architecture

In this study, we use a class of Graph Neural Networks called Message Passing Neural Networks (MPNNs) [101]. Like most MPNNs, our proposed TrackMPNN model consists of:

- a message function:

$$\vec{m}_v^k = \sum_{w \in \mathcal{N}(v)} M(\vec{h}_v^{k-1}, \vec{h}_w^{k-1}), \tag{4.1}$$

- a node/vertex update function:

$$\vec{h}_v^k = U(\vec{h}_v^{k-1}, \vec{m}_v^k), \tag{4.2}$$

- a readout/output function:

$$\hat{y}_v^k = R(\vec{h}_v^k), \tag{4.3}$$

where vertices v, w are nodes in graph G, and in our context be either detection nodes or association nodes. Note that iteration k is different from the current timestep t of the tracking sequence. Whereas t signifies the lifetime of the entire graph, k denotes the lifetime of each individual node in this dynamic graph. For our purposes, we use separate functions/weights for detection nodes and association nodes, which we detail in the next two subsections.

4.4.1 Detection node operations

Consider the illustration of a detection node d and its neighboring association nodes $a_i \in \mathcal{N}(d)$ presented in Figure 4.3. The operations associated with nodes of this type are presented below.

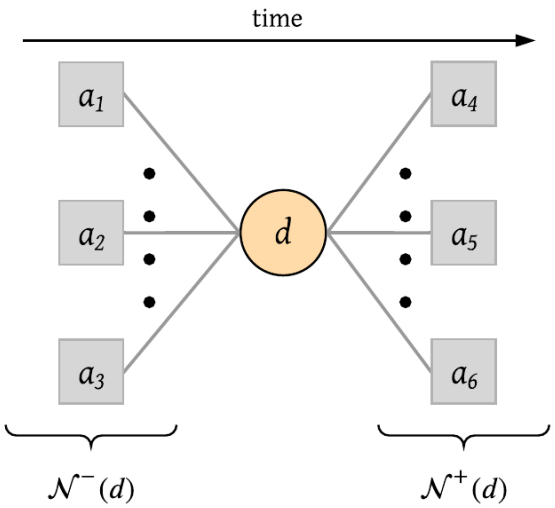


Figure 4.3: Illustration of a detection node d and its neighboring association nodes $\{a_i | a_i \in \mathcal{N}(d)\}$. The neighborhood can be split into two disjoint sets $\mathcal{N}^-(d)$ and $\mathcal{N}^+(d)$, denoting all nodes from the past and future respectively.

Initialization

The hidden state of a detection node is initialized with a pair of linear transformations (with non-linearity and normalization) of its input feature representation $\vec{x}_d$:

$$\vec{h}_d^0 = \mathbf{W}_{det}^{i'} \, \text{BatchNorm}(\vec{x'}_d) + \vec{b}_{det}^{i'}, \tag{4.4}$$

where

$$\vec{x'}_d = \text{ReLU}(\mathbf{W}_{det}^{i} \vec{x}_d + \vec{b}_{det}^{i}). \tag{4.5}$$

The input representation $\vec{x}_d$ of a detection node d can be composed of different features

and attributes describing the detected object instance (a detection produced by the object detector). In this study, we use detections produced by a recurrent rolling convolutions (RRC) detector [109] - though any other detector could be used instead. Contrary to many contemporary approaches, we only use the 2D box locations and object categories produced by the detector to describe each object instance, and rely on the model to learn the patterns of sequential locations on the image plane.

The input representation of each detection $\vec{x}_d$ is then defined as follows:

$$\vec{x}_d = [\vec{x}_{d;2d} || \vec{x}_{d;cat}], \tag{4.6}$$

where $||$ represents concatenation along the singleton dimension.

$$\vec{x}_{d;2d} = (x_1, y_1, x_2 - x_1, y_2 - y_1, score)^\intercal \tag{4.7}$$

denotes 2D features comprised of the 2D bounding box location (top-left location, width, height) and score, and,

$$\vec{x}_{d;cat} = (0, \cdots, 1, \cdots, 0)^\intercal \tag{4.8}$$

is the one-hot encoding of the object category.

Message and node/vertex update functions

The hidden state of the detection node is updated based on its previous hidden state and the previous hidden states of its neighboring association nodes, weighted by an attention mechanism that takes into account the detection nodes connected by the association:

$$\begin{aligned} \vec{h}_d^k &= \text{GRU}\big(\vec{h}_d^{k-1}, \vec{m}_d^k\big) \\ &= \text{GRU}\left(\vec{h}_d^{k-1}, \sum_{a_i \in \mathcal{N}(d)} \alpha_{da_i}^{k-1} \vec{h}_{a_i}^{k-1}\right), \end{aligned} \tag{4.9}$$

where $\vec{m}_d^k$ is the message passed to detection node d at iteration k, and $\alpha_{da_i}^{k-1}$ are the attention weights [110] assigned to each association node a_i^{k-1}.

Fully expanded out, the coefficients computed by the attention mechanism may then be expressed as:

$$\alpha_{da_i}^{k-1} = \frac{exp(LReLU(\mathbf{a}^\intercal | \mathbf{W}_{\mathbf{det}}^{\mathbf{h}} \vec{h}_d^{k-1} - \mathbf{W}_{\mathbf{det}}^{\mathbf{h}} \vec{h}_{d_i}^{k-1}|))}{\sum_{d_j \in \mathcal{N}^2(d)} exp(LReLU(\mathbf{a}^\intercal | \mathbf{W}_{\mathbf{det}}^{\mathbf{h}} \vec{h}_d^{k-1} - \mathbf{W}_{\mathbf{det}}^{\mathbf{h}} \vec{h}_{d_j}^{k-1}|))}, \tag{4.10}$$

where $d_i \in \mathcal{N}(a_i)$ and $d_i \neq d$, $\mathcal{N}^2(\cdot)$ represents the second-order neighborhood of a node, and LReLU denotes the LeakyReLU non-linearity (with negative input slope 0.2).

Readout function

The detection node output o_d^k represents a scalar confidence value obtained by simple linear transformation of its hidden state:

$$o_d^k = \mathbf{W}_{det}^o \vec{h}_d^k + \vec{b}_{det}^o. \tag{4.11}$$

4.4.2 Association node operations

Consider the illustration of an association node a and its neighboring detection nodes depicted in Figure 4.4. The operations associated with nodes of this type are presented below.

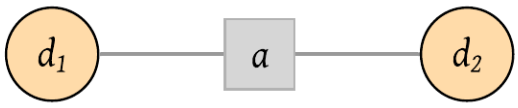


Figure 4.4: Illustration of an association node a and its two neighboring detection nodes $\{d_1, d_2\}$.

Initialization

The hidden state of an association node is initialized with a vector of of 0s.

$$\vec{h}_a^0 = \vec{0} \tag{4.12}$$

Message and node/vertex update functions

The hidden state of the association node is updated based on its previous hidden state and the previous hidden states of its two neighboring detection nodes. Given the fixed degree of an association node, we experiment with two different message update functions. The first is based on the difference between the hidden states of its neighbors:

$$\begin{aligned} \vec{h}_a^k &= \text{GRU}\big(\vec{h}_a^{k-1}, \vec{m}_a^k\big) \\ &= \text{GRU}\Big(\vec{h}_a^{k-1}, \mathbf{W}_{ass}^h[\vec{h}_{d_2}^{k-1} - \vec{h}_{d_1}^{k-1}] + \vec{b}_{ass}^h\Big), \end{aligned} \tag{4.13}$$

and the second is based on their concatenation:

$$\begin{aligned} \vec{h}_a^k &= \text{GRU}\big(\vec{h}_a^{k-1}, \vec{m}_a^k\big) \\ &= \text{GRU}\Big(\vec{h}_a^{k-1}, \mathbf{W}_{ass}^h[\vec{h}_{d_1}^{k-1} || \vec{h}_{d_2}^{k-1}] + \vec{b}_{ass}^h\Big). \end{aligned} \tag{4.14}$$

In both cases, $\vec{m}_a^k$ is the message passed to association node a at iteration k.

Readout function

The association node output o_a^k represents a scalar confidence value obtained by simple linear transformation of its hidden state:

$$o_a^k = \mathbf{W}_{ass}^o \vec{h}_a^k + \vec{b}_{ass}^o. \tag{4.15}$$

4.4.3 Training Losses

Let $G = (V, E)$ denote the dynamic graph at any given instant. We can further split the set of vertices into two disjoint sets, i.e. $V = DN \cup AN$ and $DN \cap AN = \emptyset$, where DN and AN represent the set of detection nodes and association nodes respectively.

We first apply a binary cross-entropy loss at each detection node:

$$\mathcal{L}_{bce;DN} = -\frac{1}{|DN|} \sum_{d \in DN} \Big(y_d \log(\text{Sigmoid}(o_d^{k_d})) + (1 - y_d) \log(1 - \text{Sigmoid}(o_d^{k_d})) \Big), \quad (4.16)$$

where k_d denotes the latest iteration of detection node d, and

$$y_d = \begin{cases} 1, & \text{if true positive} \\ 0, & \text{otherwise.} \end{cases} \quad (4.17)$$

Similarly, we apply a binary cross-entropy loss at each association node:

$$\mathcal{L}_{bce;AN} = -\frac{1}{|AN|} \sum_{a \in AN} \Big(y_a \log(\text{Sigmoid}(o_a^{k_a})) + (1 - y_a) \log(1 - \text{Sigmoid}(o_a^{k_a})) \Big), \quad (4.18)$$

where k_a denotes the latest iteration of association node a, and

$$y_a = \begin{cases} 1, & \text{if } \mathcal{N}(a) \text{ belongs to the same track} \\ 0, & \text{otherwise.} \end{cases} \quad (4.19)$$

We also apply a cross-entropy loss for every set of competing association nodes:

$$\mathcal{L}_{ce;AN} = -\frac{1}{|DN|}\sum_{d\in DN}\left(\frac{1}{|\mathcal{N}^{-}(d)|}\sum_{a\in\mathcal{N}^{-}(d)} y_a \log(\text{Softmax}(o_a^{k_a})) + \frac{1}{|\mathcal{N}^{+}(d)|}\sum_{a\in\mathcal{N}^{+}(d)} y_a \log(\text{Softmax}(o_a^{k_a}))\right), \quad (4.20)$$

where

$$y_a = \begin{cases} 1, & \text{if } \mathcal{N}(a) \text{ belongs to the same track} \\ 0, & \text{otherwise,} \end{cases} \quad (4.21)$$

$\mathcal{N}^{-}(\cdot)$ is the set of all neighbors from past timesteps, and $\mathcal{N}^{+}(\cdot)$ is the set of all neighbors from future timesteps.

The total loss used to train the entire model is the sum of the individual losses:

$$\mathcal{L} = \mathcal{L}_{bce;DN} + \mathcal{L}_{bce;AN} + \mathcal{L}_{ce;AN} \quad (4.22)$$

4.5 Experiments and Analyses

4.5.1 Implementation Details

Dynamic Undirected Graph

As detailed in Section 4.3, our approach operates using a temporal rolling window, where detection and association nodes falling inside the window make up the dynamic graph at any given instant. The two hyperparameters that govern this process are the current window size (CWS) and the retained window size (RWS). CWS defines the size of the rolling window in discrete timesteps/frames. RWS determines the number of discrete timesteps for which an unassociated detection node is kept in the dynamic graph after it is no longer in the rolling window. This is to ensure that objects that are temporarily occluded or missing have a chance to be re-identified

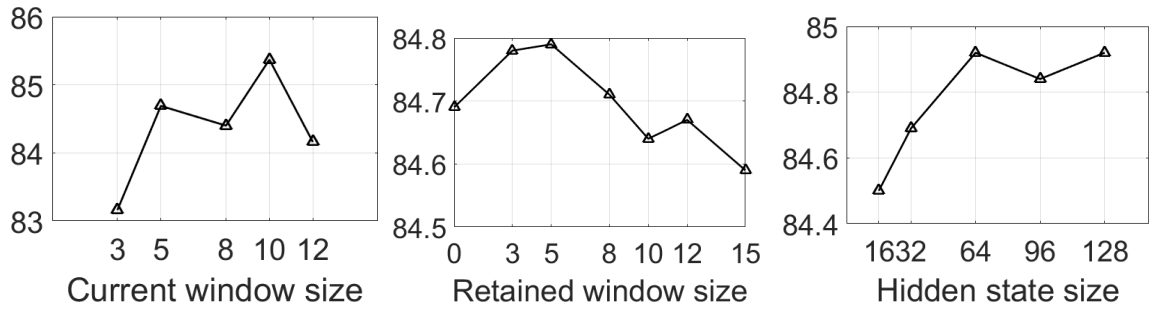
again. The rolling window is updated after each timestep by adding new nodes from the next timestep and removing nodes from the earliest timestep. Before detection nodes are removed however, they are assigned to tracks as part of the decoding process i.e., for every detection node that is removed, it is assigned to the same track as an earlier detection node corresponding to the maximum association probability as produced by the model (Eq. 4.15). If a true positive detection node does not have any high probability associations, a new track is initialized.

Training & Optimization

As depicted in Eq. 4.22, the total loss is simply the sum of the individual losses - without any weighting. We also forego mini-batch training and opt for *mini-sequence* training in its place. Instead of using a mini-batch of examples for each training sample, we use one mini-sequence for each training sample. A mini-sequence is simply a tracking sequence of CWS contiguous timesteps, randomly sampled from full-length tracking sequences. Similar to mini-batch gradient descent, the losses are accumulated, and then backpropagated after the entire mini-sequence is processed. In our experience, this tends to stabilize training, produce better gradients, and require less memory. The entire network is trained using an Adam optimizer with learning rate 0.0001, $\beta_1 = 0.9$ and $\beta_2 = 0.999$ for 50 epochs. Additionally, we augment the training split with multiple random transformations. This includes time reversal (reverse timestep/frame ordering), dropout of detections (ignore a fraction of true positive detections), and horizontal flips. Finally, we initialize the bias values in the detection and association node readout functions (Eq. 4.11, 4.15) to $+4.595$ and -4.595 respectively. This is to keep the losses manageable during the initial phase of training.

Inference

During inference, care is taken to ensure that hyperparameters associated with the dynamic graph remain unchanged from training. Instead of feeding the model mini-sequences, we instead supply the full-length sequence as a single sample. The model operates on this sequence in a rolling window manner, generating continuous tracks along the way.



(a) MOTA versus Current Window Size (RWS=0)

(b) MOTA versus Retained Window Size (CWS=5)

(c) MOTA versus Hidden State Size (CWS=5, RWS=0)

Figure 4.5: Experiments illustrating the effects of different model settings on the tracking performance using the KITTI validation split (for Cars only).

4.5.2 Ablation Experiments

To perform ablation experiments, we make use of the KITTI MOT dataset [27]. The KITTI MOT dataset is comprised of 21 training sequences and 29 testing sequences, with two categories of interest: Cars and Pedestrians. For the purpose of ablation, we use the first 11 training sequences to train the models, and leave the rest for validation. We also restrict our experiments to one object category - Cars. For evaluation, we make use of the CLEAR MOT metrics [111, 112]. In cases where only one metric is desired, we use the Multi-Object Tracking Accuracy (MOTA). The following paragraphs detail our ablation of different settings pertaining to the dynamic graph and the model.

Dynamic Graph Settings

To assess the benefits of a potentially larger rolling window size, we trained multiple models with different settings of CWS. In all cases, RWS was set to 0. Figure 4.5a depicts the plot of MOTA for different settings of CWS. Although all variants achieve very similar results, CWS=10 produces the best MOTA. CWS=5 achieves very similar results at a lower computational and memory cost. Next, we fix CWS=5 and evaluate models with different values of RWS. The results from this experiment are plotted in Figure 4.5b. Once again, we notice that the MOTA values are quite similar, with RWS=5 achieving the best results. Both these experiments also

highlight the robustness of the model across various graph settings.

Model Settings

To compare different model settings and their impacts on performance, we first define a baseline model (B) against which other variants can be compared. The baseline model is trained using graph settings CWS=5, RWS=0, and uses difference based updates for the association nodes (Eq. 4.13). Results from these comparisons are presented in Table 4.1. First, we measure the efficacy of the greedy matching procedure by comparing it with a model using the Hungarian algorithm for optimal bipartite matching (B w/ H). In this variant, the Hungarian algorithm is used to decode the graph into tracks, by optimally matching detections to existing tracks based on association probabilities. We notice that the results are better across the board. Thus, the Hungarian algorithm helps produce slightly longer and continuous tracks - at the expense of additional computational cost. Second, we train a model (B w/o TP) without true-positive classification at the detection nodes (see Eq. 4.16). In this case, every detection is assumed to be a true positive, and the model is used to only produce association probabilities. From the results presented in Table 4.5, we see that this setting also improves performance on all metrics. This makes sense because 2D box locations alone are usually not enough to distinguish true positives from false positives. Third, to compare the two different update functions for association nodes, we also train a variant of the baseline with concatenation based updates (Eq. 4.14). When compared to the difference based updates in B, concatenation produces worse results with a much lower MOTA and a higher IDS. This indicates that difference based updates can better model the similarity/dissimilarity between two detection nodes. Next, to quantify the effects of data augmentation during training, we train a model with random transformations (B w/ RT). This too improves most metrics across the board - indicating the clear benefits of our augmentation scheme - especially when trained on datasets of limited size. Finally, to gauge the effects of hidden state size on the tracking performance, we train multiple models containing GRUs with different hidden state sizes. The resulting MOTAs for different variants are plotted in Figure 4.5c. The plot

Table 4.1: Results from experiments on the KITTI validation set (for Cars only). Arrows indicate if a higher (↑) or lower value of the metric (↓) is desired.

Model type	MOTA (↑)	MOTP (↑)	MT (↑)	ML (↓)	IDS (↓)	FRAG (↓)
B[1]	84.69	0.1009	71.49%	2.26%	320	156
B w/ H[2]	85.99	0.1012	73.20%	2.26%	140	152
B w/o TP[3]	85.18	0.1013	72.85%	2.26%	239	153
B||[4]	79.30	0.1108	65.71%	5.61%	703	205
B w/ RT[5]	84.86	0.1008	72.85%	2.26%	291	153

[1] B: baseline model with difference based updates for association nodes [2] H: Hungarian algorithm [3] TP: true positive classification [4] B||: baseline model with concatenation based updates for association nodes (Eq. 4.14) [5] RT: random transformations during training

Table 4.2: Results on the KITTI multi-object tracking benchmark for Cars. Arrows indicate if a higher (↑) or lower value of the metric (↓) is desired. Our results are in bold.

Method	Sensors	Online	MOTA (↑)	MOTP (↑)	MT (↑)	ML (↓)	IDS (↓)	FRAG (↓)
CenterTrack [89]	RGB camera	✓[1]	0.8883	0.8497	82.15 %	2.46 %	254	227
TrackMPNN + CenterTrack	**RGB camera**	✓	**0.8733**	**0.8449**	**84.46 %**	**2.15 %**	**481**	**237**
JRMOT [113]	RGB camera, LiDAR	✓	0.8510	0.8528	70.92 %	4.62 %	271	273
TrackMPNN + RRC	**RGB camera**	✓	**0.8455**	**0.8507**	**70.92 %**	**4.00 %**	**466**	**482**
mono3DT [88]	RGB camera, GPS	✓	0.8428	0.8545	73.08 %	2.92 %	379	573
MOTSFusion [92]	RGB camera, stereo	✓	0.8424	0.8503	72.77 %	2.92 %	415	569
SMAT [114]	RGB camera	✓	0.8364	0.8589	62.77 %	6.00 %	198	294
mmMOT [115]	RGB camera	×[2]	0.8323	0.8503	72.92 %	2.92 %	733	570
MOTBeyondPixels [95]	RGB camera	✓	0.8268	0.8550	72.61 %	2.92 %	934	581

[1] online/near-online [2] offline/batch

indicates that increasing the dimensionality of the hidden state tends to improve performance, but this trend saturates beyond 64 dimensions.

4.5.3 Comparison with State of the Art

Model and Graph Settings

To compare our proposed approach with existing methods, we train a TrackMPNN model with CWS=5, and incorporate random transformations for data augmentation. We also discard true positive classification, and use the Hungarian algorithm during inference. This model is trained with the same optimizer settings as the baseline for a total of 30 epochs, using 18 of the 21

Table 4.3: Results on the KITTI multi-object tracking benchmark for Pedestrians. Arrows indicate if a higher (↑) or lower value of the metric (↓) is desired. Our results are in bold.

Method	Sensors	Online	MOTA (↑)	MOTP (↑)	MT (↑)	ML (↓)	IDS (↓)	FRAG (↓)
CenterTrack [89]	RGB camera	✓	0.5384	0.7372	35.40 %	21.31 %	425	618
TrackMPNN + CenterTrack	**RGB camera**	✓	**0.5210**	**0.7342**	**35.05 %**	**18.90 %**	**626**	**669**

[1] online/near-online [2] offline/batch

Figure 4.6: Qualitative examples on the KITTI multi-object tracking (MOT) testing set. Each row depicts a sequence of frames, with overlaid color-coded detection boxes. Each color represents a unique track generated by our model.

training sequences provided in KITTI MOT. The remaining 3 sequences were used for validation and early stopping.

KITTI MOT Benchmark

We compare two variants of our approach with other competing methods on the KITTI MOT benchmark in Table 4.2. One variant makes use of RRC detections [109] (TrackMPNN + RRC), and the other uses detections produced by the CenterTrack tracker [89] (TrackMPNN + CenterTrack). Our TrackMPNN + RRC model is trained to track Cars, whereas TrackMPNN + CenterTrack is trained to track all object categories on KITTI i.e., Cars, Pedestrians and Cyclists.

First, our TrackMPNN + RRC model outperforms all competing 2D methods that make use of the same set of detections [88, 92, 95, 96, 114, 115] on the MOTA metric, and remains competitive on other metrics. For RRC detections, we only fall short of JRMOT [113] which makes use of LiDAR point clouds for 3D range information. Next, when using the same set of detections as CenterTrack, we beat every other approach and are comparable in terms of performance to the state-of-the-art. Once again, we do this without relying on any visual cues or pre-training on other datasets as in [89]. The performance of our TrackMPNN + CenterTrack model is also comparable to [89] for tracking Pedestrians (see Table 4.3) when using the same set of detections. While we fall short of CenterTrack on a few metrics, we beat them in some others (MT and ML).

Qualitative Results from KITTI MOT

In addition to the quantitative results presented above, we also show some qualitative results of our method from the KITTI MOT testing set in Figure 4.6. The results depict multiple examples of tracking across frames in a sequence, overlaid with detection boxes. The boxes are color coded to indicate each unique track produced by our model.

4.5.4 Analyzing Learned Attention Weights

As described in Section 4.4.1, we make use of graph attention for detection nodes d to selectively receive messages from neighboring association nodes $\{a_i | a_i \in \mathcal{N}(d)\}$. However, we do not enforce any restrictions on which associations in the neighborhood should receive more/less weights. This is learned by the model when optimizing for the losses during training.

To better understand the nature of these attention weights assigned by the model, we train a baseline model B with multi-head attention comprised of three attention heads. We use the same settings and train-val split as the ablation experiments. The trained model is then run on the validation set, and the attention weights assigned to every association node are grouped based on whether the underlying association is true or false, i.e. if the neighboring detections connected by the association node belong to the same track or not.

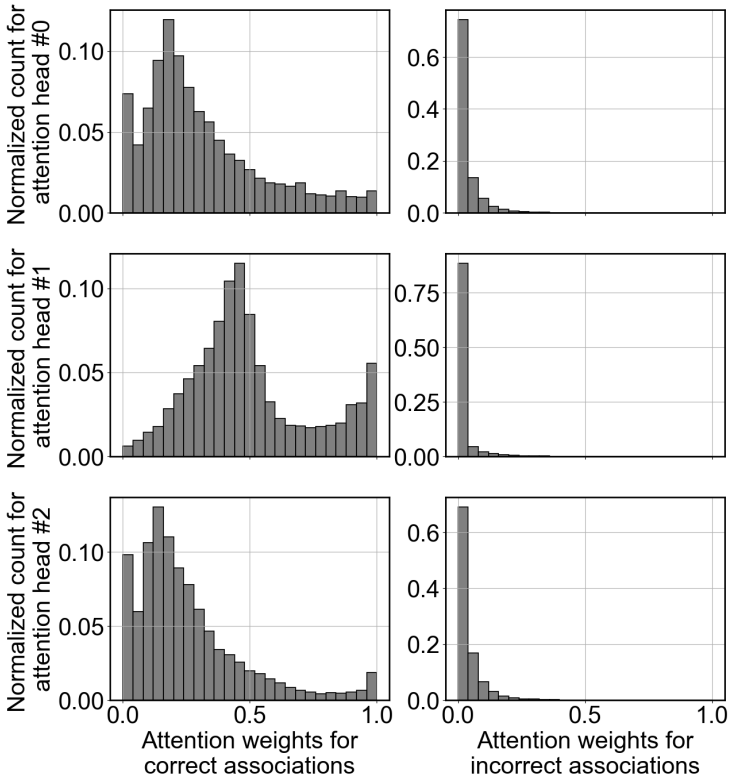
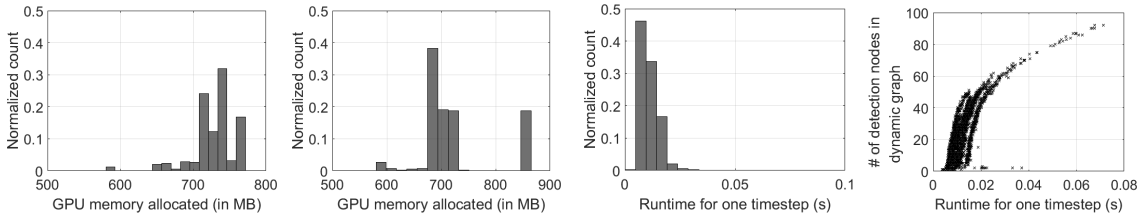


Figure 4.7: Histogram plots of the attention weights assigned to true (left column) and false associations (right column) for each of three attention heads. Each row above corresponds to one of three attention heads.

The distributions of these two separate sets of attention weights (for each of three attention heads) are presented in Figure 4.7. First, we see that higher attention weights are generally assigned to true associations, while those for incorrect associations are skewed towards 0. This implies that messages are mostly exchanged between detection nodes belonging to the same track. Next, we notice different distributions for each attention head, indicating that each head tends to receive different messages from the neighborhood. This is especially true for the second attention head in Figure 4.7. This experiment clearly indicates that the model learns to weight the messages appropriately without any explicit supervision.



(a) Memory allocated during training

(b) Memory allocated during inference

(c) Inference runtime

(d) Number of detection nodes versus runtime

Figure 4.8: Plots examining the compute requirements and runtime of our proposed approach when using RRC detections on the KITTI MOT dataset.

4.5.5 Compute Requirements and Runtime

Since this approach relies on dynamic graphs and operations on them, it is does not have a fixed memory allocation or runtime per timestep. These numbers can change depending on the size and connectivity of the dynamic graph - which in turn depends on the number of objects in the scene and the associations made in the recent past. To keep the memory requirements to a minimum, we make use of sparse operations wherever possible - especially in our vertex update functions. To understand the memory requirements of the model described in Section 4.5.3, we plot histograms of allocated GPU memory during training and inference (Figure 4.8a, 4.8b). From these plots, we notice that despite some spread, the allocated memory remains well within reasonable limits during training and inference (550-900MB). This makes it possible to train and test our model on most modern desktop and laptop GPUs.

Similarly, we plot a histogram of our model runtime on the KITTI MOT test set (Figure 4.8c) using an NVIDIA Titan X Maxwell GPU. This plot demonstrates that our entire framework is capable of operating in real-time, taking only 0.01 seconds on average to process an entire timestep. Finally, to examine the effects of the size of the dynamic graph on runtime, we plot the number of detections nodes in the graph versus the observed runtime in Figure 4.8d. We can clearly see a monotonically increasing relationship between the two; but the runtime increases at a faster rate as more detection nodes are added. This does not pose a problem for the

KITTI dataset, but for larger datasets with highly cluttered scenes, occasional pruning operations can be used to keep the memory allocation within bounds.

4.6 Chapter Summary

This chapter presents a tracking framework based on undirected graphs that represent the data association problem over multiple timesteps in the tracking-by-detection paradigm. In this framework, both individual detections and associations between pairs of them are represented as nodes in the graph. Furthermore, the graph is dynamic, where only detections (and their associations) within a certain temporal range are processed in a rolling window basis. This form of data representation offers any multi-object tracking model the following benefits - multiple competing associations can be considered while scoring any given association from two detections, including associations over multiple timesteps. Information can be stored and propagated across many timesteps through message passing operations. Mistakes in the past can be corrected as long as they are still part of the dynamic graph. False positives, duplicate and missed detections can be handled intrinsically as part of the model.

To illustrate these benefits, we also present a message passing neural network (TrackMPNN) that operates on these dynamic undirected graphs, and train it on the KITTI dataset. Our proposed training and inference schemes make this possible with limited memory and computational bandwidth, and enable real-time operation despite large number of objects in the scene. Experiments, qualitative examples and competitive results on popular MOT benchmarks for autonomous driving demonstrate the promise and uniqueness of the proposed approach.

4.7 Acknowledgements

Chapter 4, in part, is a reprint of the material as it appears in arXiv:2101.04206 (2021), by Akshay Rangesh, Pranav Maheshwari, Mez Gebre, Siddhesh Mhatre, Vahid Ramezani, and Mohan M. Trivedi. The dissertation author was the primary investigator and author of this paper.

Chapter 5

How Would Surround Vehicles Move? A Unified Framework for Maneuver Classification and Motion Prediction

Reliable prediction of surround vehicle motion is a critical requirement for path planning for autonomous vehicles. In this chapter, we present a unified framework for surround vehicle maneuver classification and motion prediction that exploits multiple cues, namely, the estimated motion of vehicles, an understanding of typical motion patterns of freeway traffic and inter-vehicle interaction. We report our results in terms of maneuver classification accuracy and mean and median absolute error of predicted trajectories against the ground truth for real traffic data collected using vehicle mounted sensors on freeways. An ablative analysis is performed to analyze the relative importance of each cue for trajectory prediction. Additionally, an analysis of execution time for the components of the framework is presented. Finally, we present multiple case studies analyzing the outputs of our model for complex traffic scenarios.

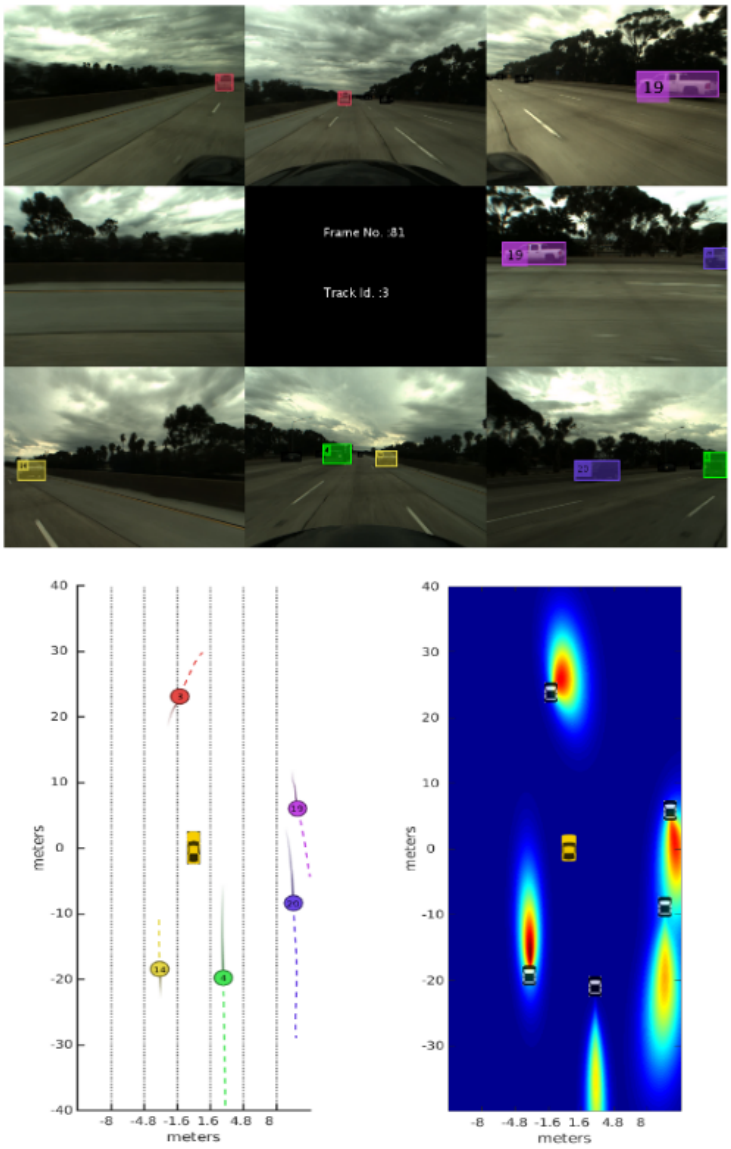


Figure 5.1: To smoothly navigate through freeways an autonomous vehicle must estimate a distribution over the future motion of the surrounding vehicles. We propose a unified model for trajectory prediction that leverages the instantaneous motion of the vehicles, the maneuver being performed by the vehicles and inter-vehicle interactions, while working purely with data captured using vehicle mounted sensors. The above figure shows the data captured by 8 surround cameras **(top)**, the track histories of surround vehicles, the mean predicted trajectories **(bottom left)** and a heat map of the predicted distribution in the ground plane **(bottom right)**.

5.1 Introduction

For successful deployment in challenging traffic scenarios, autonomous vehicles need to ensure the safety of its passengers and other occupants of the road, while navigating smoothly without disrupting traffic or causing discomfort to its passengers. Existing tactical path planning algorithms [116–118] hinge upon reliable estimation of future motion of surrounding vehicles over a prediction horizon of up to 10 s. While approaches leveraging vehicle-to-vehicle communication [119–121], offer a possible solution, these would require widespread adoption of autonomous driving technology in order to become viable. In order to safely share the road with human drivers, an autonomous vehicle needs to have the ability to predict the future motion of surrounding vehicles purely based on perception. Thus, we address the problem of surround vehicle motion prediction purely based on data captured using vehicle mounted sensors.

Prediction of surround vehicle motion is an extremely challenging problem due to a large number of factors that affect the future trajectories of vehicles. Prior works addressing the problem seem to incorporate three cues in particular: the instantaneous estimated motion of surround vehicles, an understanding of typical motion patterns of traffic and inter-vehicle interaction. A large body of work uses the estimated state of motion of surround vehicles along with a kinematic model to make predictions of their future trajectories [122–129]. While these approaches are computationally efficient, they become less reliable for long term prediction, since they fail to model drivers as decision making entities capable of changing the motion of vehicles over long intervals. An alternative is offered by probabilistic trajectory prediction approaches [130–135] that learn typical motion patterns of traffic from a trajectory dataset. However these approaches are prone to poorly modeling safety critical motion patterns that are under represented in the training data. Many works address these shortcomings of motion models and probabilistic models by defining a set of semantically interpretable *maneuvers* [8, 136–143]. A separate motion model or probabilistic model can then be defined for each maneuver for making future predictions.

Finally some works leverage inter-vehicle interaction for making trajectory predictions [143, 144].

While many promising solutions have been proposed, they seem to have the following limitations. (i) Most works consider a restrictive setting such as only predicting longitudinal motion, a small subset of motion patterns, or specific cases of inter-vehicle interaction, whereas many of the biggest challenges for vehicle trajectory prediction originate from the generalized setting of simultaneous prediction of the complete motion of all vehicles in the scene. (ii) Many approaches have been evaluated using simulated data, or based on differential GPS, IMU readings of target vehicles, whereas evaluation using real traffic data captured using perceptual vehicle mounted sensors is more faithful to the setting being considered. (iii) There is a lack of a unifying approach that combines each of the three cues mentioned above and analyzes their relative importance for trajectory prediction.

In this work, we propose a framework for holistic surround vehicle trajectory prediction based on three interacting modules: A hidden Markov model (HMM) based maneuver recognition module for assigning confidence values for maneuvers being performed by surround vehicles, a trajectory prediction module based on the amalgamation of an interacting multiple model (IMM) based motion model and maneuver specific variational Gaussian mixture models (VGMMs), and a vehicle interaction module that considers the global context of surround vehicles and assigns final predictions by minimizing an energy function based on outputs of the other two modules. We work with vehicle tracks obtained using 8 vehicle mounted cameras capturing the full surround and generate the mean predicted trajectories and prediction uncertainties for all vehicles in the scene as shown in Figure 5.1. We evaluate the model using real data captured on Californian freeways.

The main contributions of this work are:

1. A unified framework for surround vehicle trajectory prediction that exploits instantaneous vehicle motion, an understanding of typical motion patterns of traffic and inter-vehicle interaction.

2. An ablative analysis for determining the relative importance of each cue in trajectory prediction.

3. Evaluation based on real traffic data captured using vehicle mounted sensors.

5.2 Related Research

Data-Driven Trajectory Prediction

Data driven trajectory prediction approaches can be broadly classified into clustering based approaches and probabilistic approaches. Clustering based approaches [136, 145–147] cluster the training data to give a set of prototype trajectories. Partially observed trajectories are matched with a prototype trajectory based on distance measures such as DTW, LCSS or Hausdorff distance, and the prototype trajectory used as a model for future motion. The main drawback of clustering based approaches is the deterministic nature of the predictions. Probabilistic approaches in contrast, learn a probability distribution over motion patterns and output the conditional distribution over future motion given partial trajectories. These have the added advantage of associating a degree of uncertainty to the future predictions. Gaussian Processes are the most popular approach for modeling trajectories [131–133]. Other approaches include [134] and [135] where the authors use Gaussian mixture regression for predicting the longitudinal and lateral motion of vehicles respectively. Of particular interest is the work by Weist *et al.* [130] who use variational Gaussian mixture models (VGMMs) to model the conditional distribution over snippets of trajectory futures given snippets of trajectory history. This approach is much leaner and computationally efficient as compared to Gaussian process regression and was shown to be effective at predicting the highly non-linear motion in turns at intersections. While Weist *et al.* use the velocity and yaw angle of the predicted vehicle obtained from its Differential GPS data, we extend this approach by learning VGMMs for freeway traffic using positions and velocities of surround vehicles estimated using vehicle mounted sensors, similar to our prior work

on pedestrian trajectory prediction [148].

Maneuver-based Trajectory Prediction

Classification of vehicle motion into semantically interpretable maneuver classes has been extensively addressed in both advanced driver assistance systems as well as naturalistic drive studies [8, 132–134, 136–143]. Most approaches involve using heuristics [141] or training classifiers such as SVMs [138, 139], HMMs [8, 133, 136, 137], LSTMs [140] and Bayesian networks [142] using motion based features such as speed, acceleration, yaw rate and other context information such as lane position, turn signals, distance from leading vehicle.

The works most closely related to our approach are those that use the recognized maneuvers to make predictions of future trajectories. Houenou *et al.* [141] classify a vehicle's motion as a keep lane or lane change maneuver based on distance to nearest lane marking and predict the future trajectory by fitting a quintic polynomial between the current motion state of the vehicle and a pre-defined final motion state for each maneuver class. Schreier *et al.* [142] classify vehicle motion into one of six different maneuver classes using a Bayesian network based on multiple motion and context based features. A class specific motion model is then defined for each maneuver to generate future trajectories. Most similar in principle to our approach are [134], [132] and [133] where separate probabilistic prediction models are trained for each maneuver class. Tran and Firl [132] define a separate Gaussian process for three maneuver classes and generate a multi-modal distribution over future trajectories using each model. However, only case based evaluation has been presented. Laugier *et al.* [133] also define separate Gaussian processes for 4 different maneuvers that are classified using a hierarchical HMM. While they report results for maneuver classification on real highway data, they evaluate trajectory prediction in the context of risk assessment simulated data. Schlechtriemen *et al.* [134] use a random forest classifier to classify maneuvers into left or right lane changes or keep lane. They use a separate Gaussian mixture regression model for making predictions of lateral movement of vehicles for each class,

reporting results on real highway data. Along similar lines, but without maneuver classes, they also predict longitudinal motion for surround vehicles [135]. Contrary to this approach, we make predictions for the complete motion of vehicles based on maneuver class, since detection of certain maneuvers like overtakes can help predict both lateral and longitudinal motion of vehicles.

Interaction-Aware Trajectory Prediction

Relatively few works address the effect of inter-vehicle interaction in trajectory prediction. Kafer *et al.* [143] jointly assign maneuver classes for two vehicles approaching an intersection using a polynomial classifier that penalizes cases which would lead to near collisions. Closer to our proposed approach, Lawitzky *et al.* [144] consider the much more complex case of assigning maneuver classes to multiple interacting vehicles in a highway setting. However, predicted trajectories and states of vehicle motion are assumed to be given, and results reported using a simulated setting. Contrarily, our evaluation considers the combined complexity due to multiple interacting vehicles as well the difficulty of estimating their future motion. We note that inter-vehicle interaction is implicitly modeled in [134] by including relative positions and velocities of nearby vehicles as features for maneuver classification and trajectory prediction.

5.3 Overview

Figure 5.2 shows the complete pipeline of our proposed approach. We restrict our setting to purely perception based prediction of surround vehicle motion, without any vehicle-to-vehicle communication. Toward this end, the ego vehicle is equipped with 8 cameras that capture the full surround. All vehicles within 40 m of the ego vehicle in the longitudinal direction are tracked for motion analysis and prediction. While vehicle tracking is not the focus of this work, we refer readers to a multi-perspective vision based vehicle trackers described in [8, 98]. The tracked vehicle locations are then projected to the *ground plane* to generate track histories of the surround

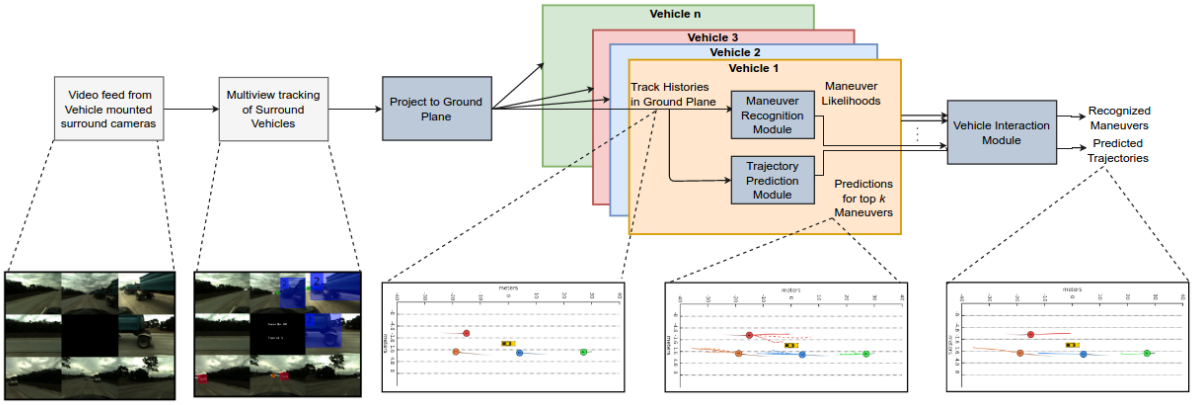


Figure 5.2: **Overview of the proposed model:** Track histories of all surround vehicles are obtained via a multi-perspective tracker and projected to the ground plane in the ego vehicle's frame of reference. The model consists of three interacting modules: The maneuver recognition module assigns confidence values to possible maneuvers being performed by each vehicle. The trajectory prediction module outputs future trajectories for each maneuver class. The vehicle interaction module assigns the true recognized maneuver for each vehicle by combining the confidence values provided by the maneuver recognition module and the feasibility of predicted trajectories given the relative configuration of all vehicles

vehicles in the frame of reference of the ego vehicle.

The goal of our model is to estimate the future positions and the associated prediction uncertainty for all vehicles in the ego vehicle's frame of reference over the next t_f seconds, given a t_h second snippet of their most recent track histories. The model essentially consists of three interacting modules, namely the *trajectory prediction module*, the *maneuver recognition module* and the *vehicle interaction module*. The trajectory prediction module is the most crucial among the three and can function as a standalone block independent of the remaining two modules. It outputs a linear combination of the trajectories predicted by a motion model that leverages the estimated instantaneous motion of the surround vehicles and a probabilistic trajectory prediction model which learns motion patterns of vehicles on freeways from a freeway trajectory training set. We use constant velocity (CV), constant acceleration (CA) and constant turn rate and velocity (CTRV) models in the interacting multiple model (IMM) framework as the motion models since these capture most instances of freeway motion, especially in light traffic conditions. We use Variational Gaussian Mixture Models (VGMM) for probabilistic trajectory prediction owing to

promising results for vehicle trajectory prediction at intersections shown in [130].

The motion model becomes unreliable for long term trajectory prediction, especially in cases involving a greater degree of decision making by drivers such as overtakes, cut-ins or heavy traffic conditions. These cases are critical from a safety stand-point. However, since these are relatively rare occurrences, they tend to be poorly modeled by a monolithic probabilistic prediction model. Thus we bin surround vehicle motion on freeways into 10 maneuver classes, with each class capturing a distinct pattern of motion that can be useful for future prediction. The intra-maneuver variability of vehicle motion is captured through a VGMM learned for each maneuver class. The maneuver recognition module recognizes the maneuver being performed by a vehicle based on a snippet of it's most recent track history. We use hidden Markov models (HMM) for this purpose. The VGMM corresponding to the most likely maneuver can then be used for predicting the future trajectory. Thus the maneuver recognition and trajectory Prediction modules can be used in conjunction for each vehicle to make more reliable predictions.

Up to this point, our model predicts trajectories of vehicles independent of each other. However the relative configuration of all vehicles in the scene can make certain maneuvers infeasible and certain others more likely, making it a useful cue for trajectory prediction especially in heavy traffic conditions. The vehicle interaction module (VIM) leverages this cue. The maneuver likelihoods and predicted trajectories for the K likeliest maneuvers for each vehicle being tracked are passed to the VIM. The VIM consists of a Markov random field aimed at optimizing an energy function over the discrete space of maneuver classes for all vehicles in the scene. The energy function takes into account the confidence values for all maneuvers given by the HMM and the feasibility of the maneuvers given the relative configuration of all vehicles in the scene. Minimizing the energy function gives the recognized maneuvers and corresponding trajectory predictions for all vehicles in the scene.

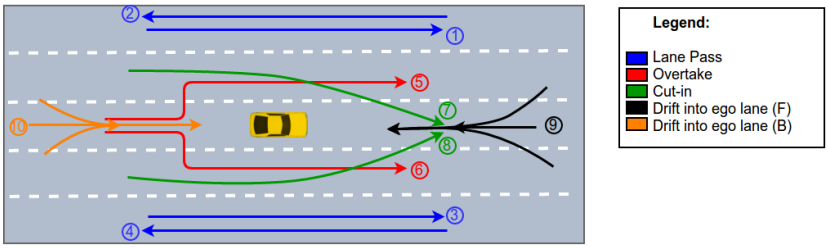


Figure 5.3: **Maneuver Classes for Freeway Traffic:** We bin the trajectories of surround vehicles in the ego-vehicle frame of reference into 10 maneuver classes: 4 lane pass maneuvers, 2 overtake maneuvers, 2 cut-in maneuvers and 2 maneuvers involving drifting into ego vehicle lane.

5.4 Maneuver Recognition Module

5.4.1 Maneuver classes

We define 10 maneuver classes for surround vehicle motion on freeways in the ego-vehicle's frame of reference. Figure 5.3 illustrates the maneuver classes.

1. *Lane Passes*: Lane pass maneuvers involve vehicles passing the ego vehicle without interacting with the ego vehicle lane. These constitute a majority of the surround vehicle motion on freeways and are relatively easy cases for trajectory prediction owing to approximately constant velocity profiles. We define 4 different lane pass maneuvers as shown in Figure 5.3

2. *Overtakes*: Overtakes start with the surround vehicle behind the ego vehicle in the ego lane. The surround vehicle changes lane and accelerates in order to pass the ego vehicle. We define 2 different overtake maneuvers, depending on which side the the surround vehicle overtakes.

3. *Cut-ins*: Cut-ins involve a surround vehicle passing the ego vehicle and entering the ego lane in front of the ego-vehicle. Cut-ins and overtakes, though relatively rare, can be critical from a safety stand-point and also prove to be challenging cases for trajectory prediction.

We define 2 different cut-ins depending on which side the surround vehicle cuts in from.

4. *Drift into Ego Lane:* Another important maneuver class is when a surround vehicle drifts into the ego vehicle lane in front or behind the ego vehicle. This is also important from a safety standpoint as it directly affects how sharply the ego vehicle can accelerate or decelerate. A separate class is defined for drifts into ego-lane in front and to the rear of the ego vehicle.

5.4.2 Hidden Markov Models

Hidden Markov models (HMMs) have previously been used for maneuver recognition [8, 136, 137] due to their ability to capture the spatial and temporal variability of trajectories. HMMs can be thought of as combining two stochastic models, an underlying Markov chain of states characterized by state transition probabilities and an emission probability distribution over the feature space for each state. The transition probabilities model the temporal variability of trajectories while the emission probabilities model the spatial variability, making HMMs a viable approach for maneuver recognition.

Previous works [8, 136] use HMMs for classifying maneuvers after they have been performed, where the HMM for a particular maneuver is trained using complete trajectories belonging to that maneuver class. In our case, the HMMs need to classify a maneuver based on a small t_h second snippet of the trajectory. Berndt *et al.* [137] address the problem of maneuver classification based on partially observed trajectories by using only the initial states of a trained HMM to fit the observed trajectory. However, this approach requires prior knowledge of the starting point of the maneuver. In our case, the trajectory snippet could be from any point in the maneuver, and not necessarily the start. We need the HMM to classify a maneuver based on any intermediate snippet of the trajectory. We thus divide the trajectories in our training data into overlapping snippets of t_h seconds and train the maneuver HMMs using these snippets.

For each maneuver, we train a separate HMM with a left-right topology with only self transitions and transitions to the next state. The state emission probabilities are modeled as mixtures of Gaussians with diagonal covariances. The x and y ground plane co-ordinates and instantaneous velocities in the x and y direction are used as features for training the HMMs. The parameters of the HMMs: the state transition probabilities and the means, variances and weights of the mixture components are estimated using the Baum-Welch algorithm [149].

For a car i, the HMM for maneuver k outputs the log likelihood:

$$\mathcal{L}_k^i = \log(P(\mathbf{x}_h^i, \mathbf{y}_h^i, \mathbf{v}_{\mathbf{x}_h}^i, \mathbf{v}_{\mathbf{y}_h}^i | m^i = k; \Theta_k)) \tag{5.1}$$

where $\mathbf{x}_h^i$, $\mathbf{y}_h^i$ are the x and y locations of vehicle i over the last t_h seconds and $\mathbf{v}_{\mathbf{x}_h}^i$, $\mathbf{v}_{\mathbf{y}_h}^i$ are the velocities along the x and y directions over the last t_h seconds. m^i is the maneuver assigned to car i and Θ_k are the parameters of the HMM for maneuver k

5.5 Trajectory Prediction Module

The trajectory prediction module predicts the future x and y locations of surround vehicles over a prediction horizon of t_f seconds and assigns an uncertainty to the predicted locations in the form of a 2×2 covariance matrix. It averages the predicted future locations and covariances given by both a motion model, and a probabilistic trajectory prediction model. The outputs of the trajectory prediction module for a prediction instant t_{pred} are given by:

$$x_f(t) = \frac{1}{2}\left(x_{f_{motion}}(t) + x_{f_{prob}}(t)\right) \tag{5.2}$$

$$y_f(t) = \frac{1}{2}\left(y_{f_{motion}}(t) + y_{f_{prob}}(t)\right) \tag{5.3}$$

$$\Sigma_f(t) = \frac{1}{2}\left(\Sigma_{f_{motion}}(t) + \Sigma_{f_{prob}}(t)\right) \tag{5.4}$$

where $t_{pred} \leq t \leq t_{pred} + t_f$

5.5.1 Motion Models

We use the interacting multiple model (IMM) framework for modeling vehicle motion, similar to [128], [129]. The IMM framework allows for combining an ensemble of Bayesian filters for motion estimation and prediction by weighing the models with probability values. The probability values are estimated at each time step based on the transition probabilities of an underlying Markov model and how well each model fits the observed motion prior to that time step. We use the following motion models in our ensemble:

1. *Constant velocity (CV):* The constant velocity models maintains an estimate of the position and velocity of the surround vehicles under the constraint that the vehicles move with a constant velocity. We use a Kalman filter for estimating the state and observations of the CV model. The CV model captures a majority of freeway vehicle motion.

2. *Constant acceleration (CA):* The constant acceleration model maintains estimates of the the vehicle position, velocity and acceleration under the constant acceleration assumption using a Kalman Filter. The CA model can be useful for describing freeway motion especially in dense traffic.

3. *Constant turn rate and velocity (CTRV):* The constant turn rate and velocity model maintains estimates of the the vehicle position, orientation and velocity magnitude under the constant yaw rate and velocity assumption. Since the state update for the CTRV model is non-linear, we use an extended Kalman filter for estimating the state and observations of the CTRV model. The CTRV model can be useful for modeling motion during lane changes

5.5.2 Probabilistic Trajectory Prediction

We formulate probabilistic trajectory prediction as estimating the conditional distribution:

$$P(\mathbf{v}_{\mathbf{x}f}, \mathbf{v}_{\mathbf{y}f} | \mathbf{x}_h, \mathbf{y}_h, \mathbf{v}_{\mathbf{x}h}, \mathbf{v}_{\mathbf{y}h}, m) \tag{5.5}$$

i.e. the conditional distribution of the vehicle's predicted velocities given the vehicles past positions, velocities and maneuver class. In particular, we are interested in estimating the conditional expected values $[\hat{\mathbf{v}}_{\mathbf{x}f}; \hat{\mathbf{v}}_{\mathbf{y}f}]$ and conditional covariance Σ_{v_f} of the distribution 5.5. The predicted locations and $\mathbf{x}_{f\,prob}$, $\mathbf{y}_{f\,prob}$ can then be obtained by taking the cumulative sum of the predicted velocities, which can be represented using an accumulator matrix $\mathbf{A}$

$$[\mathbf{x}_{f\,prob}; \mathbf{y}_{f\,prob}] = \mathbf{A}[\hat{\mathbf{v}}_{\mathbf{x}f}; \hat{\mathbf{v}}_{\mathbf{y}f}] \tag{5.6}$$

Similarly, the uncertainty of prediction Σ_{prob} can be obtained using the expression:

$$\Sigma_{f\,prob} = \mathbf{A}\Sigma_{v_f}\mathbf{A}^T \tag{5.7}$$

We use the framework proposed by Weist *et al.* [130] for estimating the conditional distribution 5.5. ($\mathbf{x}_h$,$\mathbf{y}_h$,$\mathbf{v}_{\mathbf{x}h}$,$\mathbf{v}_{\mathbf{y}h}$) and ($\mathbf{v}_{\mathbf{x}f}$,$\mathbf{v}_{\mathbf{y}f}$) are represented in terms of their Chebyshev coefficients, $\mathbf{c}_h$ and $\mathbf{c}_f$. The joint distribution $P(\mathbf{c}_f, \mathbf{c}_h | m)$ for each maneuver class is estimated as the predictive distribution of a variational Gaussian mixture model (VGMM). The conditional distribution $P(\mathbf{c}_f | \mathbf{c}_h, m)$ can then be estimated in terms of the parameters of the predictive distribution. We briefly review the the expressions for $P(\mathbf{c}_f, \mathbf{c}_h | m)$ and $P(\mathbf{c}_f | \mathbf{c}_h, m)$ here. However, the reader is encouraged to refer to [130] for a more detailed treatment.

VGMMs are the Bayesian analogue to standard GMMs, where the model parameters, $\{\pi$, $\boldsymbol{\mu}_1$, $\boldsymbol{\mu}_2$, ... $\boldsymbol{\mu}_K$, $\boldsymbol{\Lambda}_1$, $\boldsymbol{\Lambda}_2$, ... $\boldsymbol{\Lambda}_K\}$ are given conjugate prior distributions. The prior over mixture

weights π is a Dirichlet distribution

$$P(\pi) = Dir(\pi|\alpha_{\mathbf{0}}) \tag{5.8}$$

The prior over each component mean $\boldsymbol{\mu}_k$ and component precision $\boldsymbol{\Lambda}_k$ is an independent Gauss-Wishart distribution

$$P(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k) = \mathcal{N}(\boldsymbol{\mu}_k|\mathbf{m}_{0_k}, (\beta_{0_k}\boldsymbol{\Lambda}_k)^{-1})\mathcal{W}(\boldsymbol{\Lambda}_k|\mathbf{W}_{0_k}, \nu_{0_k}) \tag{5.9}$$

The parameters of the posterior distributions are estimated using the Variational Bayesian Expectation Maximization algorithm [150]. The predictive distribution for a VGMM is given by a mixture of Student's t-distributions

$$P(\mathbf{c}_h, \mathbf{c}_f) = \frac{1}{sum(\alpha)} \sum_{k=1}^{K} \alpha_k St(\mathbf{c}_h, \mathbf{c}_f|\mathbf{m}_k, \mathbf{L}_k, \nu_k + 1 - d) \tag{5.10}$$

where d is the number of degrees of freedom of the Wishart distribution and

$$\mathbf{L}_k = \frac{(\nu_k + 1 - d)\beta_k}{1 + \beta_k}\mathbf{W}_k \tag{5.11}$$

For a new trajectory history $\mathbf{c_h}$, the conditional predictive distribution $P(\mathbf{c_f}|\mathbf{c_h})$ is given by:

$$P(\mathbf{c}_f|\mathbf{c}_h) = \frac{1}{sum(\hat{\alpha})} \sum_{k=1}^{K} \hat{\alpha}_k St\left(\mathbf{c}_f|\mathbf{c}_h, \hat{\mathbf{m}}_k, \mathbf{L}_k, \nu_k + 1 - d\right) \tag{5.12}$$

where (5.13)

$$\hat{\nu_k} = \nu_k + 1 - d \tag{5.14}$$

$$\hat{\alpha_k} = \frac{\alpha_k St(\mathbf{c}_h|\mathbf{m}_{k,\mathbf{c}_h}, \mathbf{L}_{k,\mathbf{c}_h}, \hat{\nu}_k)}{\sum_{j=1}^{K} \alpha_j St(\mathbf{c}_h|\mathbf{m}_{j,\mathbf{c}_h}, \mathbf{L}_{j,\mathbf{c}_h}, \hat{\nu}_j)} \tag{5.15}$$

$$\hat{\mathbf{m}}_k = \mathbf{m}_{k,\mathbf{c}_f} + \Sigma_{k,\mathbf{c}_f\mathbf{c}_h} \Sigma^{-1}_{k,\mathbf{c}_h\mathbf{c}_h} (\mathbf{c}_h - \mathbf{m}_{k,\mathbf{c}_h}) \tag{5.16}$$

$$\hat{\mathbf{L}}^{-1}_k = \frac{\hat{\nu}_k}{\hat{\nu}_k + d - 2} \left(1 + \Delta_k^T \frac{\Sigma_{k,\mathbf{c}_h\mathbf{c}_h}}{\hat{\nu}_k} \Delta_k\right) \Sigma^*_k \tag{5.17}$$

$$\Delta_k = (\mathbf{c}_h - \mathbf{m}_{k,\mathbf{c}_h}) \tag{5.18}$$

$$\Sigma^*_k = \Sigma_{k,\mathbf{c}_f\mathbf{c}_f} - \Sigma_{k,\mathbf{c}_f\mathbf{c}_h} \Sigma_{k,\mathbf{c}_h\mathbf{c}_h^{-1}} \Sigma_{k,\mathbf{c}_h\mathbf{c}_f} \tag{5.19}$$

$$\Sigma_k = \frac{\hat{\nu}_k + d - 2}{\hat{\nu}_k + d} \mathbf{L}^{-1}_k \tag{5.20}$$

5.6 Vehicle Interaction Module

The vehicle interaction module is tasked with assigning discrete maneuver labels to all vehicles in the scene at a particular prediction instant based on the confidence of the HMM in each maneuver class and the feasibility of the future trajectories of all vehicles based on those maneuvers given the current configuration of all vehicles in the scene. We set this up as an energy minimization problem. For a given prediction instant, let there be N surround vehicles in the scene with the top K maneuvers given by the HMM being considered for each vehicle. The minimization objective is given by:

$$\mathbf{y}^* = \arg\min_{\mathbf{y}} \sum_{i=1}^{n} \sum_{k=1}^{K} y^i_k \left[E^{hmm}_{ik} + \lambda E^{ego}_{ik}\right] + \lambda \sum_{i=1}^{n} \sum_{k=1}^{K} \sum_{\substack{j=1 \\ j\neq i}}^{n} \sum_{l=1}^{K} y^i_k y^j_l E^{vi}_{ijkl} \tag{5.21}$$

s.t.

$$\sum_k y_k^i = 1 \quad \forall i \tag{5.22}$$

$$y_k^i = \begin{cases} 1, & \text{if car } i \text{ is assigned maneuver } k \\ 0, & \text{otherwise} \end{cases} \tag{5.23}$$

The objective consists of three types of energies, the individual Energy terms E_{ik}^{hmm}, E_{ik}^{ego} and the pairwise energy terms E_{ijkl}^{vi}. The individual energy terms E_{ik}^{hmm} are given by the negative of the log likelihoods provided by the HMM. Higher the confidence of an HMM in a particular maneuver, lower is $-\mathcal{L}_k^i$ and thus the individual energy term. The individual energy term E_{ik}^{ego} takes into account the interaction between surround vehicles and the ego vehicle. We define the E_{ik}^{ego} as the reciprocal of the closest point of approach for vehicle i and the ego vehicle over the entire prediction horizon, given that it is performing maneuver k, where the ego vehicle position is always fixed to 0, since it is the origin of the frame of reference. Similarly, the pairwise energy term E_{ijkl}^{vi} is defined as the reciprocal of the minimum distance between the corresponding predicted trajectories for the vehicles i and j, assuming them to be performing maneuvers k and l respectively . The terms E_{ik}^{ego} and E_{ijkl}^{vi} penalize predictions where at any point in the prediction horizon, two vehicles are very close to each other. This term leverages the fact that drivers tend to follow paths with low possibility of collisions with other vehicles. The weighting constant λ is experimentally determined through cross-validation.

The minimization objective in the formulation shown in Equations 5.21, 5.22 and 5.23 has quadratic terms in y values. In order to leverage integer linear programming for minimizing the energy, we modify the formulation as follows:

$$\mathbf{y}^*, \mathbf{z}^* = \arg\min_{\mathbf{y},\mathbf{z}} \sum_{i=1}^{n} \sum_{k=1}^{K} y_k^i \left[E_{ik}^{hmm} + \lambda E_{ik}^{ego} \right] + \lambda \sum_{i=1}^{n} \sum_{k=1}^{K} \sum_{\substack{j=1 \\ j \neq i}}^{n} \sum_{l=1}^{K} z_{k,l}^{i,j} E_{ijkl}^{vi} \tag{5.24}$$

s.t.

$$\sum_k y_k^i = 1 \quad \forall i \tag{5.25}$$

$$y_k^i \in 0,1 \tag{5.26}$$

$$z_{k,l}^{i,j} \leq y_k^i \tag{5.27}$$

$$z_{k,l}^{i,j} \leq y_l^j \tag{5.28}$$

$$z_{k,l}^{i,j} \geq y_k^i + y_l^j - 1 \tag{5.29}$$

This objective can now be optimized using integer linear programming, where the optimal values $\mathbf{y}^*$ give the maneuver assignments for each of the vehicles. These assigned maneuver classes are used by the trajectory prediction module to make future predictions for all vehicles.

5.7 Experimental Evaluation

5.7.1 Dataset

Table 5.1: Trajectory prediction dataset statistics.

Maneuver	**Number of trajectories**	**Number of trajectory snippets**
Lane Pass (Left Forward)	59	9500
Lane Pass (Left Back)	75	10332
Lane Pass (Right Forward)	110	10123
Lane Pass (Right Back)	48	12523
Overtake (Left)	8	1629
Overtake (Right)	17	2840
Cut-in (Left)	8	1667
Cut-in (Right)	19	3201
Drift into ego lane (Front)	11	1317
Drift into ego lane (Rear)	8	553

We evaluate our framework using real freeway traffic data captured using the testbed

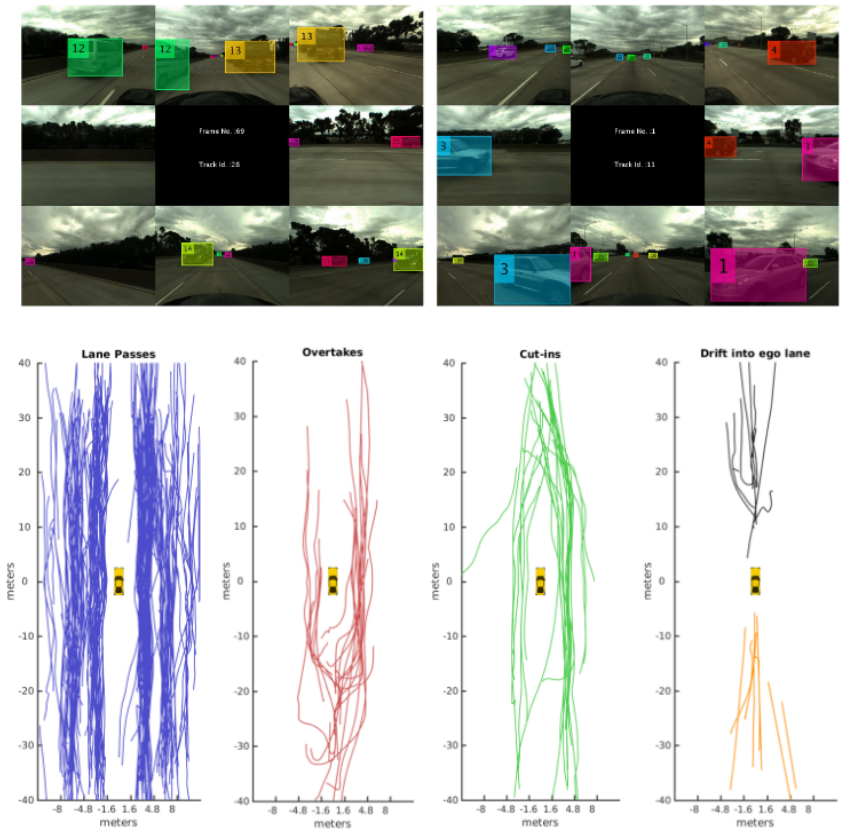


Figure 5.4: **Dataset:** Examples of annotated frames from the evaluation set (top left and top right) and trajectories belonging to all maneuver classes projected in the ground plane (bottom). We can observe that the trajectory patterns implicitly capture lane information

described in [45]. The vehicle is equipped with 8 RGB video cameras, LIDARs and RADARs synchronously capturing the full surround at a frame rate of 15 fps. Our complete dataset consists of 52 video sequences extracted from multiple drives spanning approximately 45 minutes. The sequences were chosen to capture varying lighting conditions, vehicle types, and traffic density and behavior.

The 4 longest video sequences, of about 3 minutes each were ground-truthed by human annotators and used for evaluation. Three sequences from the evaluation set represent light to moderate or *free-flowing* traffic conditions, while the remaining sequence represents heavy or *stop-and-go* traffic. The video feed from the evaluation set was annotated with detection boxes and vehicle track-ids for each of the 8 views. All tracks were then projected to the ground plane and assigned a maneuver class label corresponding to the 10 maneuver classes described in Section 5.4.1. If a vehicle track was comprised by multiple maneuvers, the start and end-point of each maneuver was marked. A multi-perspective tracker [8] was used for assigning vehicle tracks for the remaining 48 sequences. These tracks were only used for training the models. Figure 5.4 shows the track annotations as well as the complete set of trajectories belonging to each maneuver class. Since each trajectory is divided into overlapping snippets of $t_h = 3$ seconds for training and testing our models, we report the data statistics in terms of the total number of trajectories as well as the number of trajectory snippets belonging to each maneuver class in Table 5.1.

We report all results using a leave on sequence cross-validation scheme. For each of the 4 evaluation sequences, the HMMs and VGMMs are trained using data from the remaining 3 evaluation sequences as well as the 48 training sequences. Additionally, we use two simple data-augmentation schemes for increasing the size of our training datasets in order to reduce overfitting in the models:

1. *Lateral inversion:* We flip each trajectory along the lateral direction in the ego frame to give an instance of a different maneuver class. For example, a left cut-in on lateral inversion becomes a right cut in.

Table 5.2: Quantitative results showing ablative analysis of our proposed model.

Metric	Setting	All Trajectories				Overtakes and Cut-ins			Stop-and-Go Traffic		
	Prediction Horizon (s)	IMM	M-VGMM	C-VGMM	C-VGMM + VIM	IMM	M-VGMM	C-VGMM	IMM	C-VGMM	C-VGMM +VIM
Mean Absolute Error (m)	1	0.25	**0.24**	**0.24**	**0.24**	0.29	0.32	**0.29**	0.22	0.20	**0.20**
	2	0.72	0.70	**0.69**	**0.69**	0.83	0.87	**0.82**	0.68	0.65	**0.64**
	3	1.25	1.19	**1.18**	**1.18**	1.47	1.46	**1.39**	1.21	1.17	**1.14**
	4	1.78	1.70	1.68	**1.66**	2.17	2.05	**1.94**	1.74	1.68	**1.65**
	5	2.36	2.24	2.20	**2.18**	2.90	2.68	**2.49**	2.29	2.21	**2.17**
Median Absolute Error (m)	1	0.19	**0.17**	**0.17**	**0.17**	**0.23**	**0.23**	**0.23**	0.15	**0.13**	**0.13**
	2	0.55	**0.52**	**0.52**	**0.52**	0.68	**0.65**	**0.65**	0.48	0.46	**0.45**
	3	0.96	0.92	**0.91**	**0.91**	1.24	1.13	**1.12**	0.89	0.87	**0.83**
	4	1.38	1.32	1.30	**1.29**	1.92	1.71	**1.68**	1.32	1.29	**1.27**
	5	1.85	1.77	**1.72**	**1.72**	2.64	2.27	**2.12**	1.8	1.78	**1.75**
Class. acc. (%)	-	-	-	83.49	**84.24**	-	-	55.89	-	84.84	**87.19**
Exec. time (s)	-	**0.0346**	0.1241	0.0891	0.1546	-	-	-	-	-	-

2. *Longitudinal shifts:* We shift each of the trajectories by $\pm$ 2, 4 and 6 m in the longitudinal direction in the ego frame to give additional instances of the same maneuver class. We avoid lateral shifts since this would interfere with lane information that is implicitly learned by the probabilistic model.

5.7.2 Evaluation Measures and Experimental Settings

Our models predict the future trajectory over a prediction horizon of 5 seconds for each 3 second snippet of track history based on the maneuver classified by the HMMs or by the VIM. We use the following evaluation measures for reporting our results:

1. *Mean Absolute Error*: This measure gives the average absolute deviation of the predicted trajectories from the underlying ground truth trajectories. To compare how the models perform for short term and long term predictions, we report this measure separately for prediction instants up to 5 seconds into the future, sampled with increments of 1 second. The mean absolute error captures the effect of both the number of errors made by the models as well as the severity of the errors.

2. *Median Absolute Error*: We also report the median values of the absolute deviations for up to 5 seconds into the future with 1 second increments, as was done in [135]. The median

absolute error better captures the distribution of the errors made by the models while sifting out the effect of a few drastic errors.

3. *Maneuver classification accuracy*: We report maneuver classification accuracy for configurations using the maneuver recognition module or the vehicle interaction module.

4. *Execution time*: We report the average execution time per frame, where each frame involves predicting trajectories of all vehicles being tracked at a particular instant.

In order to analyze the effect of each of our proposed modules, we compare the trajectory prediction results for following systems

- *Motion model (**IMM**):* We use the trajectories predicted by the IMM based motion model as our baseline.

- *Monolithic VGMM (**M-VGMM**):* We consider the trajectories predicted by our trajectory prediction module, where the probabilistic model used is a single monolithic VGMM. This alleviates the need for the maneuver recognition module, since the same model makes predictions irrespective of the maneuver being performed

- *Class VGMMs (**C-VGMM**):* Here we consider separate VGMMs for each maneuver class in the trajectory prediction module. We use the VGMM corresponding to the maneuver with the highest HMM log likelihood for making the prediction. In this case, maneuver predictions for each vehicle are made independent of the other vehicles in the scene. To keep the comparison with the M-VGMM fair, we use 8 mixture components for each maneuver class for the C-VGMMs, while we use a single VGMM with 80 mixture components for the M-VGMM, ensuring that both models have the same complexity.

- *Class VGMMs with Vehicle Interaction Module (**C-VGMM + VIM**):* We finally consider the effect of using the vehicle interaction module. In this case, we use the C-VGMMs

with the maneuver classes for each of the vehicles in the scene assigned by the vehicle interaction module

We report our results for the complete set of trajectories in the evaluation set. Additionally, we also report results on the subsets of overtake and cut-in maneuvers and stop-and-go traffic. Since overtakes and cut-ins are rare safety critical maneuvers with significant deviation from uniform motion, these are challenging cases for trajectory prediction. Similarly, due to the high traffic density in stop-and-go scenarios, vehicles affect each others motion to a much greater extent as compared to free-flowing traffic, making it a challenging scenario for trajectory prediction.

5.7.3 Ablative Analysis

Table 5.2 shows the quantitative results of our ablation experiments. We note from the results on the complete evaluation set that the probabilistic trajectory prediction models outperform the baseline of the IMM. The M-VGMM has lower values for both mean as well as median absolute error as compared to the IMM suggesting that the probabilistic model makes fewer as well as less drastic errors on an average. We get further improvements in mean and median absolute deviations using the C-VGMMs suggesting that subcategorizing trajectories into maneuver classes leads to a better probabilistic prediction model.

This is further highlighted based on the prediction results for the challenging maneuver classes of overtakes and cut-ins. We note that the C-VGMM significantly outperforms the CV and M-VGMM models both in terms of mean and median absolute deviation for overtakes and cut-ins. This trend becomes more pronounced as the prediction horizon is increased. This suggests that the motion model is more error prone due to the non-uniform motion in overtakes and cut-ins while these rare classes get underrepresented in the distribution learned by the monolithic M-VGMM. Both of these issues get addressed through the C-VGMM. We analyze this further by considering specific cases of predictions made by the IMM, M-VGMM and C-VGMM in Section 5.7.5

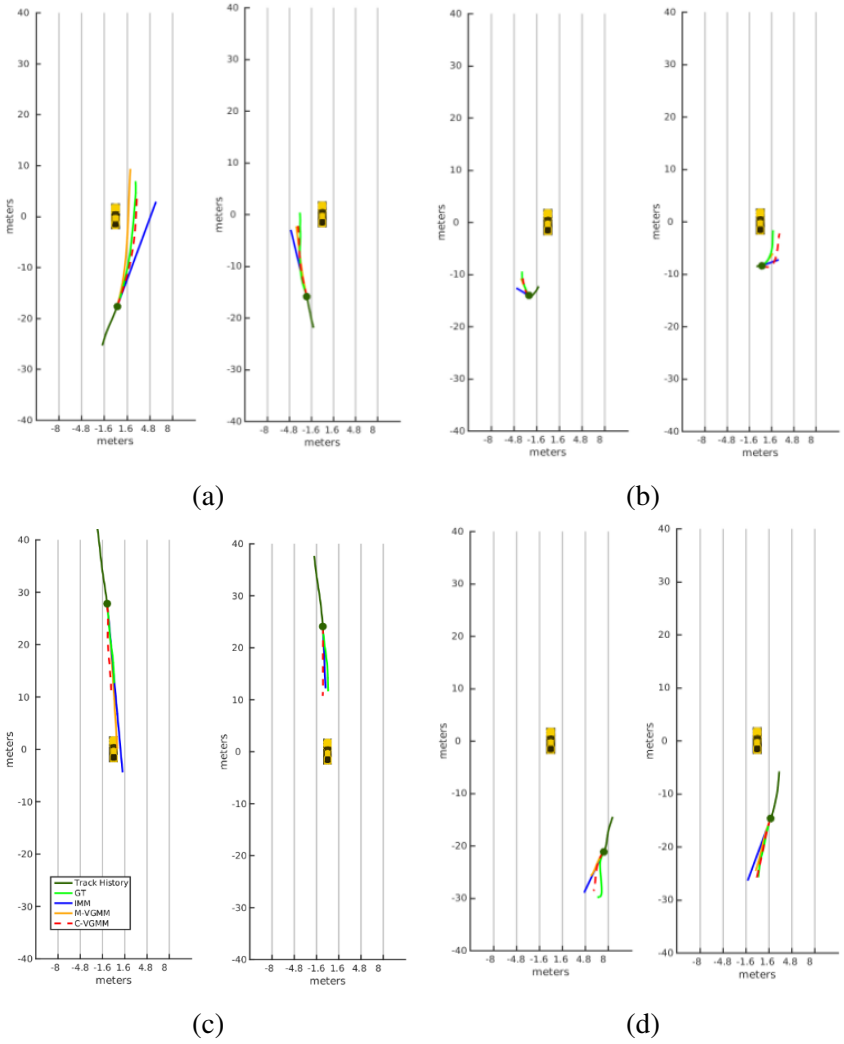
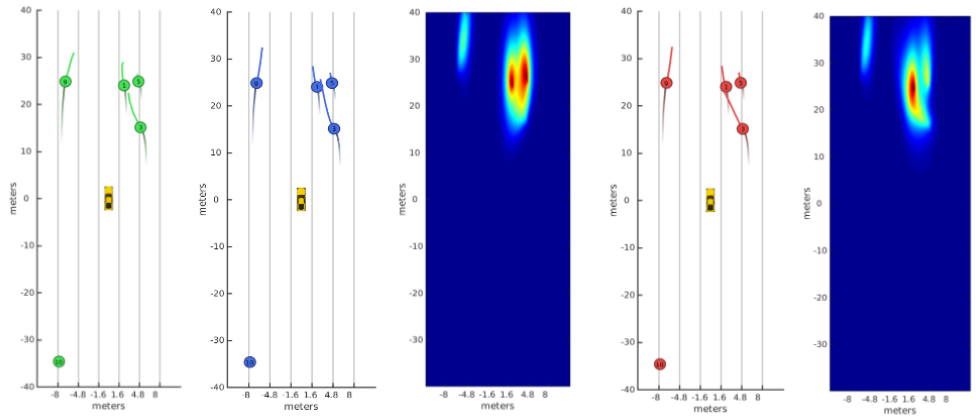
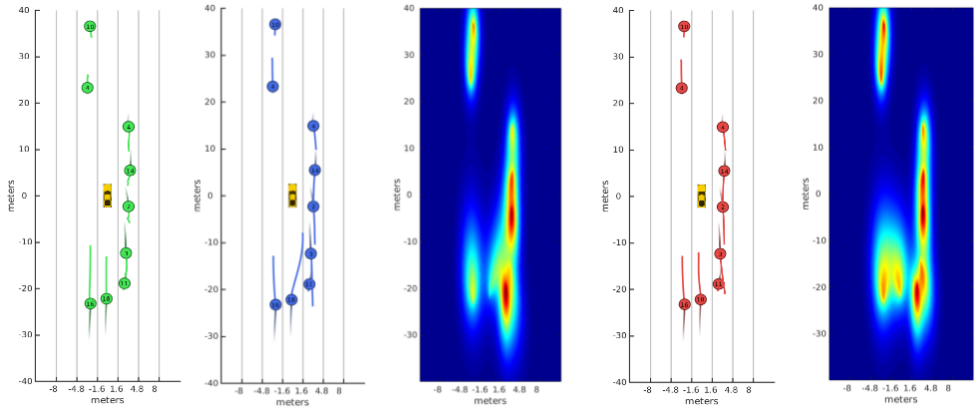


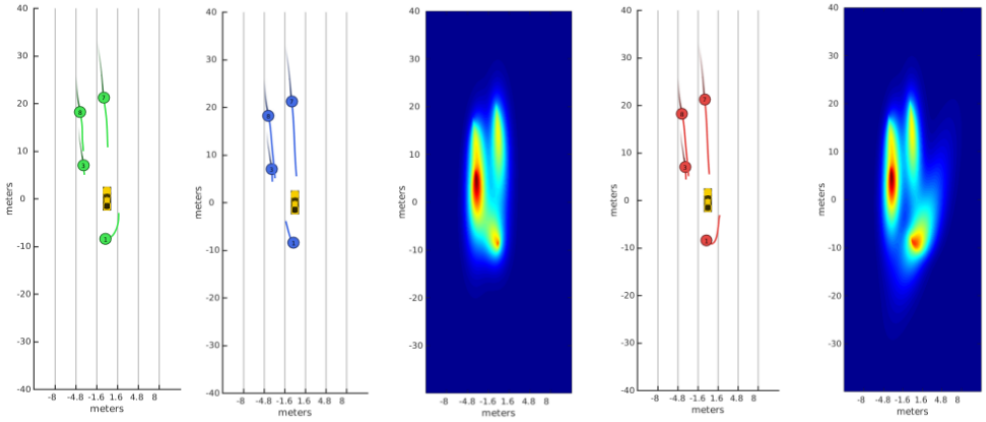
Figure 5.5: **Analysis of predictions made by CV, M-VGMM and C-VGMM models:** (a): Better prediction of lateral motion in overtakes by the probabilistic models. (b): Early detection of overtakes by the HMM. (c): Deceleration near the ego vehicle predicted by the C-VGMM. (d): Effect of lane information implicitly encoded by the M-VGMM and C-VGMM



(a) Infeasible lane pass is correctly changed to cut-in



(b) Infeasible overtake correctly changed to tail-gating



(c) Infeasible left overtake changed to the correct overtake direction

Figure 5.6: **Case Studies analyzing the effect of the VIM:** Each case shows from left to right: The ground truth, predictions made independently for each vehicle, uncertainty of the independent predictions, predictions made with the VIM, uncertainties of the VIM predictions

Comparing the maneuver classification accuracies for the case of C-VGMM and C-VGMM + VIM, we note that the VIM corrects some of the maneuvers assigned by the HMM. This in turn leads to improved trajectory prediction as seen from the mean and median absolute error values. We note that this effect is more pronounced in case of stop-and-go traffic, since the dense traffic conditions cause more vehicles to affect each others motion leading to a greater proportion of maneuver class labels to be re-assigned by the VIM. Section 5.7.6 analyses cases where the VIM reassigns maneuver labels assigned by the HMM due to the relative configuration of all vehicles in the scene.

5.7.4 Analysis of execution time

Table 5.2 also shows the average execution time per frame for the 4 system configurations considered. As expected, the IMM baseline has the lowest execution time since all other configurations build upon it. We note that the C-VGMM runs faster than the M-VGMM in spite of having the overhead of the HMM based maneuver recognition module. This is because the M-VGMM is a much bulkier model as compared to any single maneuver C-VGMM. Thus in spite of involving an extra step, the maneuver recognition module allows us to choose a much leaner model, effectively reducing the execution time while improving performance. The VIM is a more time intensive overhead and almost doubles the run time of the C-VGMM. However, even in it's most complex setting, the proposed framework can be deployed at a frame rate of almost 6 fps, which is more than sufficient for the application being considered.

5.7.5 Analyzing predictions of IMM, M-VGMM and C-VGMM models

Figure 5.5 shows the trajectories predicted by the CV, M-VGMM and C-VGMM models for 8 different instances.

Figure 5.5a shows two prediction instants where the vehicle is just about to start the

non-linear part of overtake maneuvers. We observe that the IMM makes erroneous predictions in both cases. However, both the M-VGMM and C-VGMM manage to predict the non-linear future trajectory.

Figure 5.5b shows two prediction instants in the early part of overtake maneuvers. We note that both the IMM and M-VGMM make errors in prediction. However the position of the surround vehicle along with the slight lateral motion provide enough context to the maneuver recognition module to detect the overtake maneuver early. Thus, the C-VGMM manages to predict that the surround vehicle would move to the adjacent lane and accelerate in the longitudinal direction, although there is no such cue from the vehicles existing state of motion

Figure 5.5c shows two instants the trajectory of a vehicle that decelerates as it approaches the ego vehicle from the front. This trajectory corresponds to the drift into ego-lane maneuver class. In the first case (left), the vehicle has not started decelerating, causing the IMM to assign a high probability to the CV model. The IMM thus predicts the vehicle to keep moving at a constant velocity and come dangerously close to the ego vehicle. Similarly, the M-VGMM makes a poor prediction since these maneuvers are underrepresented in the training data. The C-VGMM however manages to correctly predict the surround vehicle to decelerate. In the second case (right), we observe that the car has already started decelerating. This allows the IMM to assign a greater weight to the CA model and correct its prediction

Finally Figure 5.5d shows two interesting instances of the lane pass right back maneuver that is well represented in the training data. The vehicle makes a lane change in both of these instances. We note, as expected, that the IMM poorly predicts these trajectories. However both the M-VGMM and C-VGMM correctly predict the vehicle to merge into the lane, suggesting that the probabilistic models may have implicitly encoded lane information.

5.7.6 Vehicle Interaction Model Case Studies

Figure 5.6 shows three cases where the recognized maneuvers and predicted trajectories are affected by the VIM. In each case, the green plots show the ground truth of future tracks, the blue plots show the predictions made for each vehicle independently and the red plots show the predictions based on the VIM. Additionally we plot the prediction uncertainties for either case.

Consider the first case in Figure 5.6a, in particular vehicle 3. We note from the blue plot that the HMM predicts the vehicle to perform a lane pass. However the the vehicle's path forward is blocked by vehicles 1 and 5. The VIM thus infers vehicle 3 to perform a cut-in in with respect to the ego-vehicle in order to overtake vehicle 5.

In Figure 5.6b, the HMM predicts vehicle 18 to overtake the ego-vehicle from the right. However, we can see that the right lane is occupied by vehicles 11, 3 and 2. These vehicles yield high values of pairwise energies with vehicle 18 for the overtake maneuver. The VIM thus correctly manages to predict that vehicle 18 would end up tail-gating by assigning it the maneuver drift into ego lane (rear).

Finally Figure 5.6c shows a very interesting case where the HMM predicts vehicle 1 to overtake the ego vehicle from the left. Again, the left lane is occupied by other vehicles making the overtake impossible to execute from the left. However, compared to the previous case, these vehicles are slightly further away and can be expected to yield relatively smaller energy terms as compared to case (b). However, these terms are enough to offset the very slight difference in the HMM's confidence values between the left and right overtake since both maneuvers do seem plausible if we consider vehicle 1 independently. Thus the VIM reassigns the maneuver for vehicle 1 to a right overtake, making the prediction closely match the ground truth.

5.8 Chapter Summary

In this chapter, we have presented a unified framework for surround vehicle maneuver recognition and motion prediction using vehicle mounted perceptual sensors, that leverages the instantaneous motion of vehicles, an understanding of motion patterns of freeway traffic and the effect of inter-vehicle interactions. The proposed framework outperforms an interacting multiple model based trajectory prediction baseline and runs in real time at about 6 frames per second.

An ablative analysis for the relative importance of each cue for trajectory prediction has been presented. In particular, we have shown that probabilistic modeling of surround vehicle trajectories is a more versatile approach, and leads to better predictions as compared to a purely motion model based approach for many safety critical trajectories around the ego vehicle. Additionally, sub-categorizing trajectories based on maneuver classes leads to better modeling of motion patterns. Finally, incorporating a model that takes into account interactions between surround vehicles for simultaneously predicting each of their motion leads to better prediction as compared to predicting each vehicle's motion independently.

The proposed approach could be treated as a general framework, where improvements could be made to each of the three interacting modules.

5.9 Acknowledgements

Chapter 5, in full, is a reprint of the material as it appears in IEEE Transactions on Intelligent Vehicles (2018), by Akshay Rangesh, Nachiket Deo, and Mohan M. Trivedi. The dissertation author was one of the primary investigators and authors of this paper.

Part III

Looking-In

Chapter 6

Gaze Preserving CycleGANs for Eyeglass Removal & Persistent Gaze Estimation

A driver's gaze is critical for determining their attention, state, situational awareness, and readiness to take over control from partially automated vehicles. Estimating the gaze direction is the most obvious way to gauge a driver's state under ideal conditions when limited to using non-intrusive imaging sensors. Unfortunately, the vehicular environment introduces a variety of challenges that are usually unaccounted for - harsh illumination, nighttime conditions, and reflective eyeglasses. Relying on head pose alone under such conditions can prove to be unreliable and erroneous. In this chapter, we offer solutions to address these problems encountered in the real world. To solve issues with lighting, we demonstrate that using an infrared camera with suitable equalization and normalization suffices. To handle eyeglasses and their corresponding artifacts, we adopt image-to-image translation using generative adversarial networks to pre-process images prior to gaze estimation. Our proposed Gaze Preserving CycleGAN (GPCycleGAN) is trained to preserve the driver's gaze while removing potential eyeglasses from face images. GPCycleGAN is based on the well-known CycleGAN approach - with the addition of a gaze classifier and a gaze consistency loss for additional supervision. Our approach exhibits improved performance, interpretability, robustness and superior qualitative results on challenging real-world datasets.

Code, datasets & models: https://github.com/arangesh/GPCycleGAN

6.1 Introduction

Driver safety and accident risk are highly determined by a driver's attention levels, especially visual attention, e.g. a driver's gaze, for both conventional and self-driving vehicles [4, 46, 151–153]. It is crucial to have the ability to monitor the driver's gaze in real-time and use gaze information to efficiently enhance vehicle safety for intelligent vehicles. Previous work has proven that visual information from imaging sensors and corresponding learning algorithms are effective in determining driver gaze zones [154–157] and driver activity in general [158–161]. However, the state-of-the-art for gaze estimation does not account for the complexities of the real world (for example - drivers wearing eyeglasses, harsh illumination, nighttime conditions, etc.). According to the Vision Impact Institute [162, 163], one in five drivers have a vision problem and most of them wear correction eyeglasses while driving. Additionally, according to the National Safety Council, 50% of all accidents happen while driving in the dark [164]. However, it is difficult to estimate the gaze with a high accuracy using RGB images acquired from dark environments. It is primarily because RGB cameras suffer from a lack of photons captured by imaging sensors, resulting in low signal-to-noise ratios [165]. In this study, we find that using infrared cameras can mitigate these lighting issues; however, for face images with eye-wear, data pre-processing or algorithmic improvements are necessary to identify the driver's gaze zones. Since previous works on gaze estimation [156] achieved decent results (under ideal conditions) by using convolutional neural networks (CNNs), we adopt their basic methodology and develop novel pre-processing approaches that can improve gaze estimation in these demanding conditions.

Generative Adversarial Networks (GANs) [166] have shown promising results on a variety of computer vision tasks like image super-resolution [167], realistic face images generation [168], style transfer, image editing [169, 170], etc. There are two sub-models in GAN architecture, a generator and a discriminator. The generator updates itself to synthesize images that are indistinguishable from real images by learning the data distribution. The discriminator's task is

Figure 6.1: The real world introduces variability and complexity that driver gaze estimation systems usually ignore. Some examples include - use of eyeglasses, harsh illumination, nighttime data etc.

to differentiate between real and synthesized images. Both of them are trained together in an adversarial manner until an equilibrium is attained. More recent work on GANs has focused on addressing key issues like training on large datasets, training larger models, improving stability during training, preserving finer details, etc. In this study, we build on one such model called the CycleGAN [170], which achieved the state-of-the-art on image-to-image translation by using unpaired images from the source and target distributions.

Inspired by the CycleGAN architecture, we propose the Gaze Preserving CycleGAN (GPCycleGAN) which makes use of an additional gaze consistency loss. GPCycleGAN has the following advantages compared to other gaze estimation methods and glass removal techniques: First, GPCycleGAN preserves the gaze and allows for more accurate gaze estimation on images with eyeglasses. Second, unlike previous works [174, 175] on eyeglass removal, there is no need for paired images to train GPCycleGAN. Third, it works in different environments, lighting conditions, eyeglass types, and with significant variations of the head pose.

The four main contributions of this study are: a) An in-depth analysis of traditional gaze estimation under different conditions (e.g., with and without eyeglasses, daytime, nighttime, etc.), and its shortcomings, b) The GPCycleGAN model for eyeglass removal specifically optimized for the gaze classification task, c) Experimental analyses to illustrate how the GPCycleGAN model improves the accuracy, interpretability and robustness over a variety of baseline models, and d) A naturalistic driving dataset with labeled gaze zones that includes both IR and RGB images.

Table 6.1: Related research

(a) Selected research on gaze estimation in vehicular environments

Study	Objective	Sensor	Features	Capture conditions	Methodology
Vora et al. [156]	Gaze zone classification using CNNs	1 RGB camera	HP[1]& gaze	Daytime; w/o eyeglasses	CNN
Martin et al. [155]	Estimating gaze dynamics, glance duration and frequency	1 RGB camera	HP & gaze	Daytime; w/o eyeglasses	CNN & geometry
Naqvi et al. [171]	Gaze zone detection using NIR[2] camera and deep learning	1 NIR camera & NIR LEDs	HP & gaze	Daytime & nighttime; w & w/o eyeglasses	CNN
Yong et al. [157]	Gaze detection using dual NIR cameras and deep residual networks	2 NIR cameras & NIR LEDs	HP & gaze	Daytime & nighttime; w/ & w/o eyeglasses	CNN
Jha & Busso [154]	Gaze region estimation using dense pixelwise predictions	1 RGB camera & 1 headband with AprilTags	HP	Daytime; w/ & w/o eyeglasses	Dense Neural Networks
Wang et al. [172]	Continuous gaze estimation using RGB-D camera	1 RGB-D camera	HP & gaze	Daytime; w/ & w/o eyeglasses	Feature extraction & k-NN

(b) Selected research on eyeglass removal and related topics

Study	Objective	Methodology	Dataset	Advantages	Disadvantages
Zhu et al. [170]	Unpaired image-to-image translation	Cycle-GAN[3]	Unpaired images	Uses unpaired images; performs well on style transfer	Not realistic for eyeglass removal; does not preserve gaze direction
Hu et al. [173]	Eyeglass removal	ER-GAN	Unpaired CelebA & LFW	Good performance on eyeglass removal for frontal faces	Gaze is not preserved; only works for aligned frontal images
Amodio et al. [173]	Unpaired image-to-image translation	TraVeLGAN	Unpaired Images from multiple domains	Good performance on style transfer and eyeglass removal	Eyes are often swapped; gaze is not preserved
Wang et al. [174]	Facial obstruction removal	EC-GAN & LS-GAN	Paired CelebA	Better results than Cycle-GAN; improved face recognition accuracy	Needs an obstruction classifier
Liang et al. [175]	Learn mappings between faces with and without glasses	CNN	Images from web & surveillance cameras	Single step, end-to-end method	Images need to be aligned; needs paired images; uses synthesized eyeglasses
Li et al. [176]	Identity-aware transference of facial attributes	DIAT-GAN	Aligned CelebA	Flexibility to modify different facial attributes	Generated gaze is different; blurry outputs
Shen et al. [177]	Facial attribute manipulation	GAN	CelebA & LFW	Capability to manipulate images with modest pixel modifications	Does not completely remove eyeglasses; the eye region is not preserved

[1] Head Pose [2] Near Infra-Red [3] Generative Adversarial Networks

This work is an extension of our research presented in [178]. In this study, we extend and refine our previous approach with new losses and training schemes, update our previously reported quantitative numbers, and provide new qualitative analysis.

6.2 Related Research

Recent related studies are listed in Table 6.1 with 2 sub-tables that represent two research areas - gaze estimation and eyeglass removal. We mainly focus on vision-based approaches post 2016 for gaze estimation. Please refer to the following studies for earlier work: Vora et al. [156] for gaze estimation; Kar et al. [179] for eye-gaze estimation systems, algorithms, and performance evaluation methods in consumer platforms; a survey on driver behavior analysis for safe driving by Kaplan et al. [180]; a review on driver inattention monitoring systems by Dong et al. [181];

and a survey on head pose estimation by Murphy et al. [182]. For the second research area, we were unable to find published studies focused on eyeglass removal for drivers' face images. We instead provide discussions on general-purpose eyeglass removal studies.

6.2.1 Gaze Estimation

Gaze estimation studies can be categorized in the following ways. First, by the type of model used, they can be divided into convolutional neural networks [156], statistical learning models [183], or geometric approaches [155]. Second, methods can be distinguished by the cues they consider, i.e. studies that only use head pose [154] versus ones that use both head pose and eye information [156, 157, 184, 185]. Third, research can be categorized by the conditions they capture. For instance, studies with limited illumination changes [156] versus research with multiple environments and the variations that come with it [157]. Lastly, studies can be separated by the sensors they use, including RGB cameras [156], IR cameras [186], NIR cameras [171], multiple cameras [187], and wearable sensors [188]. However, only a few studies emphasize the problems associated with driving with eyeglasses or eye-wear. Tawari et al. [187] and Lee et al. [189] mention the unreliability of gaze estimation for drivers with glasses, and propose methods that only rely on head pose as fallback solutions. Naqvi et al. [171] combine head pose estimation and pupil detection to determine the eye gaze, but their method only works under ideal capture conditions. Jah and Busso [154] use only head pose in their method. Wang et al. [172] combine depth images of the head and RGB images of eyes, but gaze estimation for eye images is unstable and only works with frontal images under ideal conditions. In [157], Yoon et al. collect a dataset comprising of images in daytime/nighttime, images with and without eyeglasses using two NIR cameras and NIR lights. Although they achieve good performance, they do not explicitly model the presence of eyeglasses in images. In this study, we show that such approaches tend not to generalize to different settings and usually overfit the training set.

6.2.2 Eyeglass Removal

In real-world cabin environment, large variations, including head pose, eyeglass type, environment conditions, and the presence of reflection artifacts, make eyeglass removal is a complex task. Existing eyeglass removal studies make use of statistical learning, principal component analysis (PCA), or deep learning, including GAN based methods. Statistical learning and PCA were the primary approaches prior to the advent of deep learning. These approaches do not require high computational power but have limitations on eyeglasses, environmental conditions, and head poses [190, 191]. On the other hand, methods using deep learning, such as ones proposed by Liang et al. [175], Wang et al. [174], Din et al. [192], and Lee et al. [193] modified GANs for better results, but they need paired and aligned images for training. Such datasets are expensive and tedious to collect. Different GAN architectures proposed in [170], [173], and [194] do not require paired images, but their models fail to keep gaze information and do not perform well on non-frontal images. Models developed by Li et al. [176] and Shen et al. [177] are capable of changing multiple facial attributes but do not produce satisfying results on eyeglass removal. All the above methods on eyeglass removal have general or different purposes. However, none of these methods are constructed to preserve gaze of face images, and hence cannot necessarily be used out-of-the-box for gaze estimation.

6.3 Dataset

Since our primary goal is to design a gaze estimation system for the real-world, we prioritized using small form-factor infrared cameras with suitable real-time performance. We decided on an Intel RealSense IR camera and enclosed it in a custom 3D printed enclosure mounted next to the rearview mirror. To ensure a good compromise between larger fields-of-view and faster processing speeds, we settled on a capture resolution of 640×480. Similar to Vora et al. [156], we divide the driver's gaze into seven gaze zones: *Eyes Closed/Lap*, *Forward*, *Left*

Table 6.2: Dataset size (# of images, # of subjects) across different splits and capture conditions. There is no overlap of subjects between different splits to ensure cross-subject validation and testing.

Dataset split / Capture conditions	Training	Validation	Testing
daytime; w/o eyeglasses	(67151, 9)	(9908, 1)	(2758, 4)
nighttime; w/o eyeglasses	(59352, 9)	(8510, 1)	(2768, 4)
daytime; w/ eyeglasses	(43432, 5)	(9062, 1)	(3294, 4)
nighttime; w/ eyeglasses	(33189, 5)	(8103, 1)	(2897, 4)
Total (all conditions)	(203124, 9)	(35583, 1)	(11717, 4)

Figure 6.2: Example images from our dataset under different capture conditions.

Mirror, *Speedometer*, *Radio*, *Rearview*, *Right Mirror*. Unlike previous studies, we also include gaze zones related to driver inattention or unsafe driving behavior. Our entire dataset comprises of 13 subjects in different lighting conditions (daytime, nighttime and harsh lighting), wearing a variety of eyeglasses. For every gaze zone, participants were instructed to keep their gaze fixed while moving their heads within reasonable limits. Of the 13 subjects, 7 were male, with an ages ranging from 20 to 65 years. In total, 336177 frames of images were captured, which we split into training, validation, and test sets with no overlap of subjects. Table 6.2 shows the distribution of images and subjects across different splits and capture conditions. Figure 6.2 depicts exemplar images from our dataset. We ensure that the dataset is suitably diverse and challenging to represent the complexities observed in the real world.

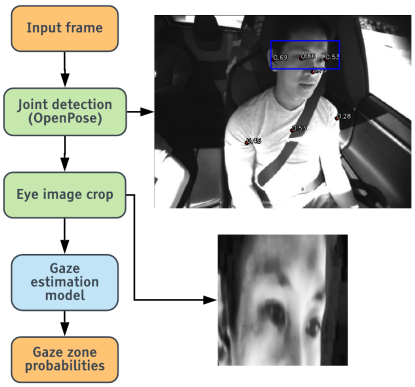


Figure 6.3: Overall processing pipeline for driver gaze zone estimation.

Table 6.3: Validation accuracies for gaze models trained on data with different capture conditions.

Model ⓜ & training data used \ Validation data	ⓐ daytime; w/o eyeglasses	ⓑ nighttime; w/o eyeglasses	ⓒ daytime; w/ eyeglasses	ⓓ nighttime; w/ eyeglasses	ⓔ w/o eyeglasses	ⓕ w/ eyeglasses	ⓖ daytime	ⓗ nighttime	ⓘ all conditions
① daytime; w/o eyeglasses	81.0774	62.4657	45.4588	11.0680	72.5677	29.7526	64.1875	38.1209	52.2757
② nighttime; w/o eyeglasses	73.5272	87.2226	29.3782	25.6684	79.7891	27.6839	52.5923	58.0671	55.0941
③ daytime; w/ eyeglasses	60.4746	42.7325	70.6334	39.7423	52.3625	56.5256	65.2918	41.3162	54.3356
④ nighttime; w/ eyeglasses	51.4649	76.7639	45.3773	64.8151	63.0322	54.2544	48.5782	71.1043	58.8720
⑤ w/o eyeglasses	85.5508	88.7809	43.9101	20.6954	87.0276	33.3080	65.8053	56.5317	61.5675
⑥ w/ eyeglasses	71.3746	82.2239	69.5738	77.7254	76.3351	73.2966	70.5207	80.0932	74.8951
⑦ daytime	77.2131	74.7569	73.5095	39.9224	76.0901	58.1704	75.4569	58.2573	67.5971
⑧ nighttime	55.1192	83.7696	49.3596	61.3520	68.2189	54.8365	52.3881	73.1514	61.8763
⑨ all conditions	81.6969	83.3084	56.7885	66.4358	82.4337	61.1944	69.8857	75.3166	72.3675

6.4 Methodology

As depicted in Figure 6.3, the steps involved in our proposed gaze estimation pipeline are as follows: (a) landmark detection using OpenPose [195], (b) eye image cropping, resizing, and equalization, (c) gaze estimation. We use OpenPose for landmark detection because of its high accuracy under different conditions and fast inference speed. Based on the work by Vora et al. [156], we crop the eye region using the estimated landmarks as per their conclusion that the upper half of the face as an input produced the best results for downstream gaze classification. Next, we use adaptive histogram equalization to improve the contrast and resize the images to 256×256 before feeding them to the gaze estimation models.

6.4.1 Issues with Lighting & Eyeglasses

To understand the impacts of different conditions on gaze estimation, we carry out an extensive experiment to analyze the performance of models trained with subsets of data, when they are tested on data from within and outside the training distribution. Table 6.3 shows validation accuracies of SqueezeNet-based gaze classifier models ① - ⑨ (as proposed in [156]), each trained on data captured under different conditions ⓐ - ⓘ. From the Table, we glean that model ⑤ validated on data ⓐ, ⓑ, and ⓔ produce similar accuracies, and model ⑨ validated on data ⓖ, ⓗ, and ⓘ also perform similarly. This demonstrates that the gaze classifier models work well on daytime and nighttime data when trained on data containing both conditions. Thus, problems related to lighting can be effectively solved by training using IR images, appropriate normalization, and histogram equalization. We also learn that nighttime data is easier to model and results in better accuracies in general. Next, to analyze the models' performance on data with and without glasses, we observe that the accuracy of model ⑤ validated on data ⓔ (87.0276%) is much higher than the accuracy of model ⑥ validated on data ⓕ (73.2966%). Similarly, the accuracy of model ① is better than model ③, and model ② is better than model ④ when their validation sets comprise of data from their training distribution. The model trained with all the data (⑨) always has better accuracies when validated on data without glasses (ⓐ, ⓑ, and ⓔ), than that with glasses (ⓒ, ⓓ, and ⓕ). This implies that simply training on data with and without eyeglasses is not sufficient to ensure good generalization across both conditions. Thus, we need to explicitly handle eyeglasses through modeling, or reduce the *domain gap* between images with and without eyeglasses. In this study, we propose to do the latter by training an eyeglass removal network.

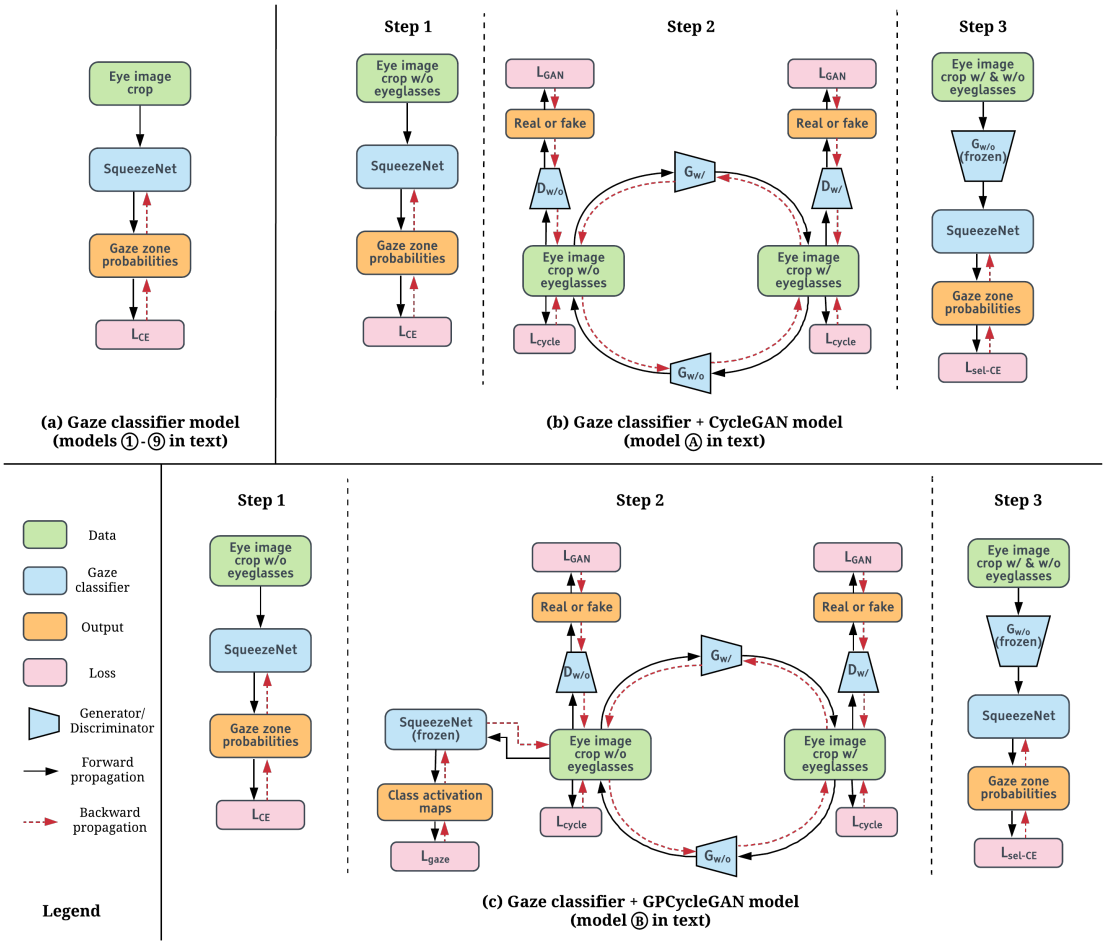


Figure 6.4: Training setup and architectures for different gaze zone estimation models.

6.4.2 Vanilla CycleGAN-based Approach

The two domains for the eyeglass removal problem are eye image crops without eyeglasses, $\mathcal{X}$, and eye image crops with eyeglasses, $\mathcal{Y}$ ($\mathcal{X}, \mathcal{Y} \subset \mathbb{R}^{H \times W \times C}$, $H = 256$, $W = 256$, and $C = 1$ for IR images, $C = 3$ for RGB images). We denote X ($X \in \mathcal{X}$) and Y ($Y \in \mathcal{Y}$) as sample images from domains $\mathcal{X}$ and $\mathcal{Y}$ respectively.

First, we consider a baseline gaze classifier model (Step 1 in Figure 6.4(b)), for which we use input images from domain $\mathcal{X}$. We use the SqueezeNet architecture as before and denote its output gaze zone probabilities as $\{p_i\}_i$. Standard cross-entropy loss ($\mathcal{L}_{CE}$) is used for training the

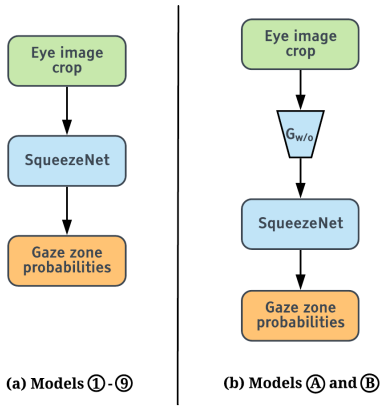


Figure 6.5: Inference setup and architectures for different gaze zone estimation models.

gaze classifier. $\mathcal{L}_{CE}$ is defined as:

$$\mathcal{L}_{CE} = -\sum_{i} z_i \log p_i, \tag{6.1}$$

where i is the class index and z is the one-hot encoding of the ground truth class.

Next, we consider a CycleGAN-based eyeglass removal network, followed by a SqueezeNet-based gaze classifier (Step 2 in Figure 6.4(b)). The CycleGAN-based eyeglass removal model learns the mapping $G_{w/o}$ from $\mathcal{Y}$ to $\mathcal{X}$ (removing eyeglasses), denoted as $G_{w/o}$: $\mathcal{Y} \rightarrow \mathcal{X}$. When trained, $G_{w/o}$ generates images $G_{w/o}(\cdot)$ using the input from either domain, and then passes them to the pre-trained gaze classifier to generate output probabilities $\{p_i\}_i$. In a similar manner, the mapping for adding eyeglasses to $X \in \mathcal{X}$ is $G_{w/} : \mathcal{X} \rightarrow \mathcal{Y}$, and it is trained simultaneously with $G_{w/o}$. The discriminator $D_{w/}$ aims to distinguish between real images Y and generated images $G_{w/}(X)$, whereas the discriminator $D_{w/o}$ aims to do so between real images X and generated images $G_{w/o}(Y)$. The losses used in this model are: cycle consistency losses [170] (Equation 6.2) that encourages cycle consistency between the real images and the reconstructed images; adversarial losses [166] (Equation 6.3) for making the generated images indistinguishable from real images; and identity losses [196] (Equation 6.4) to ensure identity mapping when the input belongs to the target domain.

$$\begin{aligned}\mathcal{L}_{\text{cyc}}(G_{w/}, G_{w/o}) = \mathbb{E}_{X\sim P(X)}\left[\|G_{w/o}(G_{w/}(X)) - X\|_1\right] \\ + \mathbb{E}_{Y\sim P(Y)}\left[\|G_{w/}(G_{w/o}(Y)) - Y\|_1\right],\end{aligned} \tag{6.2}$$

$$\begin{aligned}&\mathcal{L}_{\text{adv}}(G_{w/}, G_{w/o}, D_{w/}, D_{w/o}) \\ &\quad = \mathbb{E}_{Y\sim P(Y)}\left[\log D_{w/}(Y)\right] \\ &\quad + \mathbb{E}_{X\sim P(X)}\left[\log\left(1 - D_{w/}\left(G_{w/}(X)\right)\right)\right] \\ &\quad + \mathbb{E}_{X\sim P(X)}\left[\log D_{w/o}(X)\right] \\ &\quad + \mathbb{E}_{Y\sim P(Y)}\left[\log\left(1 - D_{w/o}\left(G_{w/o}(Y)\right)\right)\right],\end{aligned} \tag{6.3}$$

and,

$$\begin{aligned}\mathcal{L}_{\text{identity}}(G_{w/}, G_{w/o}) = \mathbb{E}_{Y\sim P(Y)}\left[\|G_{w/}(Y) - Y\|_1\right] \\ + \mathbb{E}_{X\sim P(X)}\left[\|G_{w/o}(X) - X\|_1\right],\end{aligned} \tag{6.4}$$

where $X \sim P(X)$ and $Y \sim P(Y)$ are the image distributions.

The full objective for the CycleGAN model is the following:

$$\mathcal{L}_{total} = \mathcal{L}_{\text{adv}} + \lambda_1 \mathcal{L}_{\text{cyc}} + \lambda_2 \mathcal{L}_{\text{identity}}, \tag{6.5}$$

where λ_1 is the weight of the cycle loss, and λ_2 is the weight of the identity loss.

As a final refinement step, we retrain the gaze classifier from step 1 using real images from $\mathcal{X}$ and fake images from $G_{w/o}(\mathcal{Y})$ (Step 3 in Figure 6.4(b)). Unlike step 1, we use a *selective* cross-entropy loss (L_{sel-CE}) to avoid overfitting to errors resulting from $G_{w/o}(\cdot)$. $\mathcal{L}_{sel-CE}$ is defined as:

$$\mathcal{L}_{sel-CE} = \begin{cases} -\sum_i z_i \log p_i, & \text{if } \arg\max_i p_i = \arg\max_i z_i \\ 0, & \text{otherwise,} \end{cases} \tag{6.6}$$

where only the cross-entropy loss from correctly classified samples are backpropagated. This ensures that the gaze classifier is not trained to incorrectly classify fake images with a potentially

different gaze zone than the original. After the fine-tuning step, the generator $G_{w/o}$ and the gaze classifier are deployed sequentially to estimate gaze zones in the presence of eyeglasses (Figure 6.5(b)).

6.4.3 Proposed GPCycleGAN-based Approach

Our proposed GPCycleGAN model with the gaze classifier (Figure 6.4(c)) builds on the previous model, with the addition of a gaze consistency loss ($\mathcal{L}_{gaze}$) in step 2. To preserve the gaze features during eyeglass removal, we use the trained gaze classifier from step 1 and compute the gaze consistency loss using its Class Activation Maps (CAMs). The proposed gaze consistency loss is defined as follows:

$$\mathcal{L}_{\text{gaze}}(G_{w/}, G_{w/o}) = \frac{1}{N}\sum_{i=1}^{N} ||T(A^i_{w/o;real}) - T(A^i_{w/o;rec})||_2, \tag{6.7}$$

where N stands for the total number of classes (gaze zones), i is the class index, and $\{A^i\}_{i=1}^N$ are the CAMs from the trained gaze classifier. Subscript $w/o;real$ denotes CAMs obtained from a real eye image crop without eyeglasses, and subscript $w/o;rec$ denotes CAMs corresponding to reconstructed images $G_{w/o}(G_{w/}(\cdot))$ using the same eye image crop. The transformation $T(\cdot)$ applied to the CAMs is a simple sigmoid function with temperature τ:

$$T(A^i) = \frac{1}{1 + exp(-\tau \cdot A^i)}. \tag{6.8}$$

The nature of the proposed gaze loss necessitates the use of a cyclic structure, as we do not have perfectly paired samples of images with and without glasses to enforce gaze consistency directly.

The full objective for the proposed GPCycleGAN model is:

$$\mathcal{L}_{total} = \mathcal{L}_{\text{adv}} + \lambda_1 \mathcal{L}_{\text{cyc}} + \lambda_2 \mathcal{L}_{\text{identity}} + \lambda_3 \mathcal{L}_{\text{gaze}}, \tag{6.9}$$

where λ_3 is the weight for the gaze consistency loss.

6.4.4 Implementation Details

The training and inference architectures are the same for the baseline gaze classification models ①-⑨, but different for models Ⓐ and Ⓑ. Models ①-⑨ are adopted from Vora et al. [156], and differ in the data they were trained on (see Table 6.3). Models Ⓐ and Ⓑ are identical in terms of their architectures, and only differ in their training setup and losses (i.e. the addition of a gaze consistency loss in model Ⓑ). Both models consist of generators with 9 residual blocks and 70×70 PatchGANs [170] as discriminators. As illustrated in Figure 6.4, training models Ⓐ and Ⓑ proceeds in 3 steps: Step 1 involves training a SqueezeNet gaze classifier using eye image crops from domain X; Step 2 is for training the generator $G_{w/o}$ in CycleGAN/GPCycleGAN; Step 3 is to fine-tune the gaze classifier from step 1 using generated images $G_{w/o}(x,y)$. The inference for models ① - ⑨ (Figure 6.5(a)) is a simple forward propagation through the gaze classifier to obtain the gaze zone probabilities. Models Ⓐ and Ⓑ have the same inference setup (Figure 6.5(b)), where eye image crops are first passed to the generator $G_{w/o}$ for eyeglass removal, after which they are fed to the gaze classifier which outputs the gaze zone probabilities.

We choose $\tau = 0.01$ in Equation 6.8, and $\lambda_1 = 10$, $\lambda_2 = 5$, $\lambda_3 = 1$ in Equation 6.5 and Equation 6.9. We use a learning rate of 0.0004 with an Adam optimizer for training all SqueezeNet classifiers, a learning rate of 0.0002 for CycleGAN/GPCycleGAN in step 2, and a reduced learning rate of 0.0001 for the final finetuning step. The gaze classifiers are trained for a total of 50 epochs, while the GANs are trained for 15 epochs - both with early stopping. The total runtime for inference using our approach is 27 ms when using an RTX 2080 graphics card (OpenPose: 22 ms, eyeglass removal network: 3 ms, and gaze classifier network: 2 ms) - indicating real-time performance.

Table 6.4: Test set metrics for different models.

Model	Micro-average accuracy(%)	Macro-average accuracy(%)	Confusion matrix
⑤	65.27	62.13	Figure 6.6a
⑨	76.77	75.27	Figure 6.6b
Ⓐ	73.76	71.31	Figure 6.6c
Ⓐ with fine-tuning	78.16	78.01	Figure 6.6d
Ⓑ	73.91	71.88	Figure 6.6e
Ⓑ with fine-tuning	**81.23**	**79.44**	Figure 6.6f

6.5 Experiments & Analyses

6.5.1 Quantitative Metrics & Results

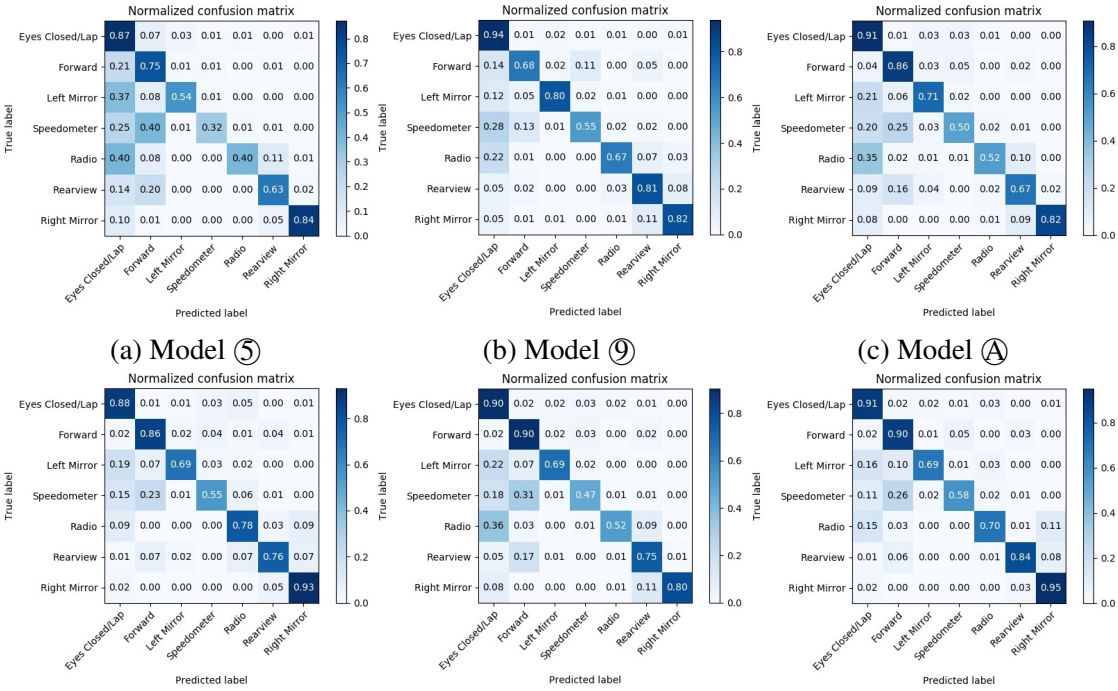
Table 6.4 presents the cross-subject test accuracies for 6 different models. We use two metrics, namely the macro-average and micro-average accuracy, defined as follows:

$$\text{Macro-average accuracy} = \frac{1}{N}\sum_{i=1}^{N}\frac{(\text{True positives})_i}{(\text{Total population})_i}, \tag{6.10}$$

$$\text{Micro-average accuracy} = \frac{\sum_{i=1}^{N}(\text{True positives})_i}{\sum_{i=1}^{N}(\text{Total population})_i}, \tag{6.11}$$

where N is the number of classes. Micro-average accuracy represents the overall percentage of correct predictions, while macro-average accuracy represents the average of all per-class accuracies. So, if the data is balanced, which means the number of images for every class is the same, micro-average accuracy will equal macro-average accuracy. In our case, there are more data in higher accuracy classes, so the micro-average accuracies are higher than macro-average accuracies. Thus, we mainly compare the macro-average accuracy since the dataset is not balanced.

First, we consider model ⑤ trained only on images without eyeglasses to illustrate the domain gap between images with and without eyeglasses. As can be seen, this model performs poorly on a test set containing images outside its training distribution. Next, model ⑨ represents the scenario where the eyeglasses are not explicitly modeled. Although it is trained on the entire



(a) Model ⑤ (b) Model ⑨ (c) Model Ⓐ

(d) Model Ⓐ with fine-tuning (e) Model Ⓑ (f) Model Ⓑ with fine-tuning

Figure 6.6: Confusion matrices on the test set for different models.

training set, the resulting accuracies indicate a performance penalty, especially when tested on images with eyeglasses.

Next, we see that adding a pre-processing network to remove eyeglasses such as in models Ⓐ and Ⓑ produces slightly worse accuracies than model ⑨. This can be attributed to the fact that the gaze classifier has only been trained on half the training set i.e. images without eyeglasses. However, after fine-tuning using the entire training set (Step 3 in Figure 6.4), the accuracies for both models Ⓐ and Ⓑ increase considerably. In conclusion, our proposed model Ⓑ demonstrates significant improvement over both the baseline model ⑨ and the vanilla CycleGAN-based model Ⓐ after fine-tuning. The above evidence implies the benefits of our proposed gaze consistency loss, and demonstrates that the generator resulting from the GPCycleGAN model acts effectively as a pre-processing step for the downstream task of gaze estimation.

In addition to the test set accuracies, we also present the corresponding confusion matrices

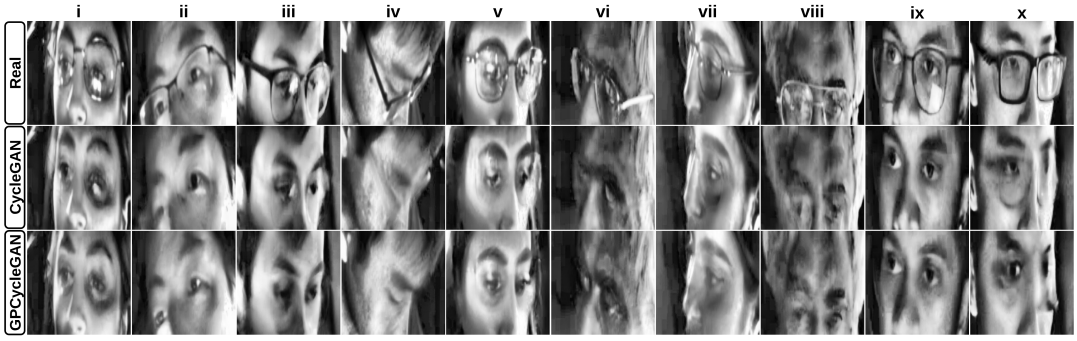


Figure 6.7: Example real infrared images with eyeglasses, and corresponding generated images after eyeglass removal.

of all 6 models in Figure 6.6. The error modes of our best performing model (Figure 6.6f) can mostly be attributed to confusion between gaze zones close in physical space (for example, *Forward* versus *Speedometer*), occlusion of the eyes by the eyeglass frame and glare, and the inability to distinguish between looking downwards versus closed eyes.

6.5.2 Qualitative Results

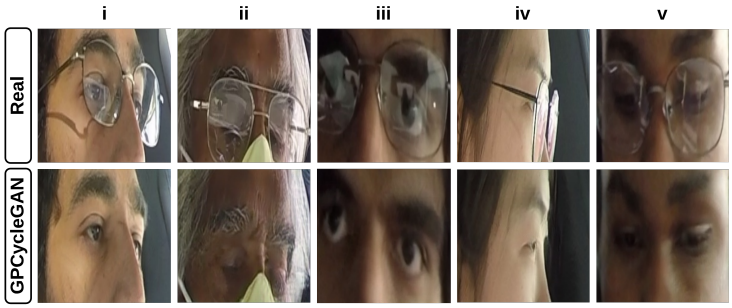


Figure 6.8: Example real RGB images with eyeglasses, and corresponding generated images after eyeglass removal.

To carry out a qualitative comparison between different GAN variants, we also show 10 examples of eyeglass removal on real images using CycleGAN and GPCycleGAN in Figure 6.7. In columns i, iii, v, vi, and viii, GPCycleGAN not only removes the eyeglasses, but also removes the

glare resulting from it; whereas CycleGAN perceives the glare as part of the sclera. Glare removal is essential for gaze estimation because the glare from glasses is a relatively common occurrence in the real world, and often occludes the eyes, making it harder for models to learn discriminative gaze features. In columns ii, iv, vi, and viii, the images generated from GPCycleGAN are realistic and preserve the gaze more accurately. Columns ii, iii, iv, vi, and vii show that our model does not only work with frontal face images but also performs well for a variety of head poses. Column ix is an example where both models perform well because the gaze is clear and not occluded by the frame or glare. The last column (x) depicts a failure case for both models. The models fail because the frame of the eyeglass is too thick, and both the frame and glare occlude the eye regions severely. These problems could potentially be solved by collecting more data with thicker eyeglass frames, increasing the image resolution, and/or by designing better GANs.

To test the viability of our approach on different data types, we repeated our prescribed training procedure using an RGB dataset collected for the same task. Figure 6.8 depicts qualitative results for eyeglass removal using our RGB model. For this RGB dataset, we notice that the resulting images after eyeglass removal tend to be sharper, more realistic, and preserve more details. We believe this is because of the higher resolution and better signal-to-noise ratio of the RGB sensor used to capture the dataset. We provide links to download both IR and RGB datasets in our repository.

6.5.3 Comparison with the Other Methods on Different Datasets

To further demonstrate the generality of our approach, we train and test it on two popular datasets - the Columbia gaze dataset [197], and the MeGlass dataset [198].

Results on the Columbia Gaze Dataset. We split the dataset such that the train and test splits contain 45 and 11 subjects respectively. Of the 21 subjects who wore prescription glasses, 6 were included in the test set and the remaining in the train set. The Columbia dataset has very few images to train a gaze classifier (5880 images for 21 gaze classes). So we instead

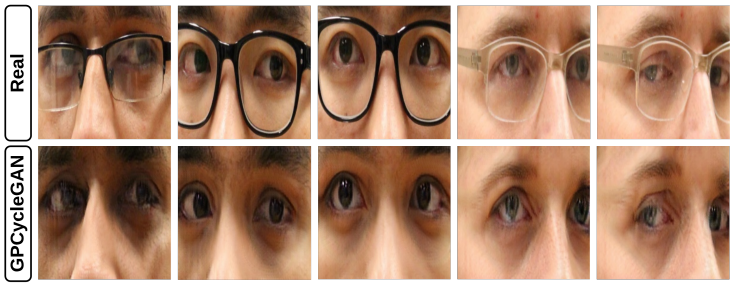


Figure 6.9: Example real RGB images with eyeglasses from the Columbia gaze dataset, and corresponding generated images after eyeglass removal.

used our pre-trained gaze classifier from the RGB dataset (Section 6.5.2) to provide the gaze consistency loss for the GPCycleGAN. The results from this model on the test split are shown in the Figure 6.9. Despite using a gaze classifier trained on a different dataset, we can see that the resulting images preserve the gaze direction in the Columbia gaze dataset.

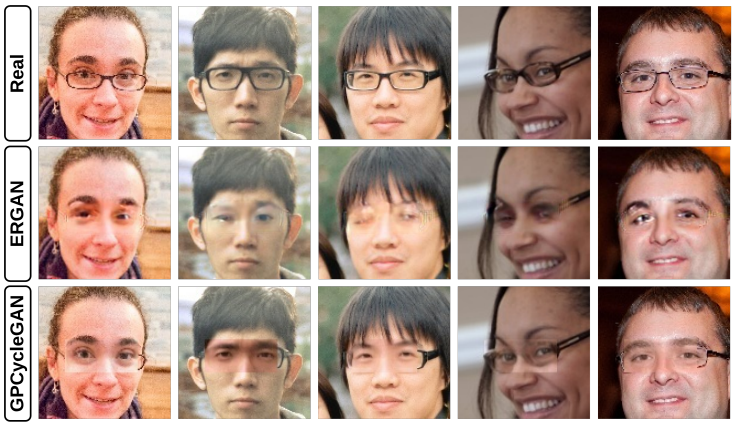


Figure 6.10: Example real RGB images with eyeglasses from the MeGlass dataset, and corresponding generated images after eyeglass removal. We compare the proposed GPCycleGAN approach with ERGAN.

Results on the MeGlass Dataset. For this experiment, we split the dataset based on person identities. Of all 1710 identities in the dataset, we assign 1197 to the train split and 513 to the test split. This ensures cross-subject testing in our evaluation. Similar to the Columbia dataset, as the MeGlass dataset does not have labelled gaze directions, we simply use our pre-trained gaze classifier to train the GPCycleGAN model. Since there is no prior work on eyeglass removal for gaze estimation, we compare our approach with the recently proposed eyeglasses removal

model ERGAN [199] (Figure 6.10). To aid comparison, we overlay our eye crop images after eyeglass removal on the original face image. From this comparison, we can see that ERGAN fails to preserve the gaze direction as it does not enforce any constraints on the gaze. On the other hand, our approach can preserve gaze despite using a gaze classifier trained on a completely different dataset. This also illustrates the benefits of applying eyeglass removal models to eye crop images rather than entire face images, especially if the downstream task is gaze estimation. We believe this is a more efficient use of the generator's modelling capacity.

6.5.4 Analyzing the Class Activation Maps

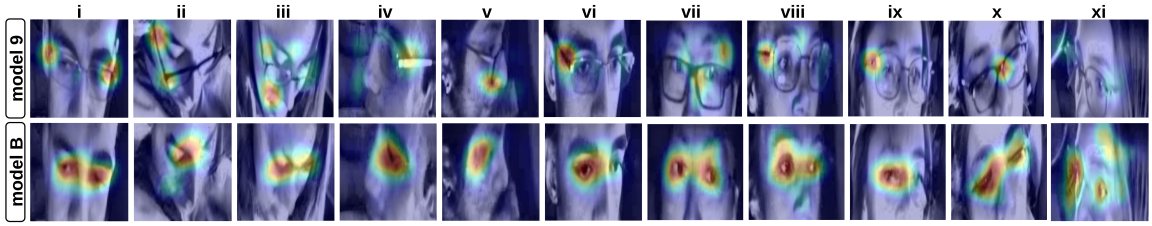


Figure 6.11: IR eye image crops from the test set overlaid with ground truth CAMs produced by the gaze classifier. **Top row:** raw eye image crops with ground truth CAMs produced by model ⑨. **Bottom row:** same eye image crops after eyeglass removal and updated ground truth CAMs produced by model Ⓑ.

Observing the Class Activation Maps (CAMs) produced by classifier models is a simple and effective way of interpreting CNNs and assessing their robustness. In this spirit, we visualize and contrast the CAMs from our proposed approach (model Ⓑ) and the baseline solution (model ⑨). This visualization is presented in Figure 6.11, where eye image crops from the test set are overlaid with the corresponding ground truth CAMs i.e. the CAMs corresponding to the correct gaze zone. In particular, we compare CAMs generated when raw eye image crops with eyeglasses are directly fed to a gaze classifier (as in model ⑨) with CAMs that are obtained by first removing eyeglasses from the eye crop images using a pre-processing network, then feeding this processed image to a fine-tuned gaze classifier (as in model Ⓑ). The CAMs clearly indicate that aside from

being superior in terms of quantitative metrics, our proposed model exhibits better interpretability, robustness, and more accurately localizes the eye regions. The baseline model - despite being trained on images with eyeglasses - fails to learn the true nature of the task, and instead overfits to unrelated visual cues. This portends better generalization for our model in comparison to the baseline.

6.6 Chapter Summary

Reliable and robust gaze estimation on real-world data is essential yet hard to accomplish. A driver's gaze is especially important in the age of partially automated vehicles as a cue for gauging driver state/readiness. In this study, we improved the robustness and generalization of gaze estimation on real-world data captured under extreme conditions, such as data with the presence of eyeglasses, harsh illumination, nighttime driving, significant variations of head poses, etc. For dealing with issues arising from bad lighting, we demonstrate that using an IR camera with suitable equalization/normalization suffices. For images that include eyeglasses, we present eyeglass removal as a pre-processing step using our proposed Gaze Preserving CycleGAN (GPCycleGAN). The GPCycleGAN enables us to train a generator that is capable of removing eyeglasses while retaining the gaze of the original image. This ensures accurate gaze zone classification by a downstream SqueezeNet model. We show that this combined model outperforms both the baseline approach and the vanilla CycleGAN + SqueezeNet model considerably. We also provide example results and illustrations to demonstrate the superiority of our model in terms of interpretability, robustness, and generality. Future work entails improving on the architectures of different components like generator, discriminator, and gaze classifier.

6.7 Acknowledgements

Chapter 6, in full, is a reprint of the material as it appears in the IEEE Transactions on Intelligent Vehicles (2021), by Akshay Rangesh, Bowen Zhang, and Mohan M. Trivedi. The dissertation author was one of the primary investigators and authors of this paper.

Chapter 7

HandyNet: A One-stop Solution to Detect, Segment, Localize & Analyze Driver Hands

Tasks related to human hands have long been part of the computer vision community. Hands being the primary actuators for humans, convey a lot about activities and intents, in addition to being an alternative form of communication/interaction with other humans and machines. In this study, we focus on training a single feed-forward convolutional neural network (CNN) capable of executing many hand related tasks that may be of use in autonomous and semi-autonomous vehicles of the future. The resulting network, which we refer to as HandyNet, is capable of detecting, segmenting and localizing (in 3D) driver hands inside a vehicle cabin. The network is additionally trained to identify handheld objects that the driver may be interacting with. To meet the data requirements to train such a network, we propose a method for cheap annotation based on chroma-keying, thereby bypassing weeks of human effort required to label such data. This process can generate thousands of labeled training samples in an efficient manner, and may be replicated in new environments with relative ease.

Video results
Dataset
Code: https://github.com/arangesh/HandyNet

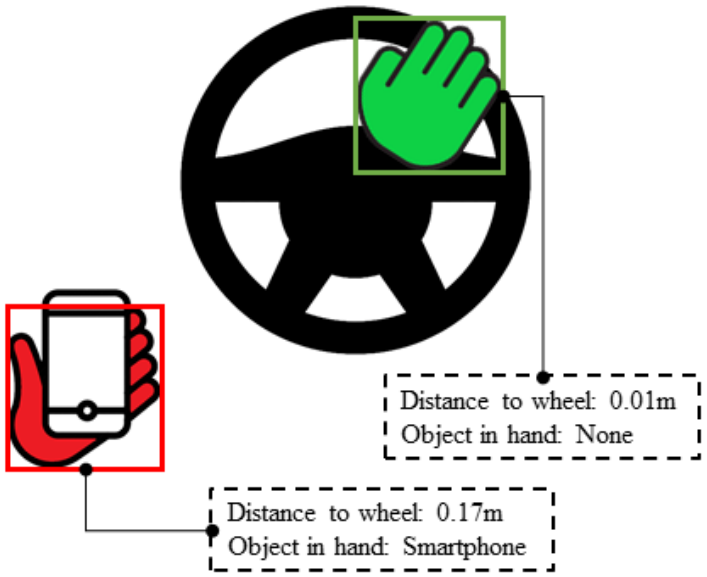


Figure 7.1: Illustration of the intended goal our research. The proposed method is capable of detecting and segmenting driver hands, localizing them in 3D, and identifying the object each hand may be holding.

7.1 Introduction

The past decade has seen a rapid increase and improvement in the automation offered on consumer automobiles. This may be attributed to a corresponding growth in the technology and engineering required to make such automation reliable enough for widespread deployment. Based on the definitions provided by SAE International, vehicles with *partial* and *conditional* automation are already used in notable numbers across the world. Such vehicles would only become more ubiquitous with the improvement in technology, increase in the number vehicles offered with some form of autonomy, and the dwindling costs associated with such vehicles. Moreover, vehicles with automation will see a continued growth in the level of automation offered, with the end goal being complete automation - a situation where the driver is just another passenger. However, until a point of complete automation is reached, the perils of partial automation need to be dealt with.

The dangers of partial automation primarily arise from the need for the driver to be

constantly monitoring the drive, or in the case of conditional automation, for the driver to be ready to take-over control at any given time. Yet decades of human factors research has shown that humans are not particularly good at tasks that require vigilance and sustained attention over long periods of time [2]. It is also well known that rising levels of automation will lead to declining levels of awareness on the part of the human operator [1]. This seems to suggest that it is not a matter of *if*, but *when* a driver will resort to non-ideal behavior, especially since most automated systems are designed to free the driver to do something else of interest. Thus, it is of extreme importance to monitor the driver and assess his/her readiness to take control in case of an unexpected failure of the system. This is the irony of automation, whereby the more advanced a system is, the more crucial may be the contribution of the human operator.

In this study, we propose a system that goes a long way towards monitoring a driver and assessing his/her readiness. In particular, we monitor and analyze driver hands beyond what is currently possible with just tactile sensors on the steering wheel. The main contributions of this study are as follows: We propose a convolutional neural network capable of detecting and segmenting driver hands from depth images. This makes localizing the hands (in 3D) inside a vehicle cabin possible, which in turn enables the calculation of the distance of each hand to critical control elements like the steering wheel. In addition to this, the network is capable of identifying objects held by the driver during a naturalistic drive. All this is made feasible by a form of semi-automatic data labeling inspired by chroma-keying which we describe in detail.

7.2 Related Research

To the best of our knowledge, there is no existing work that addresses the problem of segmenting instances of driver hands in a naturalistic driving setting. We therefore provide a brief overview of works that pertain to driver and/or passenger hands inside a vehicle cabin.

The work in [200] outlines common challenges associated with detecting driver hands

in RGB images in addition to proposing a dataset to train such detectors. The authors train and evaluate a few such detectors based on Aggregate Channel Features (ACF) as outlined in [201]. In addition to just detecting driver hands, the authors in [202] also track both hands of the driver using a tracking-by-detection approach. They also leverage common hand movement patterns to stably track hands through self occlusions. Moving over to the domain of human computer interaction, studies like [203–205] identify driver hand gestures from a sequence of depth images. These methods directly produce the identified gesture from raw depth images and do not localize driver hands as an intermediate step. There have also been numerous contributions in the field of hand pose estimation, both inside a vehicle and otherwise. We refer the reader to [206] for a detailed survey on this subject. Finally, there has been noticeable work on analyzing driver hand activity. In [207], the authors extract hand-designed features around pre-defined regions of interest like the steering wheel, infotainment control etc. to identify regions with high hand activity. The authors in [208] take a different approach to identifying regions of activity by detecting and tracking hands for short periods of time and analyzing the temporal dynamics of hand locations. They also go on to detect abnormal events and activities. More recently, the authors in [209] have proposed a unique dataset for detecting and tracking driver hands inside vehicles. This dataset is captured using a Leap Motion device mounted behind the steering wheel. Although this approach makes detecting hands on the wheel much simpler, it also forgoes the capability to know where the driver hands are, if not on the wheel.

In all studies listed above, the major limitation arises from the inherent depth ambiguity. The input to most methods are RGB images, severely restricting the utility of the outputs they produce. Simply localizing hands in 2D does not inform us about crucial information like the 3D distance to different control elements inside a vehicle cabin. Moreover, all these methods are heavily dependent on the camera view used for training, and re-training for new views would require devoting considerable efforts towards ground truth annotation. Even for methods relying on depth data, the end goal is achieved without actually localizing the hands. This is primarily due

to the lack of sufficiently labeled depth data to train such algorithms. In this study, we overcome all these stumbling blocks and produce thousands of labeled depth images with relative ease. This method may also be replicated with little effort in new environments, and for different camera views.

7.3 Semi-Automatic Labeling based on Chroma-Keying

The key contribution of our work is a method for generating large amounts of labeled data in a relatively cheap manner for the task of hand instance segmentation. As is well known, deep learning methods although extremely powerful and accurate, require large amounts of labeled data to learn and generalize well. This requirement becomes even more unwieldy for tasks like semantic and instance segmentation, where pixel level annotations are required. Such tasks entail several hundred hours of human effort to generate enough samples for successfully training networks. These difficulties are usually overcome by hiring large groups of human "annotators", either directly or through a marketplace such as the Amazon Mechanical Turk. This approach has its own limitations. First, this requires some form of monetary incentive which may be beyond the resources that are available. More importantly, networks trained on a particular dataset tend to perform best on similar data. Therefore, to ensure similar performance on the same task for a different set of data, more retraining on such data would be required; this leads to more expensive annotations. In this section, we describe a form of semi-automatic labeling based on *chroma-keying*. This method can be replicated in different environments and even for different tasks with relative ease.

7.3.1 Acquiring Instance Masks

Chroma-keying is a technique popular in the visual effects community for layering images i.e. separating specific foregrounds from a common background based on color hues. This usually

Algorithm 7.1 Pseudocode for obtaining instance masks for a given sequence by chroma-keying

Input: $\{rgb_i^{reg}, depth_i^{reg}\}_{i=1}^{N}$ ▷ registered RGB and depth for each frame i
Output: $\{inst\_masks_i\}_{i=1}^{N}$ ▷ binary mask for each hand instance in every frame

for $i \leftarrow 1$ **to** N **do**
$r_i^{reg} \leftarrow rgb_i^{reg}(:,:,1)$ ▷ red channel
$g_i^{reg} \leftarrow rgb_i^{reg}(:,:,2)$ ▷ green channel
$b_i^{reg} \leftarrow rgb_i^{reg}(:,:,3)$ ▷ blue channel
$Y_i^{reg} \leftarrow 0.3 \cdot r_i^{reg} + 0.59 \cdot g_i^{reg} + 0.11 \cdot b_i^{reg}$ ▷ relative luminance
$masks \leftarrow (g_i^{reg} - Y_i^{reg}) \geq threshold$
$\{mask_j\}_j \leftarrow CCL(masks)$ ▷ connected component labeling
/*For cases where two instances might be merged in 2D, but are disjoint in 3D*/
for each $mask_k \in \{mask_j\}_j$ **do**
$depth\_pixels \leftarrow depth_i^{reg}(mask_k)$ ▷ depth pixels corresponding to each mask
$opt\_thresh \leftarrow otsu(depth\_pixels)$ ▷ get optimal threshold using Otsu's method
$mask_{k_1} \leftarrow (depth\_pixels \geq opt\_thresh)$
$mask_{k_2} \leftarrow (depth\_pixels < opt\_thresh)$
$\{mask_j\}_j \leftarrow \{mask_j\}_j \backslash \{mask_k\}$
$\{mask_j\}_j \leftarrow \{mask_j\}_j \bigcup \{mask_{k_1}, mask_{k_2}\}$

end for
/*For cases where a hand might be partially occluded resulting in disjoint regions*/
Require: For each element in $\{mask_j\}_j$, the area in pixels is known.
$inst\_masks_i \leftarrow \{\}$
while $\{mask_j\}_j \neq \phi$ **do**
$mask_{j'} \leftarrow$ largest $mask \in \{mask_j\}_j$
if $Area(mask_{j'}) \leq 20$ **then**
$\{mask_j\}_j \leftarrow \{mask_j\}_j \backslash \{mask_{j'}\}$
continue
end if
for each $mask_k \in \{mask_j\}_j, k \neq j'$ **do**
$dist \leftarrow Distance\_3d(mask_k, mask_{j'})$
▷ distance is calculated between centroids using Equation 7.1
if $dist \leq 7cm$ **then**
$\{mask_j\}_j \leftarrow \{mask_j\}_j \backslash \{mask_k\}$
$\{mask_j\}_j \leftarrow \{mask_j\}_j \backslash \{mask_{j'}\}$
$mask_{j'} \leftarrow mask_{j'} \bigcup mask_k$ ▷ combine $mask_k$ and $mask_{j'}$
end if
end for
$inst\_masks_i \leftarrow inst\_masks_i \bigcup \{mask_{j'}\}$
end while
end for

involves the use of green screens and body suits that visually contrast the object of interest from the other elements of scene, and hence can be separated with ease.

To leverage this technique for the task at hand, we make use of a Kinect v2 sensor comprising of an RGB and infrared (depth) camera with a small baseline. Note that we chose a Kinect for its ease of use, excellent community support and high-resolution depth images; however, any calibrated pair of RGB and depth cameras may be used. The input to the proposed network is the in-painted, registered version of the raw depth image, and the desired outputs are the instance masks for each hand of the driver. The registered RGB and depth images are captured using Kinect drivers provided by OpenKinect [210], and the depth images are in-painted by applying cross-bilateral filters at three image scales [211]. The instance masks for supervision during training are obtained by chroma-keying the registered RGB images using the procedure detailed in Algorithm 7.1. This requires the driver (subject) to wear green gloves as shown in Figure 7.3. We also make the subjects wear red wrist bands to ensure a clear demarcation for where the wrist ends and the hand begins. Additionally, we add soft lights inside the vehicle cabin to ensure the green gloves are uniformly lit. Note that good lighting is extremely important for accurate chroma-keying. Registering the RGB and depth images results in a one-to-one correspondence between every pixel in both images. This ensures that the masks obtained by chroma-keying the registered RGB images are valid supervision for the registered depth images. This way, the registered RGB images are only used for labeling hand instances during training, and are unused when the trained network is deployed. This entire procedure is illustrated in Figure 7.2, and example inputs and labels generated by this procedure are shown in Figure 7.3.

At this point, we would like to point out two caveats of this approach. First, two hand instances are considered unique when their distance in 3D is sufficiently large. This then implies that two instances very close to each other (for example, when a driver clasps both hands) cannot be separated. We do not try to separate such instances and force the proposed network to detect such *merged* instances. Alternatively, one could make each subject wear two differently colored

Algorithm 7.2 Pseudocode for labeling handheld objects for a given sequence

Require: The driver holds the same object using the same hand for the entire duration of the sequence

/*Note that only one instance per frame is assigned *label*; other instances are assigned 0 corresponding to *no handheld object*. This is valid by the requirement stated above.*/

Input: $\{inst\_masks_i\}_{i=1}^{N}$, where $inst\_masks_i = \{mask_i^1, \cdots, mask_i^{M_i}\}$,
▷ binary mask for each hand instance in every frame
$label \in \{1,2,3,4\}$, ▷ label for the object used in the sequence
$m_1 \in \{1, \cdots, M_1\}$ ▷ user provided instance associated with *label* for the first frame

Output: $\{o_i\}_{i=2}^{N}$ ▷ object label for each hand instance in every frame

$o_1 \leftarrow \{0\}^{M_1}$ ▷ initialize all instances with zeros
$o_1(m_1) \leftarrow label$ ▷ assign *label* to instance holding object
$last \leftarrow mask_i^{m_1}$ ▷ store last instance holding object
/*Find instance in current frame closest to last known instance holding object*/
for $i \leftarrow 2$ **to** N **do**
    $min\_dist \leftarrow \infty$
    for $j \leftarrow 1$ **to** M_i **do**
        $cur\_dist \leftarrow Distance\_3d(mask_i^j, last)$
        ▷ distance is calculated between centroids using Equation 7.1
        if $cur\_dist \leq min\_dist$ **then**
            $m_i \leftarrow j$
            $min\_dist \leftarrow cur\_dist$
        end if
    end for
    $o_i \leftarrow \{0\}^{M_i}$ ▷ initialize all instances with zeros
    /*The following condition handles cases where the hand holding the object is temporarily occluded*/
    if $min\_dist \leq 15cm$ **then**
        $last \leftarrow mask_i^{m_i}$
        $o_i(m_i) \leftarrow label$
    end if
end for

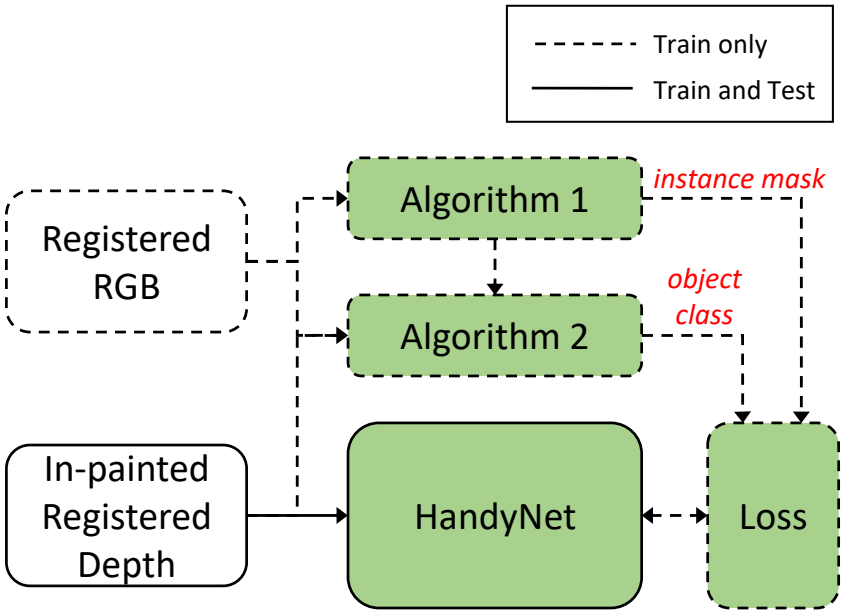
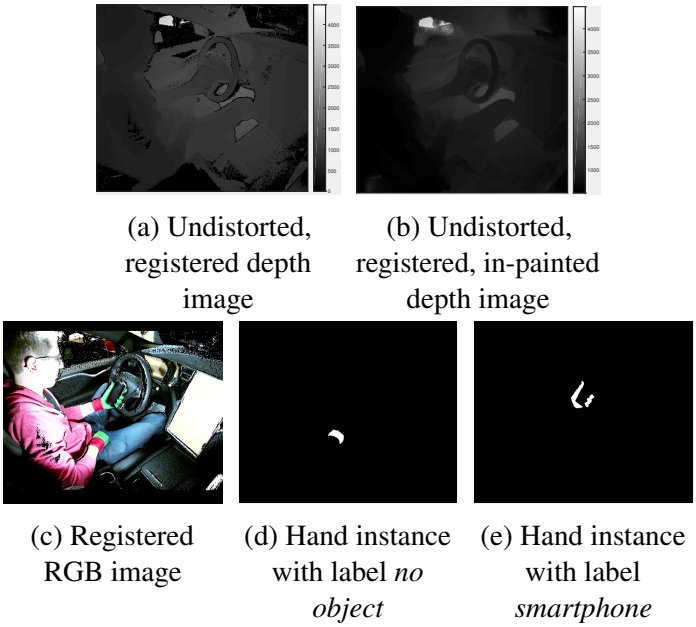


Figure 7.2: Block diagram depicting proposed training and testing methodology.

gloves. Second, one may argue that any algorithm trained on data with subjects wearing gloves and captured under controlled lighting may not generalize well to real world scenarios. We disprove such arguments by providing large amounts of qualitative results (images and videos) on multiple hours of real world drives, with subjects not present in the training split.

Although using depth images as input allows us to use chroma-keying for cheap supervision, we are motivated by other factors as well. First, using depth images as input circumvents any privacy issues that may arise with having cameras inside cars. Second, depth cameras are relatively unaffected by harsh illumination or lack thereof (e.g. during nighttime driving). Finally, once driver hands are located in depth images, it is straightforward to calculate where the hands are in 3D, and how far they are from critical control elements like the steering wheel. Such information may be extremely useful to gauge the readiness of a driver to *take-over* control from a semi-autonomous/autonomous vehicle.

For a pixel at $(x,y)^T$ in an undistorted depth image, the corresponding 3D point location



(a) Undistorted, registered depth image

(b) Undistorted, registered, in-painted depth image

(c) Registered RGB image

(d) Hand instance with label *no object*

(e) Hand instance with label *smartphone*

Figure 7.3: **Input and desired outputs used to train the proposed network:** The input to the network (shown in (b)) is obtained after smoothing the raw undistorted depth (shown in (a)), while the desired outputs ((d) and (e)) are obtained from the registered RGB image (shown in (c)) using Algorithms 7.1 and 7.2.

Table 7.1: List of handheld object classes and types of objects included in each class.

Class ID	Class Label	Objects Included
0	no object	-
1	smartphone	cellphones, smartphones
2	tablet	iPad, Kindle, tablets
3	drink	cups, bottles, mugs, flasks
4	book	newspapers, books, sheets of paper

$(X,Y,Z)^T$ is obtained as follows:

$$X = \frac{(x-c_x)\cdot d}{f_x}, Y = \frac{(y-c_y)\cdot d}{f_y}, Z = d, \tag{7.1}$$

where d is the depth value at pixel $(x,y)^T$, $(c_x,c_y)^T$ is the principal point, and $\{f_x,f_y\}$ are the focal lengths of the depth camera. For this study, we use the parameters provided by [211] without any re-calibration.

7.3.2 Acquiring Handheld Object Labels

In addition to locating hand instances in depth images, we also aim to identify objects in driver hands. To facilitate training for the same, we label each hand instance (obtained using Algorithm 7.1) in a semi-automatic manner with minimal human input. We do this by following the procedure detailed in Algorithm 7.2, the only requirement being that the driver (subject) holds the same object in the same hand for the entirety of a captured sequence. Multiple sequences are then captured with different subjects and different objects in each hand, one hand at a time. The variety of handheld objects considered for this study and the semantic classes they fall under are listed in Table 7.1.

7.4 HandyNet

7.4.1 Network Architecture

HandyNet is largely based on the state-of-the-art Mask R-CNN [212] architecture. This architecture consists of two stages in sequence. First, the Region Proposal Network (RPN) generates class agnostic regions of interest (RoIs). We use the first 4 convolutional stages of ResNet-50 with features pyramids [37] as the backbone for this purpose. The input to this network is the undistorted, registered, in-painted depth image as the sole channel.

The second stage of the Mask R-CNN architecture is made of task specific heads. In our HandyNet architecture, we retain the structure of the mask head, and split the bounding box regression and classification head into two separate heads. While the mask and bounding box regression heads receive the same RoI from the RPN, the classification head receives a slightly larger region of interest (RoI^{+}). This follows from the reasoning that unlike conventional object classification, we are attempting to classify the handheld object for a given hand instance. This slightly larger RoI^{+} might be favorable for identifying larger objects like tablets and newspapers.

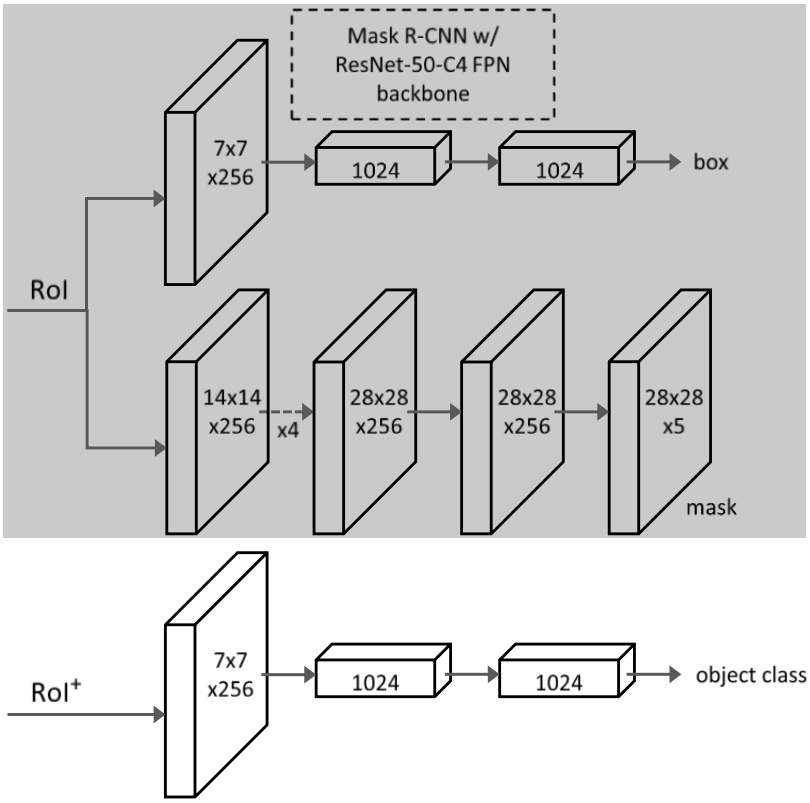


Figure 7.4: **Head Architecture of HandyNet:** We retain the mask head from Mask R-CNN [212] but separate the classification and bounding box regression head. The classification head receives a larger region of interest (RoI$^+$) to better identify larger handheld objects.

For an RoI parametrized by (x, y, w, h), we define RoI$^+$ parametrized by (x', y', w', h') as follows:

$$x' = x - \alpha \cdot w,\ y' = y - \alpha \cdot h,$$
$$w' = (1 + 2\alpha) \cdot w,\ h' = (1 + 2\alpha) \cdot h, \tag{7.2}$$

where α is the factor by which the region of interest is expanded. We choose the value of α based on cross validation.

7.4.2 Implementation Details

As in Mask R-CNN, an RoI is considered positive if it has IoU with a ground-truth box of at least 0.5 and negative otherwise. We feed the input images at full-resolution i.e. without resizing. The training and inference configurations for HandyNet are listed in Table 7.2. We make changes to the original configurations from Mask R-CNN to account for the average number of hand instances in a typical depth image, the scale of the hand instances encountered, and the more structured nature of the task at hand. All variants of the proposed network are trained for a total of 120 epochs using the following schedule: First, the RPN (backbone) is trained for 40 epochs with a learning rate of 0.001. Next, the fourth stage of the ResNet-50 backbone (RPN) along with all task specific heads are trained for an additional 40 epochs with the same learning rate. Finally, the entire network is trained for 40 epochs with a reduced learning rate of 0.0001. We use a momentum of 0.9 and a weight decay of 0.0001 throughout. Note that HandyNet is initialized from scratch with random weights. This is made possible by the huge labeled dataset available for training (see Table 7.3).

7.5 Experimental Analysis

To test the viability of our semi-automatic labeling approach, we split the entire data as follows: the training and validation sets were mostly captured indoors with suitable lighting for chroma-keying. All subjects were wearing green gloves with red wrist bands as described in Section 7.3. The subjects were asked to imitate naturalistic driving while holding different objects. The testing set is mostly comprised of data from real world drives i.e. the subject is actually driving the car. The subjects may interact with different objects as they would normally do in a drive. The remaining data in the testing set is captured indoors in a manner similar to the training/validation set. This is done to ensure that all objects included in the training set are covered in the test set as well. Moreover, all data is captured on a Tesla Model S testbed,

Table 7.2: Training and inference configurations used for HandyNet.

Input image size	424×512
Batch size	6
RPN[1] anchor scales	$\{16, 32, 64, 128\}$
RPN[1] anchor aspect ratios	$\{0.5, 1, 2\}$
Number of anchors per image used for RPN[1] training	64
Number of RoIs per image retained for training the heads	20
Ratio of positive RoIs per image	0.1
RoIs retained post NMS during training	100
RoIs retained post NMS during inference	50

[1] Regional Proposal Network

where the drivers can take their hands off the wheel for extended periods of time. More details on each split are provided in Table 7.3. In addition to this, we provide some qualitative results in Figure 7.5 from completely naturalistic drives i.e. real drives where subjects are not wearing gloves or wrist bands. We also ensure that no drivers (subjects) overlap between different splits.

7.5.1 Timing

Training: HandyNet with a ResNet-50-C4 RPN takes about 52 hours to train from scratch on our system with a 6 core Intel 990X processor and a Titan X Maxwell GPU. The training data is stored in a SATA III Solid State Drive (SSD).

Inference: Inference for HandyNet using the same system above runs at approximately 15Hz. This includes the time for fetching and pre-processing the data. The relatively smaller RPN (with ResNet-50-C4 backbone), fewer number of anchor scales, and the fewer number of RoIs

Table 7.3: Details of the train-val-test split used for the experiments.

Split	Number of Unique Drivers	Number of Frames	Number of Hand Instances	Fraction of Naturalistic Driving Data
Training	7	128,317	219,369	0.2
Validation	1	6,897	13,525	0.0
Testing	2	36,497	69,794	0.9

after non-maximal suppression all contribute to making the network run faster during inference.

7.5.2 Ablation Experiments

We perform comprehensive ablations on HandyNet using the validation and testing splits. We report standard COCO metrics [213] including AP (averaged over IoU thresholds), AP_{50}, AP_{75}, and AP_S, AP_M (AP at small and medium scales). We do not provide AP_L (AP for large objects) due to the lack of large instances. Note that AP is evaluated using mask IoU.

First, we determine the value of the hyperparameter α through cross validation. Table 7.4 lists the various APs for $\alpha \in \{0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6\}$ on the validation split. We see that expanding the RoI improves the overall performance consistently until $\alpha = 0.5$, after which we observe a saturation point. Increasing α beyond 0.5 only seems to benefit object classification for very small hand instances, which are quite seldom. Hence, it was sensible for us to ignore such rare cases and optimize the performance for the more commonly observed hand sizes. With this reasoning, we chose HandyNet with $\alpha = 0.5$ for reporting results on the testing set.

Next, we provide both class agnostic and class sensitive results for the best performing model on the testing split (Table 7.5). Note that the class in our task is the associated handheld object class and not the semantic class itself. We see that the class agnostic performance is robust for all AP metrics, indicating our networks capability to successfully localize and segment driver hands. This also proves that our network generalizes well to naturalistic driving data captured

Table 7.4: **Ablation results on validation split:** Mask AP for HandyNet with different values of expansion factor α.

α	**AP**	**AP**$_{50}$	**AP**$_{75}$	**AP**$_S$	**AP**$_M$
0.0	28.8	48.7	26.0	24.7	43.3
0.1	28.9	49.1	26.5	25.1	43.4
0.2	29.2	49.3	27.1	25.4	43.8
0.3	29.9	49.8	27.3	26.2	43.9
0.4	30.0	49.7	27.7	28.6	44.0
0.5	**30.6**	**51.8**	**28.0**	29.4	**44.9**
0.6	30.5	51.7	27.8	**29.6**	44.5

Table 7.5: **Class agnostic and class sensitive results on testing split:** Mask AP for best performing model ($\alpha = 0.5$).

Type of evaluation	**AP**	**AP**$_{50}$	**AP**$_{75}$	**AP**$_S$	**AP**$_M$
class agnostic	42.9	83.3	40.4	34.7	50.8
class sensitive	30.3	51.2	27.9	28.5	44.0

with drivers not part of the original training split. As expected, the class sensitive APs are lower in comparison to the class agnostic ones, but not by much. Based on the qualitative and quantitative evaluation we have conducted, we believe that our network successfully identifies objects that are not too dissimilar to the objects it has been trained on. However, for images with considerably different handheld objects, or in situations where the objects are not completely visible due to the manner in which they are grasped, our network fails to produce the correct output. These issues could probably be solved by either gathering more diverse data, changing the camera view, or by incorporating temporal information. We leave these questions for future work.

7.6 Chapter Summary

In this chapter, we present HandyNet - a CNN that uses depth images to execute hand related tasks that may be of use in autonomous and semi-autonomous vehicles of the future. This

includes detecting and localizing driver hands in 3D within a vehicle cabin, and additionally identifying handheld objects. Training such a network is made possible by our proposed method for semi-automatic labeling based on chroma-keying. The entire data used to train HandyNet from scratch ($128,317$ images and $219,369$ hand instances) was captured and labeled within a single day. This demonstrates the ease with which similar networks can be trained for new environments and different camera views. We hope this work inspires more ways to produce cheaply labeled data for related tasks, especially when the alternative is several hundred hours of human effort.

7.7 Acknowledgements

Chapter 7, in full, is a reprint of the material as it appears in the IEEE Conference on Computer Vision and Pattern Recognition - 3D Humans Workshop (2018), by Akshay Rangesh, and Mohan M. Trivedi. The dissertation author was the primary investigator and author of this paper.

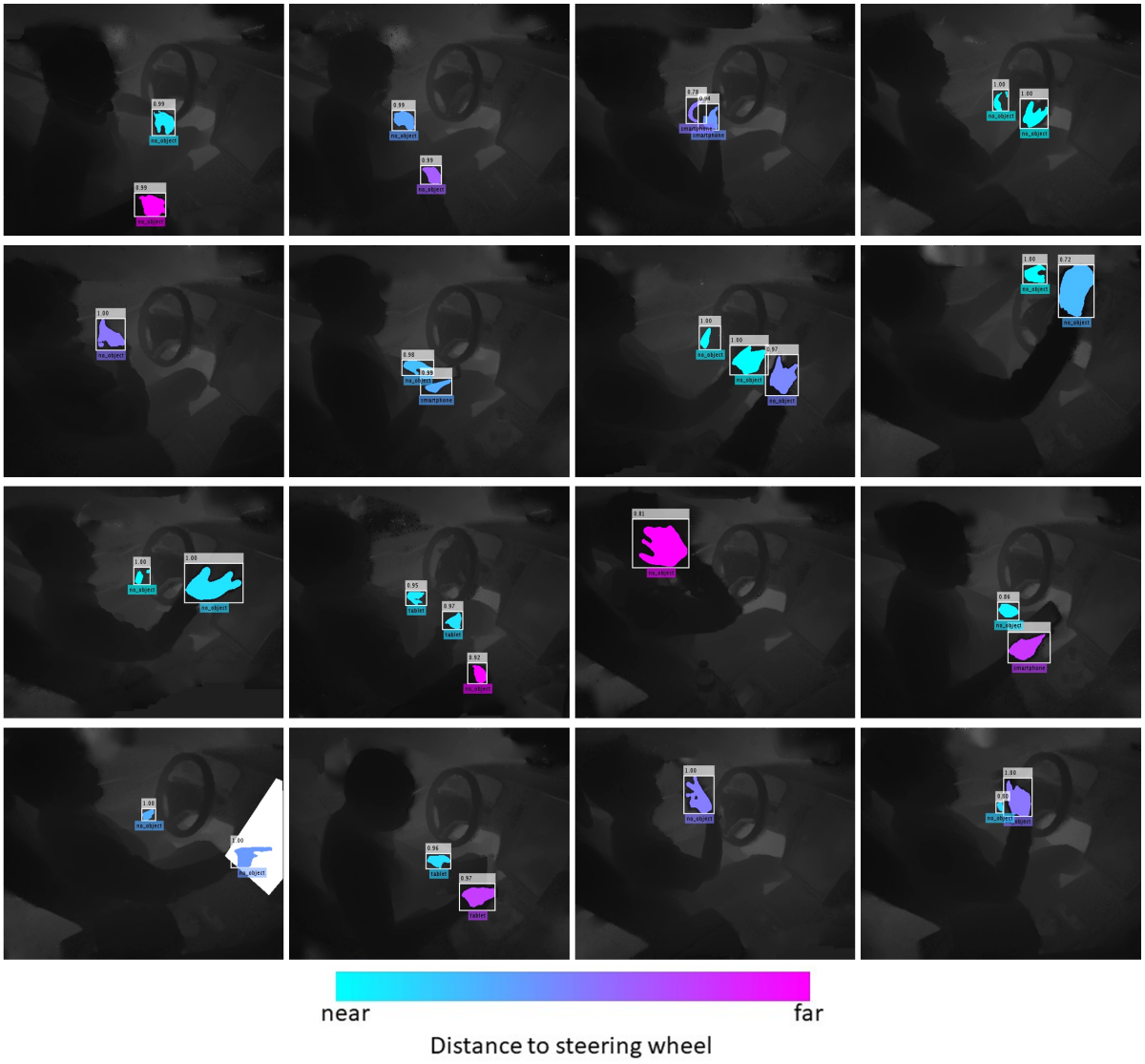


Figure 7.5: **HandyNet results on naturalistic driving data (best viewed in color):** Results consist of drivers and objects not part of the training, validation and test sets. Operating on depth data allows us to gauge accurate distances to control elements like the steering wheel (see colorbar above) and the interactive display (see bottom left image). For our experiments, the steering wheel and interactive display were labeled by a human as part of a one-time calibration setup.

Chapter 8

Forced Spatial Attention for Driver Foot Activity Classification

This chapter provides a simple solution for reliably solving image classification tasks tied to spatial locations of salient objects in the scene. Unlike conventional image classification approaches that are designed to be invariant to translations of objects in the scene, we focus on tasks where the output classes vary with respect to where an object of interest is situated within an image. To handle this variant of the image classification task, we propose augmenting the standard cross-entropy (classification) loss with a domain dependent Forced Spatial Attention (FSA) loss, which in essence compels the network to attend to specific regions in the image associated with the desired output class. To demonstrate the utility of this loss function, we consider the task of driver foot activity classification - where each activity is strongly correlated with where the driver's foot is in the scene. Training with our proposed loss function results in significantly improved accuracies, better generalization, and robustness against noise, while obviating the need for very large datasets.

Code: https://github.com/arangesh/Forced-Spatial-Attention

8.1 Introduction

Image classification being one of the fundamental tasks in computer vision receives large amounts of research effort, and consequently sees remarkable progress year after year [214–220]. This is true, especially for applications with sufficient training data per class, which is a well understood problem. To ensure better generalization, traditional image classification approaches introduce certain inductive biases, one of which is invariance to spatial translations of objects in images, i.e. the locations of objects of interest in an image does not change the true output class of the image. This is typically enforced by data augmentation schemes like random translations, rotations, crops etc. Even convolution kernels - the basis of most Convolutional Neural Networks (CNNs) are shared across the entire spatial extent of features as a means to learn translation invariant features. In this study, we are interested in image classification applications where these assumptions do not necessarily hold true. Specifically, we focus on tasks where relative locations of objects in the scene influence the output class of the image.

Many real world examples of such tasks can be found in the surveillance domain. For example, consider the scenario where we would like identify when an unauthorized person is in close proximity to a stationary object like a car/door/safe etc. If this were set up as an image classification problem, the desired output class would vary based on where the unauthorized person is in the image i.e. if the person is very close to the stationary object and exhibiting unusual behavior, then trigger an alarm; else do not. In this study, we try to solve a similar problem from the automotive domain. In particular, we wish to design a very simple and reliable system to classify the foot activity of drivers in cars. This problem is comprised of 5 classes of interest, namely - *away from pedals*, *hovering over accelerator*, *hovering over break*, *on accelerator*, and *on break*. As can be inferred from the individual class identities, the desired output changes based on where the driver's foot is in the image. We chose these classes as they are good indicators of a driver's preparatory motion, and are also strongly tied to the time it takes for a driver completely

regain control of the car from an autonomous agent [221, 222] - also known as the *take-over time*.

Before describing our approach, we would also like to address some straightforward ways in which one could potentially solve such problems. One obvious way to encode spatial information in predictions is to use a fully connected (FC) output layer. This however comes at a huge cost of computation, storage, and possibly generalization. Introducing an FC layer would also increase the data requirements considerably, something that is not available in many applications. Another way to approach these problems is to split the task into specialized portions, leading to better generalization and interpretability [223]. For instance, you could have one algorithmic block dedicated to detecting all objects of interest in an image, followed by a second block that would reason over their spatial locations. The major drawback of such approaches is the requirement of ground truth object locations in the image for training the individual blocks. Once again, this is quite expensive to obtain and is not available in many applications of interest. Our proposed approach attempts to reliably solve this class of problems without introducing any of the aforementioned drawbacks.

Our main contributions in this work can be summarized as follows - 1) We propose a simple procedure to modify the training of CNNs that make use of Class Activation Maps (CAMs) [224] so as to introduce spatial and domain knowledge related to the task at hand 2) To this end, we propose a new Forced Spatial Attention (FSA) loss that compels the network to attend to specific regions in the image based on the true output class. 3) Finally, we carry out qualitative and quantitative comparisons with standard image classification approaches to illustrate the advantages of our approach using the task of driver foot activity classification.

8.2 Related Research

Driver foot activity research

Tran et al. conducted some of the earliest research on modeling foot activity inside cars

for driver safety applications. In [225, 226], they track the driver's foot using optical flow, while maintaining the current state of foot activity using a custom Hidden Markov Model (HMM) comprising of seven states. Maximizing over conditional state probabilities then produces an estimate of the most likely foot activity at any given time step. This system was intended as a solution to identify and prevent pedal misapplications, a common cause for accidents at the time. More recently, Wu et al. [227] propose a more holistic system comprising of features obtained from visual, cognitive, anthropometric and driver specific data. They use two models - a random forest algorithm was used to predict the likelihood of various pedal application types, and a multinomial logit model was used to examine the impact of prior foot movements on an incorrect foot placement. Although these resulted in high classification errors, the authors were able to identify features important for identifying and preventing pedal misapplications. In their following study [228], the authors analyze foot trajectories from a driving simulator study, and use Functional Principal Component Analysis (FPCA) to detect unique patterns associated with early foot movements that might indicate pedal errors. Inspired by previous work, the Zeng et al. [229] also incorporated vehicle and road information by looking outside the vehicle to model driver pedal behavior using an Input-Output HMM (IOHMM). Unlike most other methods that make use of potentially privacy limiting video sensors, the authors in [230] use capacitive proximity sensors to recognize four different foot gestures.

Driver foot activity has also been an area of interest for many human factors studies. Recent examples include [231], where the authors collect and reduce naturalistic driving data to identify and understand problematic behaviors like pressing the wrong pedal, pressing both pedals, incorrect trajectories, misses, slips, and back-pedal hooks etc. Elsewhere, Wang et al. [232] conduct a simulator based study to compare unipedal (using the right foot to control the accelerator and the brake pedal) and bipedal (using the right foot to control the accelerator and the left foot to control the brake pedal) behavior among drivers. They found the throttle reaction time to be faster in the unipedal scenario, whereas brake reaction time, stopping time, and stopping

distance showed a bipedal advantage. For a more detailed and historical perspective on driver (and human) foot behavior and related studies, we refer the reader to [81, 143, 233–235].

Class Activation Maps (CAMs)

In this study, we manipulate CAMs by *forcing* them to activate only at certain predefined regions depending on the output class. CAMs originated from weakly-supervised classification research [224], where the authors demonstrated that using a Global Average Pooling (GAP) operation instead of an output FC layers resulted in per-class feature maps that loosely localize objects of interest. This offered additional benefits such as relatively better interpretability and reduced model size. More recently, several studies have tried to improve the localization in CAMs in the weakly supervised regime. Singh et al. [236] improve the localization in CAMs by randomly hiding patches in the input image, thereby forcing the network to pay attention to other relevant parts that contribute to an accurate classification. Other popular methods [237–239] typically contain multiple stages of the same network. The CAMs from the first stage are used to mask out the inputs/features to the second stage, thereby forcing the network to pay attention to other salient parts of an image. This results in a more complete coverage of parts relevant to the true class of an image.

8.3 Methodology

8.3.1 Network Architecture

Our primary focus in this study is to propose a general procedure for training CNNs for image classification in a setting where the output classes are tied to domain dependent spatial locations of activity. Although any CNN architecture could be chosen, we decide to work with the Squeezenet v1.1 architecture [240] for the following reasons: the Squeezenet model is extremely lightweight and therefore less data-hungry, while still retaining sufficient representation power. The model also makes use of CAMs instead of FC layers, thereby making it naturally amenable

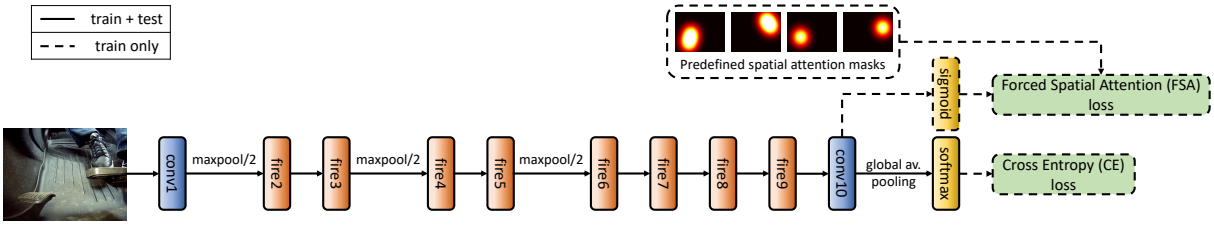


Figure 8.1: Proposed network architecture for training and inference. The network is based on the Squeezenet v1.1 architecture [240] with an additional training-only output branch used to *force* the network's spatial attention.

to the proposed FSA loss that we apply to the normalized CAMs. It must however be noted that models with FC layers can also be made compatible with our procedure by using Gradient-weighted Class Activation Maps (Grad-CAMs) [241]. Finally, using a lightweight architecture like Squeezenet is extremely useful deployment in the real world, where power and computational efficiency are critical.

Most of our experiments begin with a Squeezenet v1.1 model pretrained on Imagenet. During training, we augment the existing architecture with a Forced Spatial Attention (FSA) head that branches off from the existing *conv10* layer that produces the CAMs, before the global average pooling operation (GAP) is applied. This modification is illustrated in Figure 8.1. The FSA head takes as input the CAMs, then normalizes them to $[0, 1]$ through a sigmoid operation. These normalized CAMs along with predefined, domain dependent spatial masks are then used to compute the FSA loss which is backpropagated throughout the network along with the conventional cross entropy (classification) loss. The FSA head and the corresponding FSA loss are used only during training, as a means to inject domain specific spatial knowledge into the network. Once trained, the FSA head is removed and the architecture reverts to its original form.

8.3.2 Forced Spatial Attention

Class Activation Maps (CAMs) are generally used as a means to provide visual reasoning for observed network outputs, i.e. to understand which regions a network attended to, while

producing the observed output. Conversely, if one knows which spatial locations the network must attend to for a desired output class, this can be used as a supervisory signal to train the network. If done correctly, this should reduce overfitting and improve generalization, as the network is *forced* to attend to relevant regions only, while ignoring extraneous sources of information. This is the goal of our proposed FSA loss. We explain this loss more concretely in the context of our desired application, i.e. driver foot activity classification.

The goal of our driver foot activity classification task is to predict one of five activity classes: *away from pedals*, *hovering over accelerator*, *hovering over break*, *on accelerator*, and *on break*, using images from a camera observing the driver's foot inside a vehicle cabin. Examples of these images are provided in Figure 8.2. The next step in our procedure is to create spatial attention masks for some/all output classes. The key idea is to create spatial attention masks with *peaks* at regions depicting the activity corresponding to the output class. Examples of these predefined attention masks for various images and different classes are illustrated in Figure 8.2. Note that the *Away from pedals* class is not associated with any attention maps because it is not tied to any spatial location by definition. On the other hand, certain classes are associated with multiple spatial locations due to slight changes in camera perspective, and also because of the very nature of the activity. For example, the activity *On break* could be associated with different attention masks depending on how far the break pedal is pushed (see Figure 8.2). One issue with having multiple attention masks per class is that we do not know which mask is to be used for a given training image. We address this issue using a two stage training approach described below.

Let A^C denote the CAM and $\mathcal{H}^C = \{H_1^C, H_2^C, \cdots, H_{N_C}^C\}$ denote the set of predefined spatial attention masks for class C. Note that the number of spatial attention masks N_C could be different for each class C. Our classes range from $C = 1, 2, \cdots, 5$ to indicate the five possible output classes. As mentioned earlier, we first apply a pixelwise sigmoid transformation to the CAMs to normalize them to $[0, 1]$:

$$T(A^C) = \frac{1}{1 + \exp(-A^C)}. \tag{8.1}$$

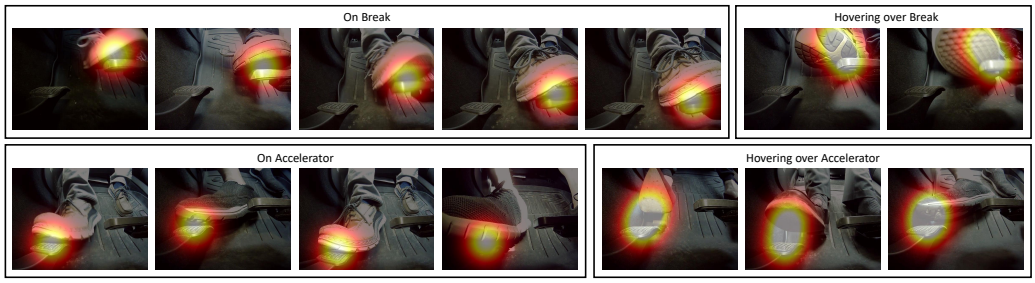


Figure 8.2: Predefined spatial attention masks for each class overlaid on an exemplar input images from the class. Classes are associated with multiple attention masks to account for different foot positions during activities, and slight camera movements. The class *away from pedals* is not associated with a spatial attention mask and has been omitted above.

Next, to resolve the ambiguities arising from having multiple predefined attention maps per class, we use a two stage training procedure - each associated with a different FSA loss. In the first stage, we force the network to attend to all possible regions of interest per class. This is achieved through the loss function:

$$\begin{aligned} L_{FSA}^{stage-1} &= \big(T(A^{C^*}) - \max(\mathcal{H}^{C^*})\big)^2 + \\ &\lambda_{FSA}^{reg} \sum_{\substack{C=1 \\ C \neq C^*}}^{5} \text{mean}\big(T(A^{C^*}) \odot T(A^C)\big), \end{aligned} \tag{8.2}$$

where C^* denotes the ground truth class for a given input image, $\max(\mathcal{H}^{C^*})$ denotes the pixelwise maximum operation applied to all transformed attention maps of the true class C^*, and $\odot$ denotes the Hadamard product between two matrices. We note that the first term of the FSA loss is simply the MSE loss between the ground truth CAM and the pixelwise maximum of all predefined attention masks belonging to the same class. The second term is a regularizer term to encourage independence between CAMS. We observe that omitting the second term leads to *activation leakage*, where CAMs for other classes have high activations in spatial locations corresponding

to the ground truth class. The total loss for the network in stage-1 of training is thus given by

$$L^{stage-1} = L_{CE} + \lambda_{FSA} L_{FSA}^{stage-1}, \tag{8.3}$$

where L_{CE} denotes the standard cross entropy classification loss.

In stage-1 of training, the network is forced to attend to all possible regions of interest for a specific class. In stage-2 of training, we would like the network to *contract* its attention to the region pertinent to the input image. With this in mind, the FSA loss for stage-2 is defined as:

$$\begin{aligned} L_{FSA}^{stage-2} = \min_i \left(\left(T(A^{C^*}) - H_i^{C^*} \right)^2 \right) + \\ \lambda_{FSA}^{reg} \sum_{\substack{C=1 \\ C \neq C^*}}^{5} \text{mean}\left(T(A^{C^*}) \odot T(A^C) \right), \end{aligned} \tag{8.4}$$

where we modify only the first term of the FSA loss. Specifically, we only apply an MSE loss between the ground truth CAM and the predefined attention mask that is most similar in an L2 sense. The reasoning behind this is to make the network choose attention masks that retain features that are most discriminative for each input image. As before, the total loss for the network in stage-2 of training is given by

$$L^{stage-2} = L_{CE} + \lambda_{FSA} L_{FSA}^{stage-2}. \tag{8.5}$$

We demonstrate through our experiments that such a two stage loss results in the network learning to choose the correct attention mask without explicit supervision.

8.3.3 Implementation Details

To create the class specific attention masks $\mathcal{H}^C$, we first collected a set of representative images for each class. These images were chosen to represent the different regions of activity

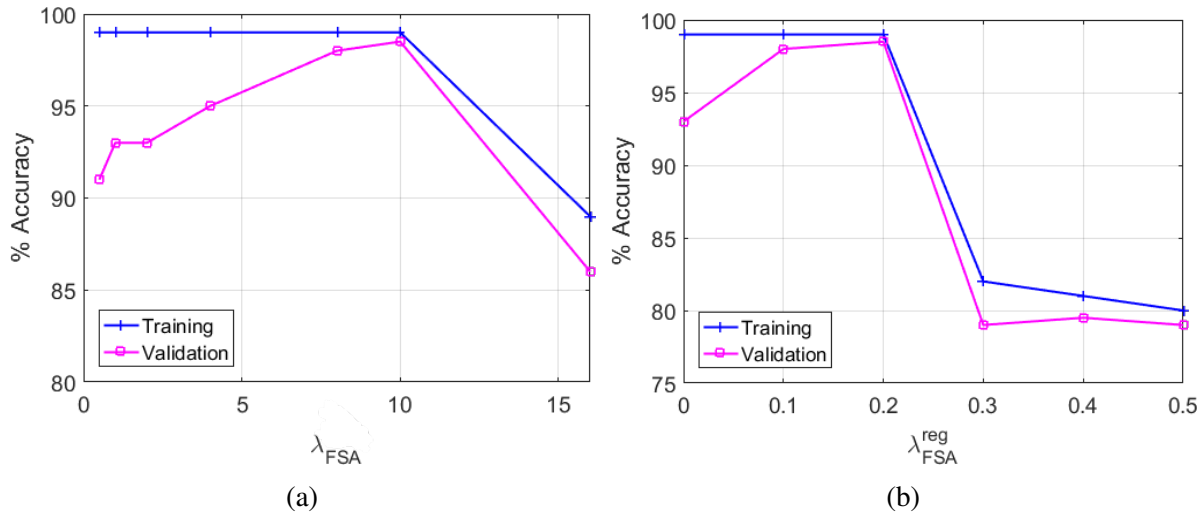


Figure 8.3: Plot of training and validation accuracies for different values of hyperparameter (a)λ_{FSA} and (b)λ^{reg}_{FSA}.

within a given class. Next, we created the various attention masks by manually overlaying a 2D Gaussian peak with suitable variance over each image. For certain classes such as *hovering over accelerator*, we placed two Gaussian peaks in close locality to cover the larger spatial extent of such activities. The resulting attention masks for each class are depicted in Figure 8.2.

For our classification model, we initialize the Squeezenet v1.1 model with Imagenet pretrained weights. The training is carried out in two stages for a total of 30 epochs. Standard mini batch Stochastic Gradient Descent (SGD) with a batch size of 64 is used to train the network. We use a learning rate of 0.0005 with a momentum equal to 0.9, and a weight decay term to reduce model complexity.

The network is trained for the first 15 epochs using the $L^{stage-1}$ loss (Equation 8.3), and then using the $L^{stage-2}$ loss for the remaining epochs. The hyperparameters λ_{FSA} and λ^{reg}_{FSA} are determined through extensive cross-validation, the results of which are shown in Figure 8.3. Our final choices for hyperparameters λ_{FSA} and λ^{reg}_{FSA} were 10 and 0.2 respectively. The qualitative effect of our two stage training approach is illustrated in Figure 8.4 for further clarity. In the depicted examples from the training and validation sets, we observe that during the first stage of training, the network learns to attend to large regions corresponding to various possible regions

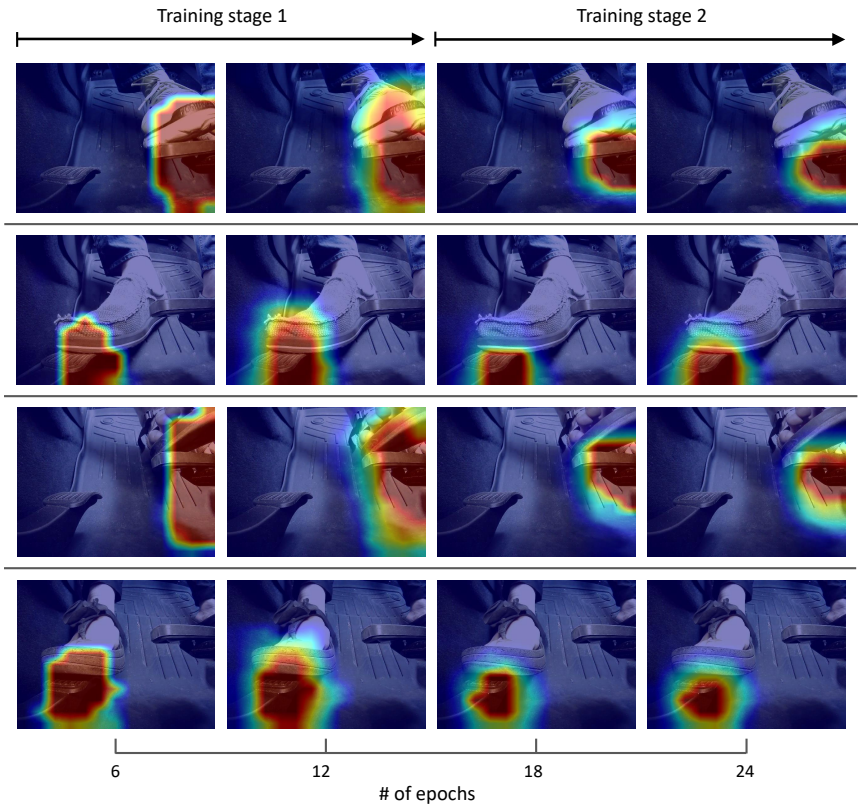


Figure 8.4: Class Activation Maps (CAMs) for the correct output class as a function of the number of training epochs. Each row is a different example.

of activity, while the region of attention gradually contracts to the specific region of activity corresponding to the given input image in the second stage of training. In particular, we observe that the attention contracts to the location where the foot hits the pedal, for different locations of the foot and pedal.

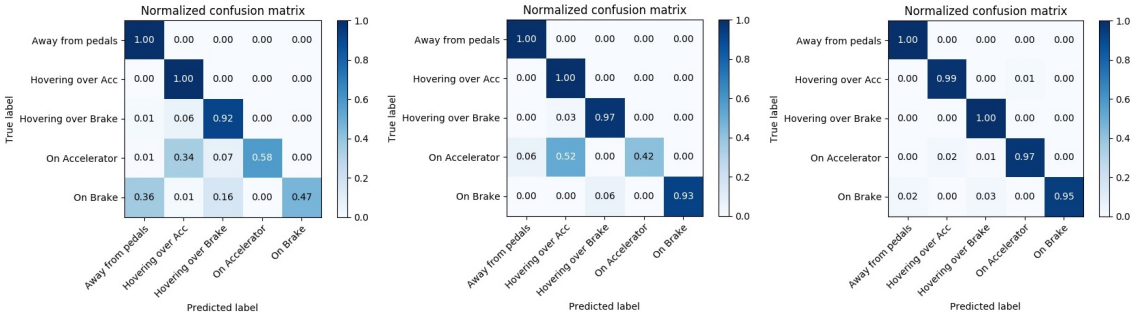
8.4 Experimental Evaluation

8.4.1 Dataset

To train and evaluate our proposed model and its variants, we collect a diverse dataset of images capturing driver foot activities. This data was collected during naturalistic drives, with

Table 8.1: Details of the train-val-test split used for the experiments.

Split	Number of unique drivers	Number of images
Train	7	19,385
Validation	1	1,867
Test	3	7,698



(a) CE loss (b) (CE + MSE) loss (c) (CE + FSA) loss

Figure 8.5: Confusion matrices on the test split for networks trained using different losses.

many different drivers as subjects. Details of our complete dataset and the *train*, *validation*, and *test* splits are listed in Table 8.1. In particular, we ensure that no subjects overlap between the three splits so as to test the cross-subject generalization of our models. We also try our best to keep the class distributions similar across the three splits.

8.4.2 Results

We first compare the overall classification accuracies of different variants of our Squeezenet v1.1 model on the test split (see Table 8.2). All variants of Squeezenet v1.1 were initialized with pretrained Imagenet weights before training. First, we have the model trained only using the standard cross entropy classification loss. This model produces a reasonable accuracy of 85.99% and provides a strong baseline to compare our proposed approach against. Next, we compare different versions of our model that make use of the predefined attention masks during training,

Table 8.2: Classification accuracies for different model variants on the test split.

Model	Loss	Accuracy (%)
SqueezeNet v1.1	**CE**	85.99
SqueezeNet v1.1	**CE+MSE**	89.67
SqueezeNet v1.1	**CE + FSA (stage 1 only)**	92.30
SqueezeNet v1.1	**CE + FSA (stage 2 only)**	85.49
SqueezeNet v1.1	**CE + FSA (both stages)**	**97.49**
SqueezeNet v1.1 w/ FC output layer	**CE**	63.31
AlexNet v1.1 w/ FC output layer	**CE**	60.99
VGG16 v1.1 w/ FC output layer	**CE**	67.03

CE: Cross Entropy loss MSE: Mean Squared Error loss
FSA: Forced Spatial Attention loss

but differ in the losses they use to force spatial attention. It is observed that simply incorporating domain specific spatial knowledge leads to an improvement in overall accuracy, irrespective of the specific choice of the loss function. Adding a simple MSE loss (i.e. using only the first term from the loss defined in Equation 8.5) between the CAMs and their corresponding attention masks leads to a modest improvement over the baseline. We also observe that using either one of the two stages of the FSA loss also improves the overall accuracy, but not as much as when they are used in conjunction over two stages. Our proposed two stage FSA loss leads to the best overall accuracy of 97.49%- a significant improvement over the baseline. Finally, we also provide accuracies for Squeezenet v1.1, AlexNet [242], and VGG16 [43] with output FC layers. Even though an FC layer by nature can produce location specific features, we observe that the large size of the models and limited size of the dataset make it a bad fit for the task at hand.

We can also gather some insights about the performance of each variant by looking at

both their confusion matrices on the test split (Figure 8.5) and their CAMs for different input images (Figure 8.6). Although the baseline model results in a reasonable overall accuracy, it fails to learn the true concept of each class and overfits to background information. This is illustrated by its confusion between classes that are very different to one another and its mostly uniform CAMs. Next, we observe that incorporating domain specific spatial information using predefined attention masks and an MSE loss makes the model better and more robust, with much more informative CAMs. However, we can also see activation leakage between classes (CAMs with high activations in the same region), resulting in confusion between similar classes. Finally, we see that adding a regularizing term as in the two stage FSA loss resolves these issues. It not only reduces the confusion between similar classes, but also produces more confident outputs as illustrated by the corresponding CAMs.

The failure cases we generally observe are at the boundaries of the *hovering over* ___ and the *on* ___ classes, especially when the foot is hovering very close to one of the pedals. We find this to be acceptable because of the relative difficulty that humans have in confidently labelling these examples.

8.5 Chapter Summary

In this chapter, we introduce a simple approach to solve image classification tasks where the output classes are tied to relative spatial locations of objects in the image. We do so by augmenting the standard classification loss with a Forced Spatial Attention (FSA) loss that compels the network to attend to specific regions in the image associated to the desired output class. The FSA loss function provides a convenient way to incorporate spatial priors that are known for a certain task, thereby improving robustness and generalization without requiring additional labels. The benefits of our approach are demonstrated for the driver foot activity classification task, where we improve the baseline accuracy by approximately 13% without

modifying the network architecture. Such an improvement is especially valuable for ensuring the robustness and reliability of downstream safety critical tasks such as driver vigilance estimation and take-over time prediction.

8.6 Acknowledgements

Chapter 8, in full, is a reprint of the material as it appears in the IEEE Conference on Computer Vision - Workshop on Assistive Computer Vision and Robotics (2019), by Akshay Rangesh, and Mohan M. Trivedi. The dissertation author was the primary investigator and author of this paper.

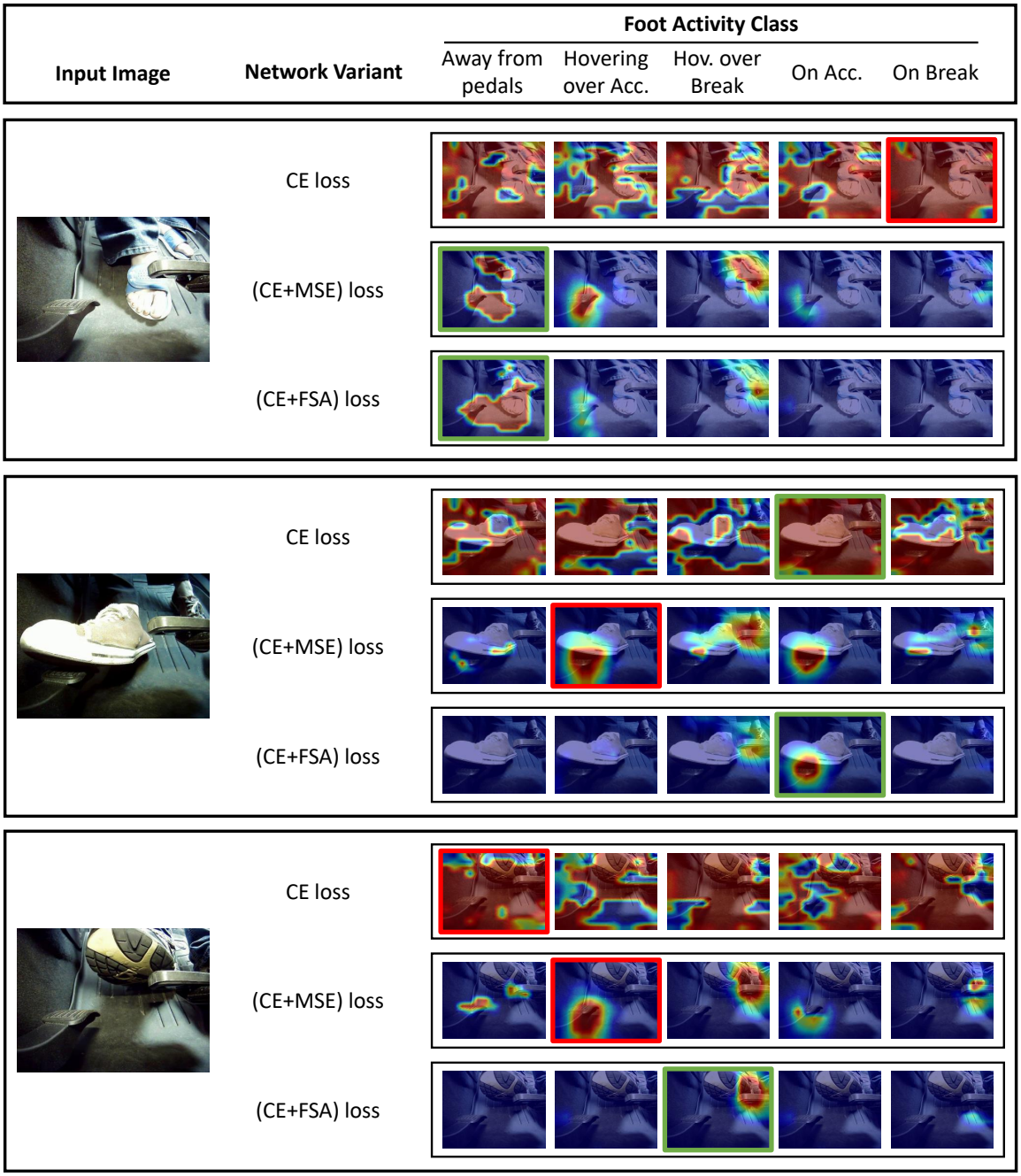


Figure 8.6: Class Activation Maps (CAMs) resulting from networks trained with different loss functions. The three major rows correspond to three different input images. The green boxes shows the ground truth class labels while the red boxes shows if the network made an incorrect prediction.

Part IV

Looking-In & Looking-Out

Chapter 9

Autonomous Vehicles that Alert Humans to Take-Over Controls

With increasing automation in passenger vehicles, the study of safe and smooth occupant-vehicle interaction and control transitions is key. In this study, we focus on the development of contextual, semantically meaningful representations of the driver state, which can then be used to determine the appropriate timing and conditions for transfer of control between driver and vehicle. To this end, we conduct a large-scale real-world controlled data study where participants are instructed to take-over control from an autonomous agent under different driving conditions while engaged in a variety of distracting activities. These take-over events are captured using multiple driver-facing cameras, which when labelled result in a dataset of control transitions and their corresponding take-over times (TOTs). We then develop and train TOT models that operate sequentially on mid to high-level features produced by computer vision algorithms operating on different driver-facing camera views. The proposed TOT model produces continuous predictions of take-over times without delay, and shows promising qualitative and quantitative results in complex real-world scenarios.

9.1 Introduction

Control transitions in the context of partially automated vehicles entail the transfer of vehicle controls from the autonomous agent to the human driver and vice versa. In this study, we

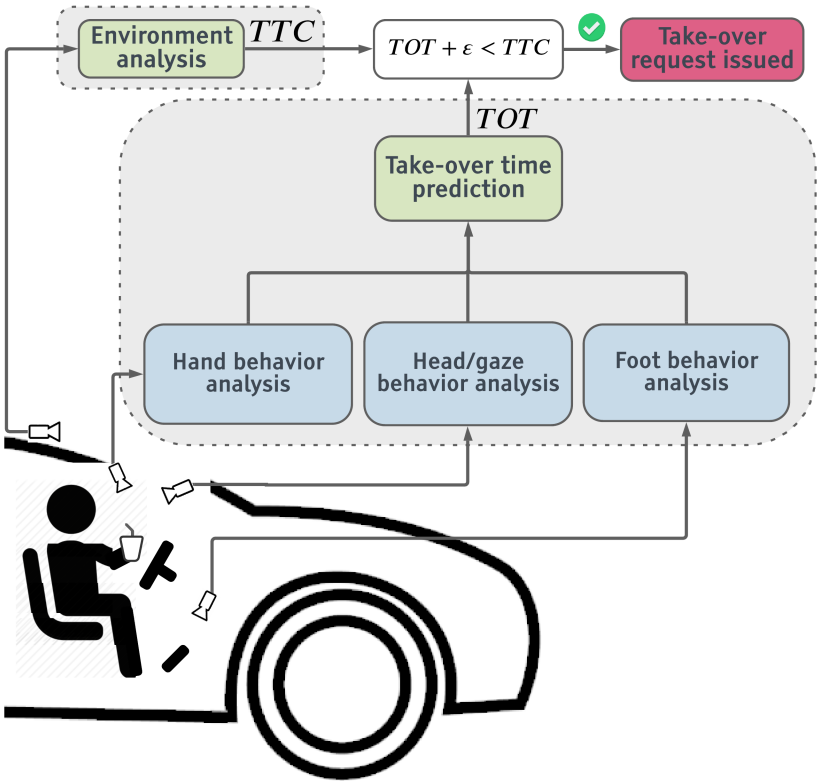


Figure 9.1: Illustration of the proposed take-over time (TOT) prediction approach using driver-facing cameras. The governing logic in an autonomous vehicle must understand and model driver behavior continuously in order to ensure safe and smooth transfer of control between automation and humans.

primarily focus on transitions from the autonomous agent to the human driver - as these scenarios are inherently risky and require timely human intervention to avoid collision. In describing such control transitions, we make use of the take-over time (TOT) metric, defined as the interval of time between a take-over request (TOR) being issued and the assuming of human control. The take-over request could be an auditory/visual/tactile cue used to indicate to the driver that their intervention is immediately needed. Due to the complexity of human attention, we define the assumption of control as the completion of the following three behaviors:

1. Hands-on-wheel: hand(s) return to the vehicle's steering control.

2. Foot-on-pedal: foot returns (from floorboard or hovering) to make contact with any driving

pedal.

3. Eyes-on-road: gaze is directed forward, toward the active driving scene.

We make the assumption that these three cues occurring simultaneously are sufficient to consider the driver both attentive to the scene and in control of the vehicle.

As depicted in Fig. 9.1, the transition of control from an autonomous agent to the human driver should be a function of both the surrounding scene and the state of the driver. The surrounding scene can be concisely expressed using a metric such as time-to-collision (TTC), whereas the state of the driver can be captured by the predicted TOT. Combined, this forms a criterion for safe control transitions:

$$TOT + \varepsilon < TTC, \tag{9.1}$$

where ε is a marginal allowance for human maneuvering. A system that takes the state of the driver into account can decide between handing over control if the driver is ready, versus coming to a safe and smooth halt if not.

While there are many approaches to predict TTC, TOT prediction (especially in the real world) remains relatively unexplored. This study is the first to train TOT prediction models using in-cabin sensors that monitor human subjects in real (non-simulated) driving scenarios. By using real-world data, we train and develop a sequential neural network based on LSTMs to accurately predict TOTs for drivers engaged in a variety of secondary tasks. In conjunction with a TTC model, the proposed TOT prediction model could allow the governing logic to make informed decisions during control transitions.

9.2 Related Research

As stated in [46], "...the design of intelligent driver-assistance systems, especially those that activate controls of the car to prevent accidents, requires an accurate understanding of human behavior as well as modeling of human-vehicle interactions, driver activities while behind the wheel, and predictable human intent." Without understanding and cooperating with the driver state, take-over requests may be issued with inadequate time to execute manual take-over. A comprehensive, cross-disciplinary survey of driver attention modeling, [243] introduces a progression of approaches toward understanding in-cabin factors which are immediately relevant in the context of control transitions from automated to manual control.

In most previous studies, control transitions in autonomous vehicles have been framed around particular moments expected during take-over events. A transition begins with a take-over request, to which the driver will react. At some point, the driver will have established control, traditionally evidenced by (and also evaluated by) the quality of performance metrics such as braking patterns, speed, and steering wheel angle.

Many studies in the realm of human factors and human-robot interaction have sought to understand and improve autonomous vehicle control transitions, evaluating the effects of take-over request methodologies, sample demographics, and secondary non-driving-related tasks. Due to possible catastrophic failure cases, video simulation is used in lieu of real-world driving for many of these studies. In [244], researchers analyzed the take-over time of participants in a Highly Automated Driving simulator. Participants may have been assigned a secondary task, such as reading a magazine, before being presented with an audio-visual take-over request, shown to correspond to significantly longer control transition times. [245] found that when take-over requests were correlated with traffic scene cues such as vehicle speed and traffic density, the anticipatory scene information led to quicker participant reactions to take-over requests. They make a distinction between driver reaction and driver control, the latter (measured as visual

attention and lane deviation) showing no significant change whether take-over requests were scene-informed or sporadic. Looking inside the cabin, studies such as [246–249] explore the effects of demographics, cognitive load of secondary tasks, and nature of take-over request on driver response time and quality. While in-cabin activity has been classified using machine learning approaches, as in [221, 250–252], there is additional importance in using this information to assess driver readiness and predict take-over times.

Most indicators studied in previous works are non-predictive by nature, as these take-over time or quality signals appear following the instance of take-over. Rather than measuring human reaction time in simulated data, in our work, we explore the gray area that exists between initial reaction and control by analyzing human behavior in the frames between the issued take-over and brake or steering action. Contrasting previous methods, we show that by using non-intrusive visual modalities, it is possible to observe a foot at-the-ready to brake without measuring 10% pedal depression, or ready hands-on-the-wheel without a 2-degree steering jitter. This ensemble of visual driver readiness features, developed in [253] has been shown to closely match human expectation of vehicle control in [158], so it is a natural extension to use these cues for the purpose of take-over time prediction.

Further, our work is unique in its use of real-world driving data, as opposed to the more widely-studied video simulations. While studies such as [254] have replaced video simulation with Wizard-of-Oz driving and others such as [255] have explored secondary activities while driving or supervising, our work is first to feature a vehicle in exclusive, live control of the driver and AV driver-assistance system during control transitions. Additionally, our data is representative of a wide range of secondary tasks, which vary in both duration and cognitive load. By combining the visual readiness features extracted from naturalistic driving data, we show an effective data-driven approach for predicting take-over time.

9.3 Dataset & Labels

9.3.1 Controlled Data Study (CDS)

To capture a diverse set of real-world take-overs, we conduct a large-scale study under controlled conditions. More specifically, we enlist a representative population of 89 subjects to drive a Tesla Model S testbed mounted with three driver-facing cameras that capture the gaze, hand, and foot activity of the driver. In this controlled data study (CDS), we required each subject to drive the testbed for approximately an hour in a pre-determined section of the roadway, under controlled traffic conditions. During the drive, each test subject is asked to undertake a variety of distracting secondary activities while the autopilot is engaged, following which an auditory take-over request (TOR) is issued at random intervals. This initiates the control transition during which the driver is instructed to take control of the vehicle and resume the drive. Each such transition corresponds to one take-over event, and our CDS produces 1,375 take-over events in total.

9.3.2 Labelling Procedure

Automated Video Segmentation

Each driving session is first segmented into 30 second windows surrounding take-over events, consisting of 20 seconds prior to the take-over request (TOR) and 10 seconds after the take-over event. The times at which these TORs were initiated are known and help us locate all take-over events from a driving session.

Event Annotations

For each 30 second clip corresponding to a take-over event, we annotate the following:

1. **Eyes-on-road:** We mark the first frame after the take-over request when the driver's eyes are on the road.

2. **Hands-on-wheel:** We mark the first frame after the take-over request when the driver's hands are on the wheel.

3. **Foot-on-pedal:** We mark the first frame after the take-over event when the driver's foot is on the pedals.

4. **Driver secondary activity:** We label the secondary activity being performed by the driver during each take-over event. We assign one of 8 possible activity labels: (1) No secondary activity (i.e. attentive), (2) talking to co-passenger, (3) looking at lap/eyes closed, (4) texting, (5) phone call, (6) using infotainment unit, (7) counting change, (8) reading a book or magazine.

9.3.3 Take-Over Time Statistics

Of the 1,375 control transitions obtained from the CDS, 308 correspond to performing no secondary activity (attentive), 182 correspond to the driver talking to co-passengers, 85 correspond to drivers with their eyes closed or looking at their lap, 262 correspond to the driver texting, 42 correspond to the driver being on a phone call, 299 correspond to the driver interacting with the infotainment unit, 97 correspond to the driver counting coins, and finally 100 correspond to the driver reading a book or magazine.

Figure 9.2 shows the average times corresponding to eyes-on-road, hands-on-wheel and foot-on-pedal for each of the 8 secondary activities. It also shows the overall take-over time, defined as the maximum of the three markers for each event. We note that texting, phone-calls, counting change and reading correspond to longer average take-over times, as compared to talking to the co-passenger or using the infotainment unit, which can be reasonably expected. Counter to intuition, the 'eyes closed/looking at lap' activity has low take-over times. This is mainly because the drivers are merely pretending to be asleep, since actual sleep could not have been achieved given the duration and nature of each trial. We also note that the 'hands-on-wheel'

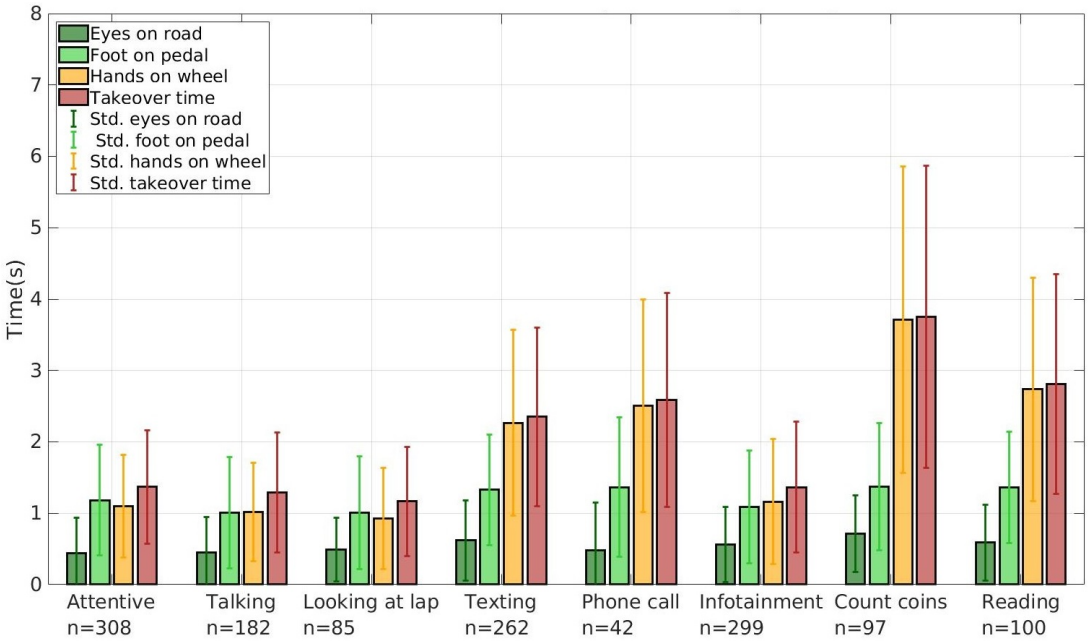


Figure 9.2: Take-over time statistics from the CDS. We plot the mean values of the different take-over related event timings for each secondary activity. The error bars depict one standard deviation in each direction.

event seems to take much longer on average, as compared to eyes-on-road or foot-on-pedal. This reinforces the need for driver hand analysis. Finally, we note that for the more distracting secondary activities (reading, texting, phone calls, counting change), even the foot-on-pedal times are longer compared to the other secondary activities, although the secondary activities do not involve the driver's feet. Thus, there seems to be a delay corresponding to the driver shifting attention from secondary activity to the primary activity of driving.

9.3.4 Take-Over Time Dataset & Augmentation Strategies

Take-over time data is very limited and expensive to capture and label. This is illustrated by the size of the CDS training set (1065 total take-overs) despite the scale and duration of the study. This introduces challenges during the training of machine learning models (especially neural networks) for prediction, as these models typically require tens of thousands of training samples. Care must also be taken to avoid overfitting, as this is more prevalent in the limited data

regime. To address these issues, we propose a new data augmentation scheme to increase the number of samples in the dataset by an order of magnitude - with the hope of reducing overfitting.

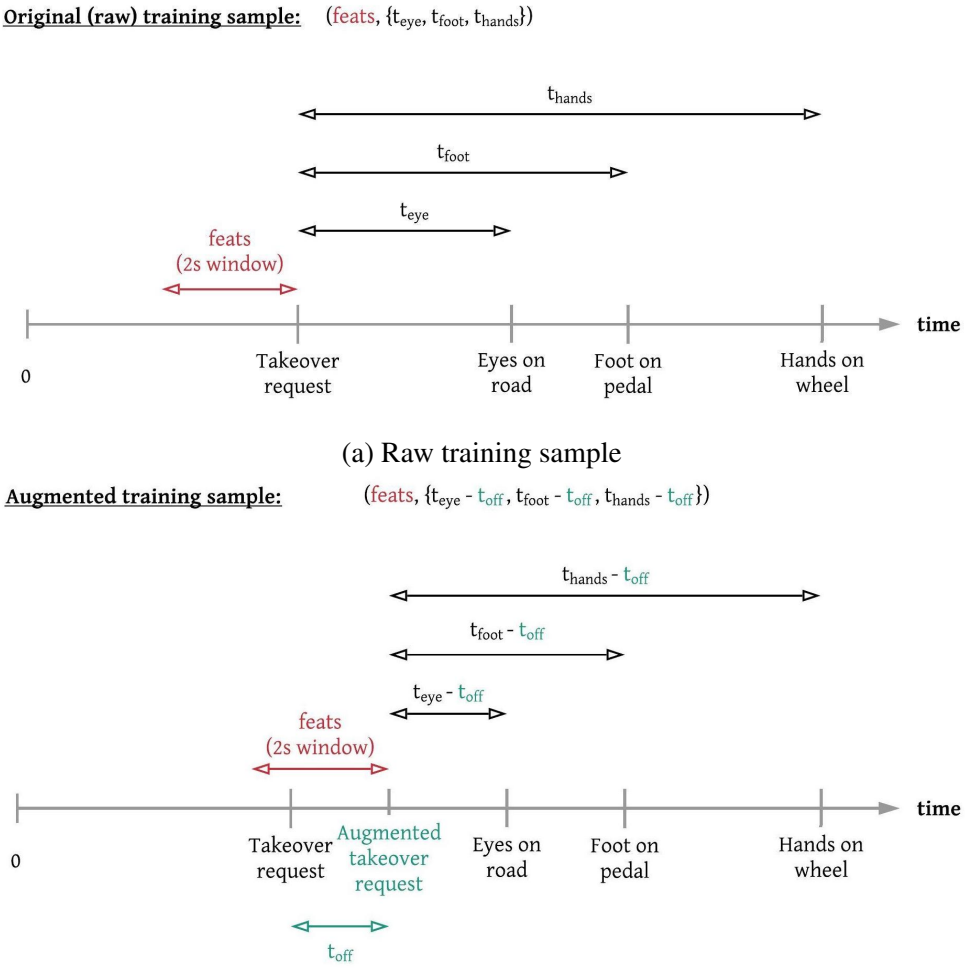


Figure 9.3: Illustration of proposed TOT dataset augmentation scheme.

This process is illustrated in Figure 9.3. A raw sample of the TOT prediction dataset is illustrated in Figure 9.3a. Figure 9.3b depicts how an augmented sample for TOT prediction can be created from the original raw sample by introducing an offset to the original take-over request (TOR). Essentially, if

$$(feats(t_{TOR} - 2 : t_{TOR}), \{t_{eye}, t_{foot}, t_{hands}\}) \tag{9.2}$$

Table 9.1: Size of take-over time prediction training set before and after augmentation.

Dataset	Number of samples
CDS (R[1])	1065
CDS (A[2])	47,461

[1] raw dataset [2] augmented dataset

is a raw sample from the TOT prediction dataset, an augmented sample can be denoted as:

$$(feats(t_{TOR}+t_{off}-2:t_{TOR}+t_{off}),\{t_{eye}-t_{off},t_{foot}-t_{off},t_{hands}-t_{off}\}), \tag{9.3}$$

where $t_{off} \in (0, max\{t_{eye}, t_{foot}, t_{hands}\}]$ is the offset introduced to the actual TOR. Each offset t_{off} (and thus augmented take-over request) generates one additional sample. This augmented sample is valid because the driver has already received a TOR, and thus proceeds to behave in the same manner as if they had just received a TOR (assuming instant reaction times).

We gain many samples by moving the augmented take-over request frame-by-frame from the actual take-over request until the take-over is complete. More importantly, we get many diverse scenarios in our dataset; many of which would not have been captured otherwise. For example, even though an original sample may have sought to capture the time from initial "eyes closed, hands in lap" to final "eyes forward, hands on wheel", this augmentation scheme asserts we have additional implicit data for "eyes halfway open, hands midway to steering wheel" - as well as every other intermediate state of the driver.

With the proposed augmentation scheme, we get a dataset vastly larger in size as depicted in Table 9.1. Our proposed augmentation strategy increases the size of the dataset by an order of magnitude. Note that this augmentation scheme is only applied to the training split. The validation and test splits are left untouched for accurate evaluation.

9.4 Take-Over Time Prediction Model

9.4.1 Overview

It is important to preserve both the diverse and sequential nature of all features related to driver behavior while designing a holistic take-over time (TOT) prediction framework. High level tasks such as TOT prediction are influenced by different driver behaviors, both in the short and medium to long term. With this in mind, we propose the overall framework illustrated in Figure 9.4 comprising of 4 unique blocks (or stages). We provide a brief description of each block below:

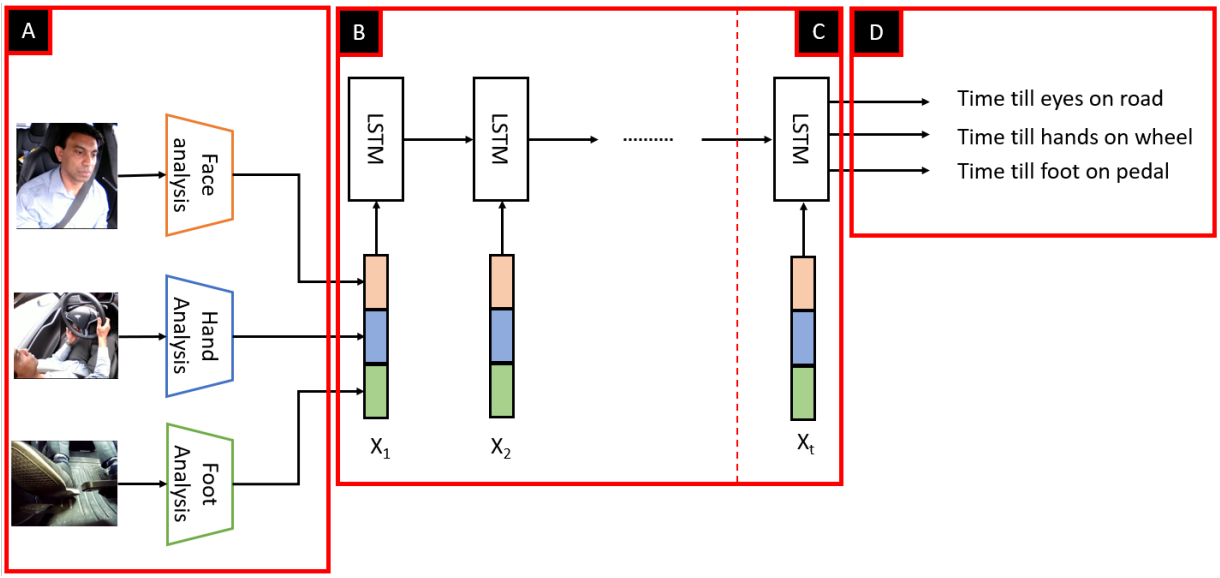


Figure 9.4: Algorithm workflow for the overall *Take-over Time* prediction framework. Each of the 4 blocks (A-D) above are described in the text.

Block A

This block comprises of the *low-level* models that analyze specific driver behaviors related to the face [156, 256, 257], hands [258] and foot [259]. Each of these models run as independent processes, on separate threads of execution. These algorithms operate on their respective camera feeds and produce mid to high-level features like gaze direction, hand locations, handheld objects, foot locations etc. on a per-frame basis.

Block B

This block comprises of a sequential model responsible for learning features useful for TOT prediction. This downstream model is a recurrent LSTM network, capable of learning both short and long term temporal patterns. This model is run independently, grabbing the latest available outputs from Block A, processing them, and finally updating its hidden state.

Block C

This block is usually considered part of the previous block, but is highlighted separately for clarity. Block C depicts the latest operation performed by LSTM model described in Block B, and produces the set of outputs for the current (latest) timestep.

Block D

This block lists the post-processed outputs produced by the LSTM model. These outputs are produced at each timestep, not just at the end of a sequence. This makes it possible to get continuous predictions of take-over times for every frame.

We detail the take-over time model introduced in blocks B, C, and D in the following subsection.

9.4.2 Take-Over Time Model Architecture

Instead of directly feeding the input features produced by block A to a recurrent network, we propose to parallely process the input features using separate LSTMs to predict the three times of interest. The reasoning behind this is to accommodate different hidden state update rates for different driver behaviors, for example – eyes-on-road behavior is generally faster (short term) than hands-on-wheel behavior (mid/long term). Having multiple independent LSTMs allows each one to update at different rates, thereby capturing short/mid/long term behaviours separately.

As depicted in Figure 9.5, the input features are first transformed using a fully-connected (FC) layer (plus non-linearity), which is then fed to an LSTM at each timestep. The LSTM layer receives the transformed input features at each timestep and updates its hidden state. In

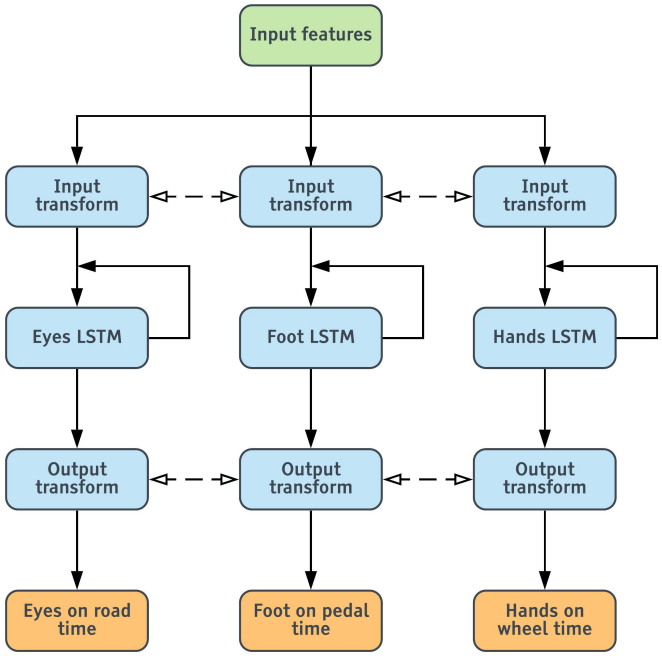


Figure 9.5: The proposed independent LSTMs model architecture.

all our experiments, we choose a 2 second window of features as input to our models. After 2 seconds worth of inputs and updates, the hidden state of the LSTM after the latest timestep is passed through an output transformation (FC layer plus non-linearity) to predict the three times of interest. Although each branch has its own LSTM cell, the input and output transformations are still shared between the three LSTMs as the feature inputs to the three branches are the same. This tends to reduce overfitting based on our experiments.

We apply a simple $L1$ loss to train this network. Let o_e, o_f, and o_h be the outputs produced by the model. Assuming t_e, t_f, and t_h are the target eyes-on-road time, foot-on-pedal time, and hands-on-wheel time respectively, the total loss is:

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}|t_e^i - o_e^i| + \frac{1}{N}\sum_{i=1}^{N}|t_f^i - o_f^i| + \frac{1}{N}\sum_{i=1}^{N}|t_h^i - o_h^i|. \tag{9.4}$$

The entire model is trained using an Adam optimizer with a learning rate of 0.001 for 10 epochs.

Table 9.2: Prediction errors for different times of interest on the CDS validation set when trained on a variety of datasets.

Training dataset (s)	Eyes-on-road MAE (s)	Foot-on-pedal MAE (s)	Hands-on-wheel MAE (s)	Take-over time MAE (s)
CDS (R)	0.3676	0.5435	0.8285	0.8576
CDS (A)	**0.3266**	**0.4841**	**0.7113**	**0.7912**

[1] raw dataset [2] augmented dataset

9.5 Experiments & Evaluation

9.5.1 Ablation Experiments

In this subsection, we go through several ablation experiments conducted on the validation set, to evaluate and contrast the importance of various design choices, feature combinations, and training procedures.

First, we conduct an experiment to assess the effects of our proposed data augmentation scheme. To do this, we train the proposed TOT model on the raw and augmented datasets respectively. We use mean absolute errors (MAEs) for each time of interest and the overall TOT as metrics for comparison. Results from these experiments are presented in Table 9.2.

From Table 9.2, we notice that training on the augmented dataset (as proposed in Section 9.3.4) consistently and considerably improves performance across all metrics. We believe that doing so prevents overfitting, provides regularization, smooths the outputs of model, and adds new training samples that would be cumbersome or impossible to capture.

Next, we conduct an experiment to assess the relative importance of different input features and their combinations. To isolate effects from features, we train the same independent LSTMs model with different input feature combinations. We use individual and TOT mean absolute errors (MAEs) as metrics for comparison. Table 9.3 contains results from this experiment.

We notice that hand features are the most important, followed by foot and gaze features respectively. This might be because gaze dynamics are relatively predictable during take-overs as

Table 9.3: Prediction errors for different times of interest on the CDS validation set for a variety of feature combinations.

Features					Eyes on road MAE (s)	Foot on pedal MAE (s)	Hands on wheel MAE (s)	Take-over time MAE (s)
F[1]	G[2]	H[3]	S[4]	O[5]				
✓					0.3587	0.5018	0.8599	0.8856
	✓				0.3332	0.5690	0.8411	0.8837
		✓			0.3729	0.5384	0.7565	0.9012
		✓	✓		0.3783	0.5109	0.7369	0.8315
		✓		✓	0.3702	0.4973	0.7177	0.8621
		✓	✓	✓	0.3747	0.4857	0.7141	0.7983
	✓	✓		✓	0.3244	0.5220	0.7163	0.7920
	✓	✓	✓	✓	0.3299	0.5124	0.7215	0.7921
✓	✓	✓	✓		**0.3222**	0.5059	0.7870	0.8475
✓	✓	✓		✓	0.3277	0.5074	0.7144	0.7918
✓	✓	✓	✓	✓	0.3266	**0.4841**	**0.7113**	**0.7912**

[1] **foot features:** probabilities for all 5 foot activities, namely - away from pedal, on break, on gas, hovering over break, and hovering over gas
[2] **gaze features:** probabilities for all 8 gaze zones, namely - front, speedometer, rearview, left mirror, right mirror, over the shoulder, infotainment, and eyes closed/looking down
[3] **hand features:** probabilities for all 6 hand activities (left and right hand), namely - on lap, in air, hovering over steering wheel, on steering wheel, cupholder, and infotainment
[4] **stereo hand features:** distance of left and right hand from the steering wheel
[5] **hand-object features:** probabilities for all 7 hand object categories (left and right hand), namely - no object, cellphone, tablet/iPad, food, beverage, reading, and others

the first thing drivers tend to do is look at the road to assess the situation, leading to less variance in eyes-on-road behavior. Next, we notice that adding more informative hand feature like 3D distances to the steering wheel and hand-object information improves the performance further. Hand-objects in particular seem to vastly improve the performance in general. This makes sense as hand-objects provide the strongest cue regarding the secondary activities of drivers. Adding stereo (3D) hand features improves the results, but not by much. Adding foot features also tends to reduce the errors considerably, illustrating the importance of having a foot-facing camera.

In conclusion, one could get close to peak performance by utilizing 3 cameras - 1 foot, 1 hand, and 1 face camera respectively. Hand features (including hand-object features) are most informative, followed by foot and gaze features respectively.

Table 9.4: Prediction errors for different models on the take-over time test set.

Model type (s)	Eyes-on-road MAE (s)	Foot-on-pedal MAE (s)	Hands-on-wheel MAE (s)	Take-over time MAE (s)
LSTM[1]	**0.2365**	0.5007	0.8710	0.9457
ID LSTMs[2]	0.2497	**0.4650**	**0.8055**	**0.9144**
ID LSTMs (75%[3])	0.2557	0.5013	0.8474	0.9779
ID LSTMs (90%[4])	0.2514	0.4851	0.8482	0.9424

[1] baseline LSTM model [2] Independent LSTMs [3] 75% of the dataset used for training
[4] 90% of the dataset used for training

9.5.2 Results & Analysis

In this subsection, we provide qualitative and quantitative results of our proposed TOT prediction model to enhance our understanding of their workings and to gain more insights. All results in this subsection are reported on a test set separate from our training and validation splits.

Quantitative Results

First, we present quantitative error metrics on the test set for the proposed model in Table 9.4. To measure the benefits of the independent LSTMs approach, we compare our model to a baseline model with a single LSTM. We observe that the independent LSTMs model outperforms the baseline model across most metrics, and falls slightly short of the baseline for eyes-on-road MAE. We also notice that hands-on-wheel MAEs are usually the largest owing to large variance in hand behaviors and large absolute values associated with hands-on-wheel times.

We also show results for ID LSTMs when trained on 75% and 90% of available training data. This helps us gauge the expected improvement in performance as more training data is added. Based on the numbers presented in Table 9.4, we can expect meager improvements as more data is added. This indicates a case of diminishing returns.

Prediction Errors by Secondary Activity

Since our CDS dataset is comprised of a variety of distracting secondary activities, it is useful to conduct error analysis for each activity separately. Figure 9.6 shows the MAE values for each secondary activity in the CDS test set. We note that the MAE values have a similar

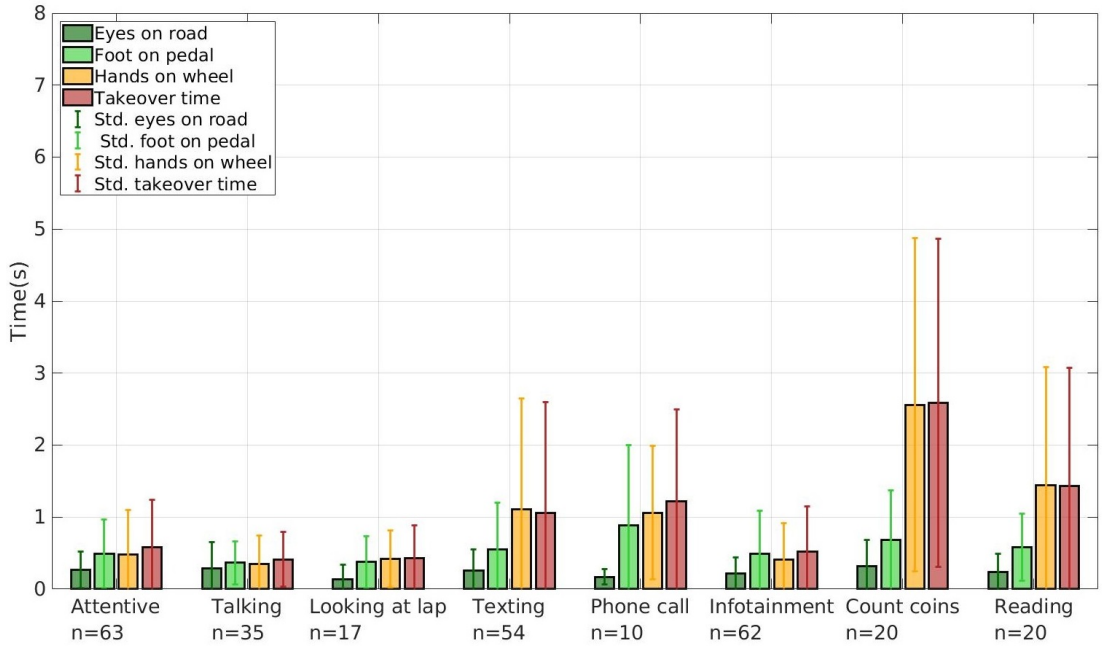


Figure 9.6: Mean absolute error (MAE) per secondary activity in the CDS dataset. The error bars depict one standard deviation in each direction.

trend as the take-over time statistics (Figure 9.2) i.e., activities with larger TOT values have larger MAEs owing to larger variance. However, the errors are typically much smaller than the average take-over times for each secondary activity - indicating small relative errors across all activities.

Finally, Figure 9.7 shows the average values predicted by the model along with 95% confidence intervals for each of the 8 secondary activities in the CDS, as a function of time from the TOR. We note that for each of the 8 activities, the predicted TOT has a relatively higher value prior to the TOR, while the driver is performing the secondary activity. This is followed by a transition phase, corresponding to the actual take-over, followed by a relatively low value, which corresponds to the completion of the take-over. The secondary activities which have longer take-over times tend to have a more gradual transition phase as compared to secondary activities with shorter take-over times.

Qualitative Results and Examples

Finally, we also provide qualitative examples of predictions made by the ID-LSTMs model for all 8 secondary activities (Figure 9.8). Each example shows the 5 camera views at the

instant where the TOR is issued. The true values of the 3 times (eyes-on-road, hands-on-wheel, foot-on-pedal) are shown in the plot as solid circular markers, while the corresponding predicted values are shown as hollow circular markers of the same color. We show the the ground truth take-over time as a solid purple line and the predicted take-over time as a dashed purple line. We note that the model accurately predicts short take-over times when the driver is attentive or operating the infotainment unit, and longer take-over times for activities that impose a higher cognitive load such as texting or reading a magazine.

9.6 Chapter Summary

This chapter presents one of the largest real-world studies on take-over time (TOT) prediction and control transitions in general. To predict TOTs in the real world, we conduct a controlled data study (CDS) with 89 subjects, each performing multiple take-overs while engaged in a variety of distracting secondary activities. The CDS was used to capture real-world take-over events and corresponding take-over times for a diverse pool of drivers. This dataset of take-over times, along with our proposed data augmentation scheme was then used to train downstream models for TOT prediction that operate sequentially on mid to high-level features produced by computer vision algorithms operating on different driver-facing camera views. In addition to the TOT prediction model, we also provide results from various ablation studies to compare and contrast the effects of different model architectures, feature combinations, data augmentation and other design choices. We believe that this study outlines the sensors, datasets, algorithms and models that can truly benefit the intermediate levels of automation by accurately assessing driver behavior and predicting take-over times - both of which can then be used to smoothly transfer control between humans and automation.

9.7 Acknowledgements

Chapter 9, in part, is a reprint of the material as it appears in the IEEE Conference on Intelligent Transportation Systems (2021), by Akshay Rangesh, Nachiket Deo, Ross Greer, Pujitha Gunaratne, and Mohan M. Trivedi. The dissertation author was one of the primary investigators and authors of this paper.

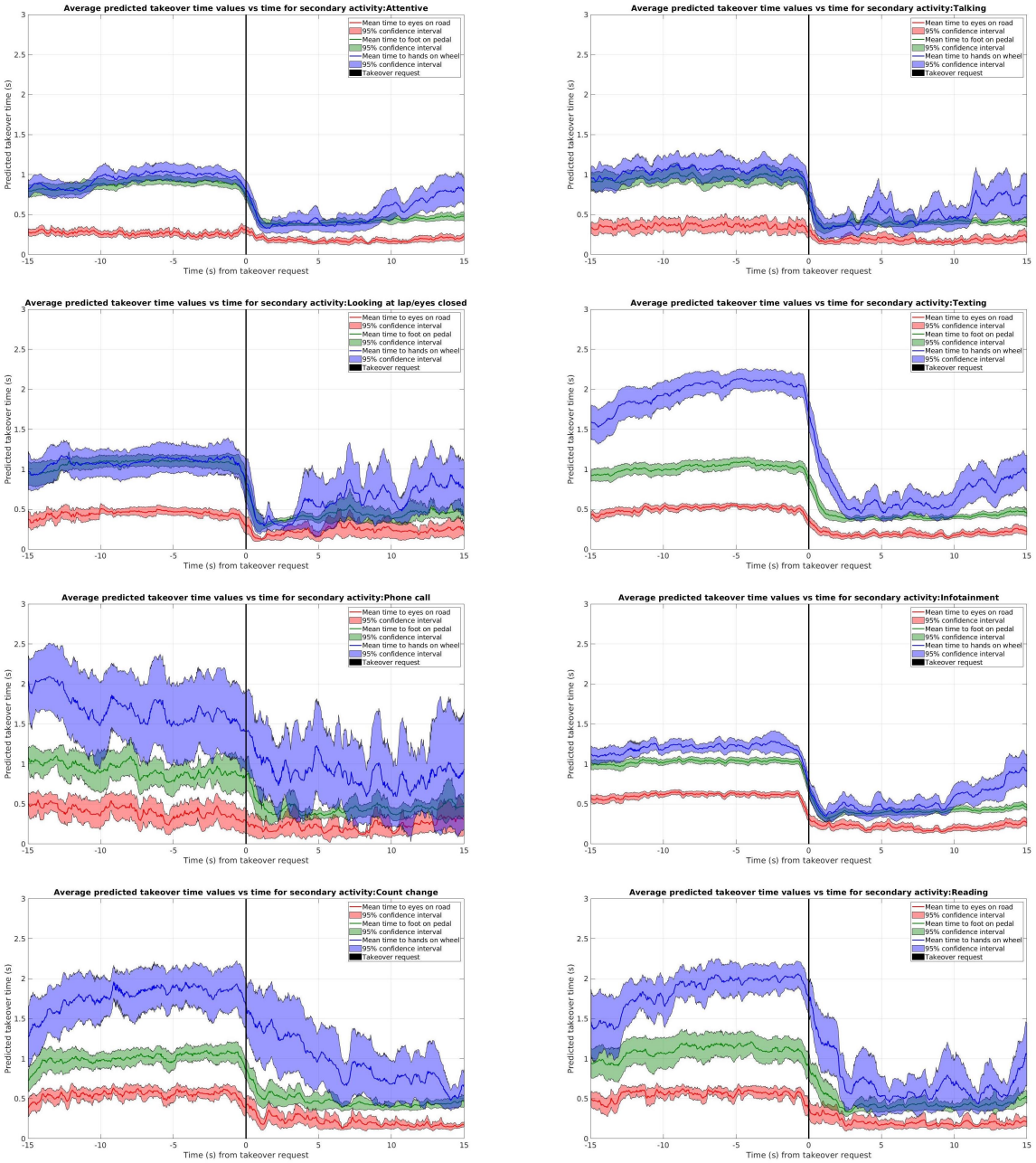


Figure 9.7: Predicted take-over times as a function of time from take-over request.

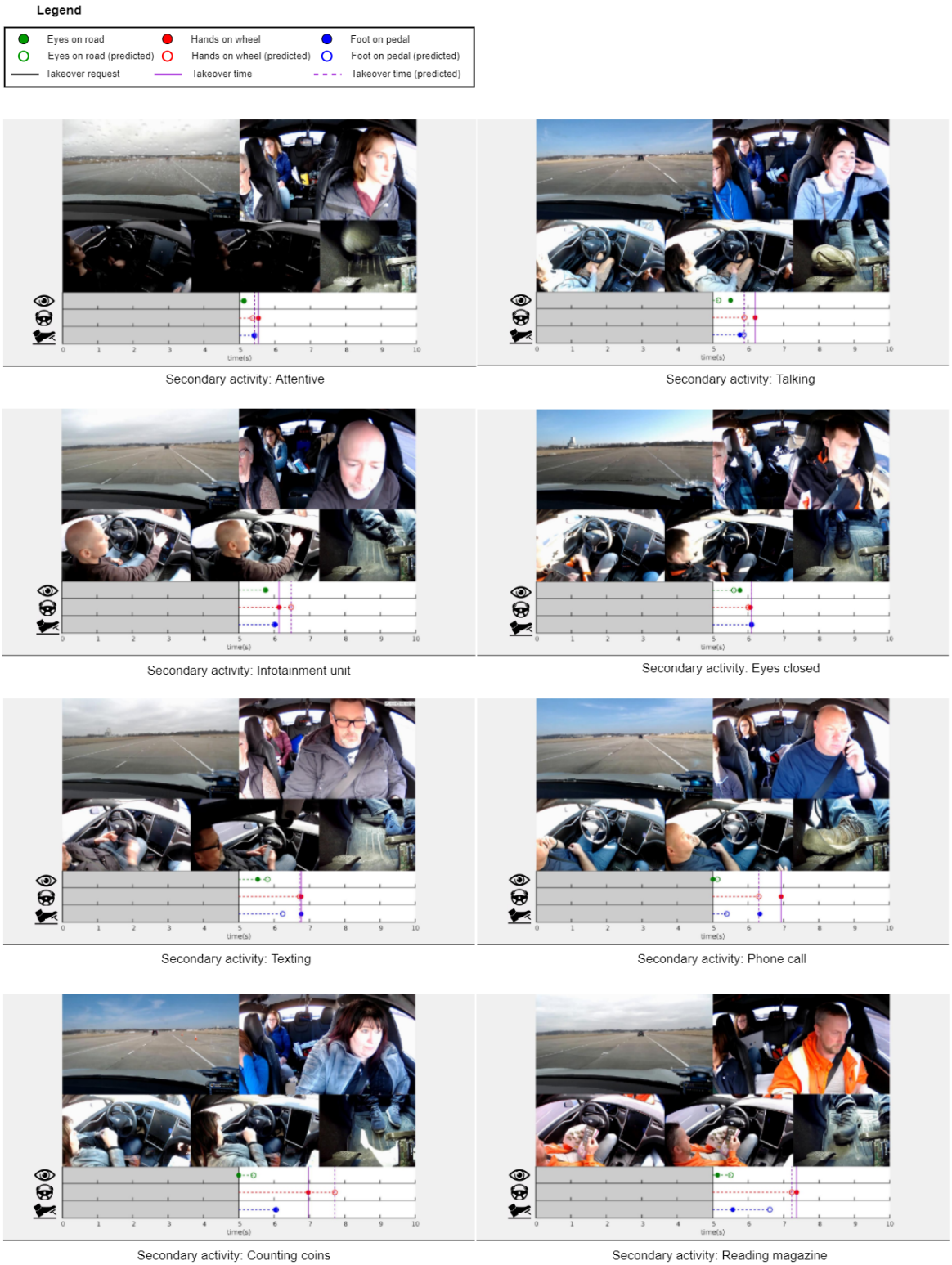


Figure 9.8: Qualitative examples showing predicted take-over times for different secondary activities in the CDS dataset.

Part V

Conclusion

Chapter 10

Concluding Remarks & Thoughts on the Future

In this dissertation, we have introduced and described a framework for the intelligent and highly-automated vehicles of the future. Our approach is inspired by the Looking-In & Looking-Out (LILO) philosophy, whereby human occupants are part of the decision and control process, instead of an afterthought. By incorporating the states and behaviors of human occupants into our system, we are able to ensure that both the autonomous agent and the human driver are aware of each other's states, goals and limitations, thereby avoiding issues such as absentee or unprepared drivers, overconfidence in the autonomous agent, complacency, etc. We believe that systems with such capabilities are essential in the era of partial automation, and may retain utility even after "complete" automation is achieved. It is also important to note that despite the prevailing categorization of automation capabilities (i.e. the SAE levels of automation), technological progress is rarely linear. One could argue that complete automation may never be realized, and that there will always exist extenuating circumstances or long-tail events that cannot be accounted for by the system's designers. Thus, it is essential to deploy systems that have an

engaged and vigilant driver as a fallback option, even if this is rarely relied upon.

In addition to offering a modern interpretation of the LILO framework, this dissertation also introduces unique models and algorithms to perform the individual tasks that comprise such a system. These models and algorithms were designed to be deployed in the real world, and hence came with a set of hard-and-fast requirements. This includes real-time operation, reasonable compute requirements, and limited memory usage. While our proposed models and algorithms are by no means complete or the silver bullet to all driver safety problems, we have tried our best to ensure that they meet the constraints of the real world, and are deployable in some reasonable form.

In Chapter 2, we introduce an approach to monocular 3D object detection based on ground planes and geometric constraints. Since our approach primarily relies on the successful prediction of 2D entities from images, it is capable of better generalization across different camera sensors, lenses, and terrains. However, some limitations of using only 2D images to estimate the locations of 3D objects remain. While monocular detectors are sufficient for near-range ADAS (Advanced Driver Assistance Systems) functionality, higher levels of automation probably require 3D sensors. This is especially evident from comparisons between LiDAR [260–262] or even stereo based object detectors [263, 264] and existing monocular object detectors [265]. However, most high-resolution LiDARs are orders of magnitude more expensive in comparison to cameras and radars. Even so, the future is quite promising - with cheaper LiDARs and better early-fusion methods likely around the corner. A cheap, low-resolution LiDAR coupled with a high-resolution camera and radar could potentially be the solution to robust object detection at distances of up to 100-200m.

Chapter 3 presents our approach to robust and interpretable lane detection using affinity fields. While lane detection is already a mature technology that finds use in many real world systems, there are some enhancements yet to be made. This includes the improved detection of curved lanes, unmarked lanes, faded lanes etc. While recent approaches (including ours) address

some of these issues, there are other shortcomings that arise purely from the lack of complex datasets. For example, one would ideally have a model that could detect all visible lanes (as opposed to the laterally nearest ones), stop/yield lines, curbs, and crosswalks. Although this is not available for public use at this moment, I believe that such datasets are not too far off in the future.

In Chapter 4, we propose a framework for multi-object tracking (MOT) based on dynamic, bipartite graphs that represent the data association problem over multiple timesteps. We then create a message-passing neural network that operates on it to assign likelihoods to every association therein. Although we achieve promising results using this framework, our approach and other MOT approaches in general have several challenges to overcome. This includes the long-term tracking of multiple objects (see Appendix A), drastically reducing fragmentation and ID switches, limiting/removing noise from the upstream object detector, and handling large extents of clutter. Unsurprisingly, models that try to achieve some of these goals [106, 266, 267] are typically more computationally expensive, run slower and possibly offline (i.e. they are non-causal). Additionally, the community is yet to reach a consensus on the best metrics for evaluation and comparison. While there are promising works that address some of the aforementioned issues [268], there is still a ways to go before we achieve the desired levels of performance and robustness in most environments.

Chapter 5 of this dissertation introduces a framework for joint maneuver classification and trajectory prediction for multiple agents in the scene. Since the publication of our work, the field of motion forecasting has gained tremendous traction. Recent years have seen the introduction of models that make use of scene topology [80, 269, 270], inter-agent relationships [271], ego-vehicle plans [272], and surround agent intentions [270]. While these models perform extremely well with high-quality, noise-free inputs, it is unclear how they would fare with noisy track histories generated by on-board multi-object trackers. Other issues to be addressed include the joint prediction of all agent futures instead of independently predicting each of them, better

ranking/scoring of predicted futures, ensuring real-time operation, and better metrics that penalize illegal/impossible futures.

In the second part of this dissertation, Chapters 6, 7 and 8 introduce unique models and approaches to monitor the driver state, by separately modelling individual aspects of driver behavior, i.e. the gaze, hands, and feet. These models have proven reliable under a variety of real-world conditions (including harsh illumination, nighttime driving, and the presence of eyeglasses in the case of gaze estimation). They also operate in real-time, and could potentially be deployed in a real-world system with additional engineering. However, there are still some concerns to be addressed. For instance, the models introduced in these chapters are somewhat sensitive to perspective changes of the imaging sensors. Additional research would be required before deploying these models across different vehicular cabins, and with potential changes in perspective. Another issue to be addressed is that of driver/occupant privacy. The introduction of imaging sensors into personal spaces such as a vehicle's cabin requires extensive efforts to ensure that the privacy and confidentiality of the vehicle's occupants are protected. While some studies have tried to address this [273], this research is still nascent and requires further investigation.

Finally, in Chapter 9, we present our recent study on control transitions and take-over times (TOTs) in partially automated vehicles. This multi-year effort resulted in one of the largest real-world datasets on control transitions involving a variety of challenging scenarios and a large pool of participant drivers. Another key contribution of this study was the use of specialized models (introduced in Chapters 6, 7 and 8) to analyze different aspects of driver behavior and produce mid-level features, followed by a sequential downstream model to predict TOT. This is in contrast to recent diver activity approaches [159, 255, 274] that infer activities directly from raw videos. Our approach has several advantages over these "single-shot" models - faster runtimes because of smaller model sizes, additional robustness to noise and divergences in the inputs, less data hungry and more data efficient, ability to generalize to new activities/behaviors not observed in the training data, and so on. Moreover, since the inputs to our TOT models are sensor-agnostic

mid-level features like gaze direction, distance of hands to the wheel, foot location, etc., we can easily transfer our trained TOT model to a different environment without any modifications (only the specialized vision models would have to be re-trained). This is especially important because datasets for tasks such as TOT prediction are very laborious and expensive to collect in large amounts.

Through this dissertation and Chapter 9 in particular, we are also able to provide a concrete example of how the LILO framework can be put to use in practice - where the TOTs predicted by *looking-in* are used in conjunction with metrics computed by *looking-out* to initiate safer and smoother control transitions. We believe that this is just one of many such driver-safety applications that could benefit from the simultaneous modelling of the vehicle's inside and outside. The field of self-driving cars and intelligent vehicles is fast-paced, innovative, and holds potential to significantly benefit humanity as a whole. Even though the community has made great strides in the performance and reliability of models/algorithms across a variety of driving related tasks, there is still much to do and many exciting problems that require solving. However, we must always remember that with any real-world deployments of such technology, it is *human lives* that are at stake.

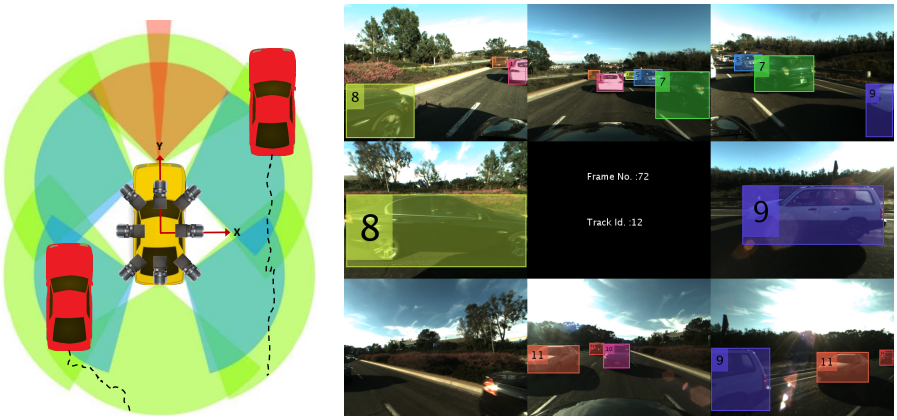
Appendix A

No Blind Spots: Full-Surround Multi-Object Tracking for Autonomous Vehicles using Cameras & LiDARs

Online multi-object tracking (MOT) is extremely important for high-level spatial reasoning and path planning for autonomous and highly-automated vehicles. In this appendix, we present a modular framework for tracking multiple objects (vehicles), capable of accepting object proposals from different sensor modalities (vision and range) and a variable number of sensors, to produce continuous object tracks. This work is a generalization of the Markov Decision Process (MDP) framework for MOT proposed in [275], with some key extensions - First, we track objects across multiple cameras and across different sensor modalities. This is done by fusing object proposals across sensors accurately and efficiently. Second, the objects of interest (targets) are tracked directly in the *real world*. This is a departure from traditional techniques where objects are simply tracked in the image plane. Doing so allows the tracks to be readily used by an autonomous agent for navigation and related tasks.

To verify the effectiveness of our approach, we test it on real world highway data collected from a heavily sensorized testbed capable of capturing full-surround infor-

Video results
Dataset download



(a) (b)

Figure A.1: **(a)** Illustration of online MOT for autonomous vehicles. The surrounding vehicles (in red) are tracked in a right-handed coordinate system centered on the ego-vehicle (center). The ego-vehicle has full-surround coverage from vision and range sensors, and must fuse proposals from each of them to generate continuous tracks (dotted lines) in the real world.
(b) An example of images captured from a full-surround camera array mounted on our testbed, along with color coded vehicle annotations.

mation. We demonstrate that our framework is well-suited to track objects through entire maneuvers around the ego-vehicle, some of which take more than a few minutes to complete. We also leverage the modularity of our approach by comparing the effects of including/excluding different sensors, changing the total number of sensors, and the quality of object proposals on the final tracking result.

A.1 Introduction

Tracking for autonomous vehicles involves accurately identifying and localizing dynamic objects in the environment surrounding the vehicle. Tracking of surround vehicles is essential for many tasks crucial to truly autonomous driving, such as *obstacle avoidance*, *path planning*, and *intent recognition*. To be useful for such high-level reasoning, the generated tracks should be accurate, long and robust to sensor noise. In this study, we propose a full-surround MOT framework to create such desirable tracks.

Traditional MOT techniques for autonomous vehicles can roughly be categorized into 3 groups based on the sensory inputs they use - 1) dense point clouds from range sensors, 2) vision

sensors, and 3) a fusion of range and vision sensors. Studies like [276–279] make use of dense point clouds created by 3D LiDARs like the Velodyne HDL-64E. Such sensors, although bulky and expensive, are capable of capturing finer details of the surroundings owing to its high vertical resolution. Trackers can therefore create suitable mid-level representations like 2.5D grids, voxels etc. that retain unique statistics of the volume they enclose, and group such units together to form coherent objects that can be tracked. It must be noted however, that these approaches are reliant on having dense point representations of the scene, and would not scale well to LiDAR sensors that have much fewer scan layers. On the other hand, studies such as [280–283] make use of stereo vision alone to perform tracking. The pipeline usually involves estimating the disparity image and optionally creating a 3D point cloud, followed by similar mid-level representations like stixels, voxels etc. which are then tracked from frame to frame. These sensors are limited by the quality of disparity estimates and the field of view (FoV) of the stereo pair. Unlike 3D LiDAR based systems, they are unable to track objects in full-surround. There are other single camera approaches to surround vehicle behavior analysis [118, 284], but they too are limited in their FoVs and localization capabilities. Finally, there are fusion based approaches like [285–287], that make use of LiDARs, stereo pairs, monocular cameras, and Radars in a variety of configurations. These techniques either perform *early* or *late* fusion based on their sensor setup and algorithmic needs. However, none of them seem to offer full-surround solutions for vision sensors, and are ultimately limited to fusion only in the FoV of the vision sensors.

In this study, we take a different approach to full-surround MOT and try to overcome some of the limitations in previous approaches. Specifically, we propose a framework to perform full-surround MOT using calibrated camera arrays, with varying degrees of overlapping FoVs, and with an option to include low resolution range sensors for accurate localization of objects in 3D. We term this the M^3OT framework, which stands for multi-perspective, multi-modal, multi-object tracking framework. To train and validate the M^3OT framework, we make use of naturalistic driving data collected from our testbed (illustrated in Figure A.1) that has full-surround coverage

from vision and range modalities.

Since we use vision as our primary perception modality, we leverage recent approaches from the 2D MOT community which studies tracking of multiple objects in the 2D image plane. Recent progress in 2D MOT has focused on the *tracking-by-detection* strategy, where object detections from a category detector are linked to form trajectories of the targets. To perform tracking-by-detection *online* (i.e. in a causal fashion), the major challenge is to correctly associate noisy object detections in the current video frame with previously tracked objects. The basis for any data association algorithm is a similarity function between object detections and targets. To handle ambiguities in association, it is useful to combine different cues in computing the similarity, and learn an association based on these cues. Many recent 2D MOT methods such as [112, 288–291] use some form of learning (online or offline) to accomplish data association. Similar to these studies, we formulate the online multi-object tracking problem using Markov Decision Processes (MDPs) proposed in [275], where the lifetime of an object is modeled with a MDP (see Figure A.5), and multiple MDPs are assembled for multi-object tracking. In this method, learning a similarity function for data association is equivalent to learning a policy for the MDP. The policy learning is approached in a reinforcement learning fashion which benefits from advantages of both offline-learning and online-learning in data association. The M^3OT framework is also capable to naturally handle the birth/death and appearance/disappearance of targets by treating them as state transitions in the MDP, and also benefits from the strengths of online learning approaches in single object tracking ([292–295]).

Our main contributions in this work can be summarized as follows - 1) We extend and improve the MDP formulation originally proposed for 2D MOT [275], and modify it to track objects in 3D (real world). 2) We make the M^3OT framework capable of tracking objects across multiple vision sensors in calibrated camera arrays by carrying out efficient and accurate fusion of object proposals. 3) The M^3OT framework is made highly *modular*, capable of working with any number of cameras, with varying degrees of overlapping FoVs, and with the option to

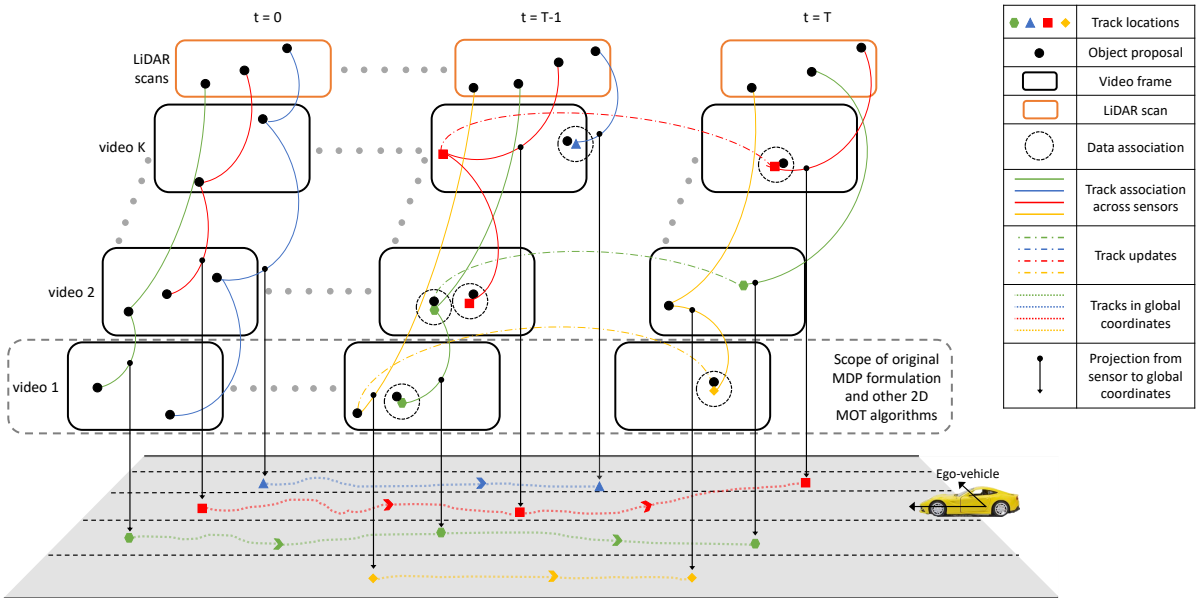


Figure A.2: Illustration of proposed M^{3}OT framework and its scope in comparison to traditional 2D MOT algorithms. Data is associated not only within video frames, but also across other videos and sensors. The algorithm produces tracks in each individual sensor coordinate frame, and in the desired global coordinate frame using cross-modal fusion in an online manner.

include range sensors for improved localization and fusion in 3D. The above contributions and the wider scope and applicability of this work in comparison to traditional 2D MOT approaches are highlighted in Figure A.2. Finally, we carry out experiments using naturalistic driving data collected on highways using full-surround sensory modalities, and validate the accuracy, robustness and modularity of our framework.

A.2 Related Research

We highlight some representative works in 2D and 3D MOT below. We also summarize the some key aspects of related 3D MOT studies in Table A.1. For a recent survey on detection and tracking for intelligent vehicles, we refer the reader to [84].

2D MOT: Recent research in MOT has focused on tracking-by-detection, where the main challenge is data association for linking object detections to targets. Majority of batch (*offline*)

methods ([112, 296–301]) formulate MOT as a global optimization problem in a graph-based representation, while *online* methods solve the data association problem either probabilistically [302–304] or deterministically (e.g., Hungarian algorithm [305] in [288,289] or greedy association [306]). A core component in any data association algorithm is a similarity function between objects. Both batch methods [112, 290] and online methods [288, 289, 291] have explored the idea of learning to track, where the goal is to learn a similarity function for data association from training data. In this work, we extend and improve the MDP framework for 2D MOT proposed in [275], which is an online method that uses reinforcement learning to learn associations.

3D MOT for autonomous vehicles: In [280], the authors use a stereo rig to first calculate the disparity using Semi-Global Matching (SGM), followed by height based segmentation and free-space calculation to create a mid-level representation using *stixels* that encode the height within a cell. Each stixel is then represented by a 6D state vector, which is tracked using the Extended Kalman Filter (EKF). In [281], the authors use a *voxel* based representation instead, and cluster neighboring voxels based on color to create objects that are then tracked using a greedy association model. The authors in [282] use a grid based representation of the scene, where cells are grouped to create objects, each of which is represented by a set of control points on the object surface. This creates a high dimensional state-space representation, which is accounted for by a Rao-Blackwellized particle filter. More recently, the authors of [283] propose to carry out semantic segmentation on the disparity image, which is then used to generate generic object proposals by creating a scale-space representation of the density, followed by multi-scale clustering. The proposed clusters are then tracked using a Quadratic Pseudo-Boolean Optimization (QPBO) framework. The work in [8] use a camera setup similar to ours, but the authors propose an offline framework for tracking, hence limiting their use to surveillance related applications.

Alternatively, there are approaches that make use of dense point clouds generated by LiDARs as opposed to creating point clouds from a disparity image. In [276], the authors first

Table A.1: Relevant research in 3D MOT for intelligent vehicles.

Study	Sensors used					Tracker Type			Experimental Analysis	
	Monocular camera	Stereo pair	Full-surround camera array	LiDAR	Radar	Single object tracker	Multi-object tracker	Online (causal)	Dataset	Evaluation metrics
Choi et al. [276]	-	-	-	✓	-	-	✓	✓	Proposed	Distance and velocity errors
Broggi et al. [281]	-	✓	-	-	-	-	✓	✓	Proposed	True positives, false positives, false negatives
Song et al. [277]	-	-	-	✓	-	✓	✗	✓	KITTI	Position error, intersection ratio
Cho et al. [285]	✓	-	-	✓	✓	-	✓	✓	Proposed	Correctly tracked, falsely tracked, true and false positive rates
Asvadi et al. [278]	-	-	-	✓	-	-	✓	✓	KITTI	-
Vatavu et al. [282]	-	✓	-	-	-	-	✓	✓	Proposed	-
Asvadi et al. [279]	-	-	-	✓	-	-	✓	✓	KITTI	Number of missed and false obstacles
Asvadi et al. [286]	✓	-	-	✓	-	✓	✗	✓	KITTI	Average center location errors in 2D and 3D, orientation errors
Ošep et al. [283]	-	✓	-	-	-	-	✓	✓	KITTI	Class accuracy, GOP recall, tracking recall
Allodi et al. [287]	-	✓	-	✓	-	-	✓	✓	KITTI	MOTP, IDS, Frag, average localization error
Dueholm et al. [8]	-	-	✓	-	-	-	✓	✗	VIVA Surround	MOTA, MOTP, IDS, Frag, Precision, Recall
This work[1] (M^{3}OT)	-	-	✓	✓	-	-	✓	✓	Proposed	MOTA, MOTP, MT, ML, IDS for a variety of sensor configurations

[1] this framework can work with or without LiDAR sensors, and with any subset of camera sensors.

carry out ground classification based on the variance in the radius of each scan layer, followed by a 2.5D occupancy grid representation of the scene. The grid is then segmented, and regions of interest (RoIs) identified within, each of which is tracked by a standard Kalman Filter (KF). Data association is achieved by simple global nearest neighbor. Similar to this, the authors in [278] use a 2.5D occupancy grid based representation, but augment this with an occupancy grid *map* which accumulates past grids to create a coherent global map of occupancy by accounting for ego-motion. Using these two representations, a 2.5D *motion* grid is created by comparing the map with the latest occupancy grid, which isolates and identifies dynamic objects in the scene. Although the work in [279] follows the same general idea, the authors propose a piece-wise ground plane estimation scheme capable of handling non-planar surfaces. In a departure from grid based methods, the authors in [277] project the 3D point cloud onto a virtual image plane, creating an object appearance model based on 4 image-based cues for each template of the desired target. A particle filtering framework is implemented, where the particle with least reconstruction error with respect to the stored template is chosen to update the tracker. Background filtering and occlusion detection are implemented to improve performance.

Finally, we list recent methods that rely on fusion of different sensor modalities to function. In [285], the authors propose an EKF based fusion scheme, where measurements from each modality are fed sequentially. Vision is used to classify the object category, which is then used to choose appropriate motion and observation models for the object. Once again, observations are associated based on a global nearest neighbor policy. This is somewhat similar to the work in [286], where given an initial 3D detection box, the authors propose to project 3D points within the box to the image plane and calculate the convex hull of projected points. In this case, a KF is used to perform both fusion and tracking, where fusion is carried out by projecting the 2D hull to a sparse 3D cloud, and using both 3D cues to perform the update. In contrast, the authors of [287] propose using the Hungarian algorithm (for bipartite matching) for both data association and fusion of object proposals from different sensors. The scores for association are obtained from Adaboost classifiers trained on high-level features. The objects are then tracked using an Unscented Kalman Filter (UKF).

A.3 Fusion of Object Proposals

In this study, we make use of full-surround camera arrays comprising of sensors with varying FoVs. The M^{3}OT framework, however, is capable of working with any type and number of cameras, as long as they are calibrated. In addition to this, we also propose a variant of the framework for cases where LiDAR point clouds are available. To effectively utilize all available sensors, we propose an *early fusion* of object proposals obtained from each of them. At the very start of each time step during tracking, we identify and fuse all proposals belonging to the same object. These proposals are then utilized by the M^{3}OT framework to carry out tracking. It must be noted that this usage of "early fusion" is in contrast to the traditional usage of the term to refer to fusion of raw sensor data to provide a merged representation.

Projection & Back-Projection

It is essential to have a way of associating measurements from different sensors to track objects across different camera views, and to carry out efficient fusion across sensor modalities. This is achieved by defining a set of *projection* mappings, one from each sensor's unique coordinate system to the global coordinate system, and a set of *back-projection* mappings that take measurements in the global coordinate system to individual coordinate systems. In our case, the global coordinate system is centered at the mid-point of the rear axle of the ego-vehicle. The axes form a right-handed coordinate system as shown in Figure A.1.

The LiDAR sensors output a 3D point cloud in a common coordinate system at every instant. This coordinate frame may either be centered about a single LiDAR sensor, or elsewhere depending on the configuration. In this case, the projection and back-projection mappings are simple 3D coordinate transformations:

$$P_{range \to G}(\mathbf{x}^{range}) = \boldsymbol{R}_{range} \cdot \mathbf{x}^{range} + \mathbf{t}_{range}\mathbf{l}, \tag{A.1}$$

and,

$$P_{range \leftarrow G}(\mathbf{x}^{G}) = \boldsymbol{R}^{T}_{range} \cdot \mathbf{x}^{G} - \boldsymbol{R}^{T}_{range}\mathbf{t}_{range}, \tag{A.2}$$

where $P_{range \to G}(\cdot)$ and $P_{range \leftarrow G}(\cdot)$ are the projection and back-projection mappings from the LiDAR (range) coordinate system to the global (G) coordinate system and vice-versa. The vectors $\mathbf{x}^{range}$ and $\mathbf{x}^{G}$ are the corresponding coordinates in the LiDAR and global coordinate frames. The 3×3 orthonormal rotation matrix $\boldsymbol{R}_{range}$ and translation vector $\mathbf{t}_{range}$ are obtained through calibration.

Similarly, the back-projection mappings for each camera $k \in \{1, \cdots, K\}$ can be defined

as:

$$P_{cam_k \leftarrow G}(\mathbf{x}^G) = (u, v)^T, \tag{A.3}$$

$$\text{s.t} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \boldsymbol{C}_k \cdot [\boldsymbol{R}_k | \mathbf{t}_k] \cdot [(\mathbf{x}^G)^T | 1]^T, \tag{A.4}$$

where the set of camera calibration matrices $\{\boldsymbol{C}_k\}_{k=1}^K$ are obtained after the intrinsic calibration of cameras, the set of tuples $\{(\boldsymbol{R}_k, \mathbf{t}_k)\}_{k=1}^K$ obtained after extrinsic calibration, and $(u, v)^T$ denotes the pixel coordinates after back-projection.

Unlike the back-projection mappings, the projection mappings for camera sensors are not well defined. In fact, the mappings are one-to-many due to the depth ambiguity of single camera images. To find a good estimate of the projection, we use two different approaches. In case of a vision-only system, we use the *inverse perspective mapping* (IPM) approach:

$$P_{cam_k \rightarrow G}(\mathbf{x}^k) = (x, y)^T, \tag{A.5}$$

$$\text{s.t.} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \boldsymbol{H}_k \cdot [(\mathbf{x}^k)^T | 1]^T, \tag{A.6}$$

where $\{\boldsymbol{H}_k\}_{k=1}^K$ are the set of homographies obtained after IPM calibration. Since we are only concerned with lateral and longitudinal displacements of vehicles in the global coordinate system, we only require the $(x, y)^T$ coordinates, and set the altitude coordinate to a fixed number.

Sensor Measurements & Object Proposals

As we adopt a tracking-by-detection approach, each sensor is used to produce object proposals to track and associate. In case of vision sensors, a vehicle detector is run on each individual camera's image to obtain multiple detections d, each of which is defined by a bounding

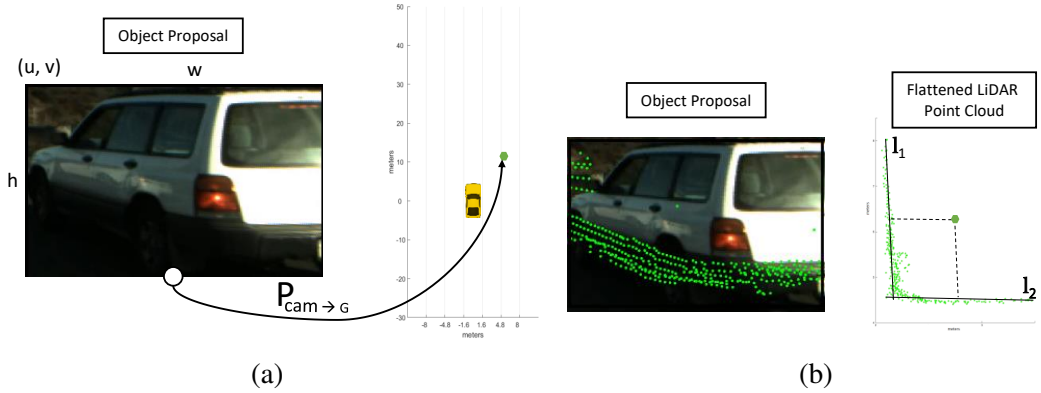


Figure A.3: Projection of object proposals using: (a) **IPM:** The bottom center of the bounding boxes are projected into the global coordinate frame (right), (b) **LiDAR point clouds:** LiDAR points that fall within a detection window are flattened and lines are fit to identify the vehicle center (right).

box in the corresponding image. Let (u,v) denote the top left corner and (w,h) denote the width and height of a detection d respectively.

In case of a vision-only system, the corresponding location of d in the global coordinate system is obtained using the mapping $P_{cam_k \to G}((u+\frac{w}{2}, v+h)^T)$, where k denotes the camera from which the proposal was generated. This procedure is illustrated in Figure A.3a. Alternatively, this could be replaced by a purely vision based 3D detector like the one proposed in [307], where both the global pose and 2D bounding box of an object are obtained from the same algorithm.

In cases where LiDAR sensors are available, an alternative is considered (shown in Figure A.3b). First, the back-projected LiDAR points that fall within a detection box d are identified using a look-up table with pixel coordinates as the key, and the corresponding global coordinates as the value. These points are then flattened by ignoring the altitude component of their global coordinates. Next, a line $\mathbf{l}_1$ is fitted to these points using RANSAC with a small inlier ratio (0.3). This line aligns with the dominant side of the detected vehicle. The other side of the vehicle corresponding to line $\mathbf{l}_2$ is then identified by removing all inliers from the previous step and repeating a RANSAC line fit with a slightly higher inlier ratio (0.4). Finally, the intersection of lines $\mathbf{l}_1$ and $\mathbf{l}_2$ along with the vehicle dimensions yield the global coordinates of the vehicle

center. The vehicle dimensions are calculated based on the extent of the LiDAR points along a given side of the vehicle, and stored for each track separately for later use.

Depending on the type of LiDAR sensors used, object proposals along with their dimensions in the real world can be obtained. However, we decide not to make use of LiDAR proposals, but rather use vision-only proposals with high recall by trading off some of the precision. This was seen to provide sufficient proposals to track all surrounding vehicles, at the expense of more false positives which the tracker is capable of handling.

Early Fusion of Proposals

Since we operate with camera arrays with overlapping FoVs, the same vehicle may be detected in two adjacent views. It is important to identify and *fuse* such proposals to track objects across camera views. Once again, we propose two different approaches to carry out this fusion. For vision-only systems, the fusion of proposals is carried out in 4 steps: i) Project proposals from all cameras to the global coordinate system using proposed mappings, ii) Sort all proposals in descending order based on their confidence scores (obtained from the vehicle detector), iii) Starting with the highest scoring proposal, find the subset of proposals whose euclidean distance in the global coordinate system falls within a predefined threshold. These proposals are considered to belong to the same object and removed from the original set of proposals. In practice, we use a threshold of $1m$ for grouping proposals. iv) The projection of each proposal within this subset is set to the mean of projections of all proposals within the subset. This process is repeated for the remaining proposals until no proposals remain.

Alternatively, for a system consisting of LiDAR sensors, we project the 3D point cloud onto each individual camera image. Next, for each pair of proposals, we make a decision as to whether or not they belong to the same object. This is done by considering the back-projected LiDAR points that fall within the bounding box of each proposal (see Figure A.4). Let $\mathcal{P}_1$ and $\mathcal{P}_2$ denote the index set of LiDAR points falling within each bounding box. Then, two proposals are

Figure A.4: Fusion of object proposals using LiDAR point clouds: Points common to both detections are drawn in green, and the rest are drawn red.

said to belong to the same object if:

$$\max\left(\frac{\mathcal{P}_1 \cap \mathcal{P}_2}{|\mathcal{P}_1|}, \frac{\mathcal{P}_1 \cap \mathcal{P}_2}{|\mathcal{P}_2|}\right) \geq 0.8, \tag{A.7}$$

where $|\mathcal{P}_1|$ and $|\mathcal{P}_2|$ denote the cardinalities of sets $\mathcal{P}_1$ and $\mathcal{P}_2$ respectively. It should be noted that after fusion is completed, the union of LiDAR point sets that are back-projected into fused proposals can be used to obtain better projections.

A.4 M^{3}OT Framework

Once we have a set of fused object proposals, we feed them into the MDP as illustrated in Figure A.5. Although the MDP framework introduced in [275] forms a crucial building block of the proposed M^{3}OT framework, we believe that extending this to account for multiple sensors and modalities is a non-trivial endeavor (see Figure A.2). This section highlights and explains the modifications we propose to achieve these objectives.

A.4.1 Markov Decision Process

As detailed in [275], we model the lifetime of a target with a Markov Decision Process (MDP). The MDP consists of the tuple $(\mathcal{S}, \mathcal{A}, T(\cdot,\cdot), R(\cdot,\cdot))$, where:

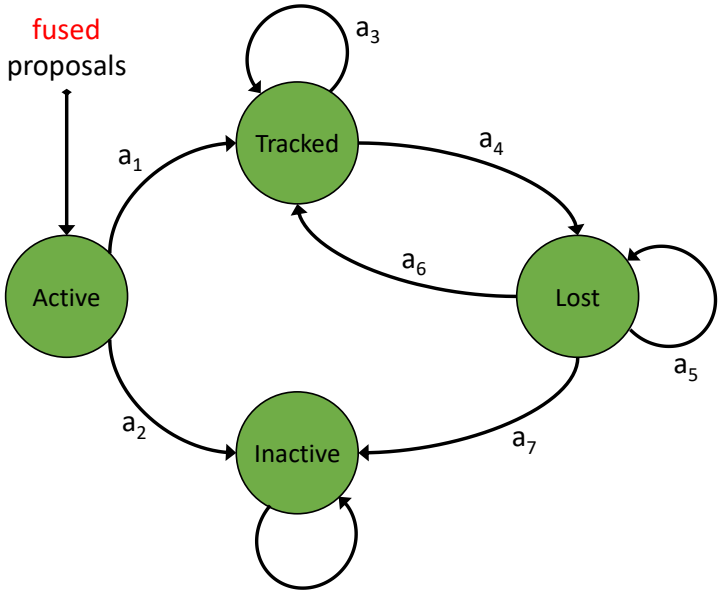


Figure A.5: The Markov Decision Process (MDP) framework proposed in [275]. In this work, we retain the structure of the MDP, and modify the actions, rewards and inputs to enable multi-sensory tracking.

- States $s \in \mathcal{S}$ encode the status of the target.
- Actions $a \in \mathcal{A}$ define the actions that can be taken.
- The state transition function $T : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S}$ dictates how the target transitions from one state to another, given an action.
- The real-valued reward function $R : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ assigns the immediate reward received after executing action a in state s.

In this study, we retain the states $\mathcal{S}$, actions $\mathcal{A}$ and the state transition function $T(\cdot,\cdot)$ from the MDP framework for 2D MOT [275], while changing only the reward function $R(\cdot,\cdot)$.

States

The state space is partitioned into four subspaces, i.e., $\mathcal{S} = \mathcal{S}_{Active} \cup \mathcal{S}_{Tracked} \cup \mathcal{S}_{Lost} \cup \mathcal{S}_{Inactive}$, where each subspace contains infinite number of states which encode the information of the target depending on the feature representation, such as appearance, location, size and history

of the target. Figure A.5 illustrates the transitions between the four subspaces. *Active* is the initial state for any target. Whenever an object is detected by the object detector, it enters an *Active* state. An active target can transition to *Tracked* or *Inactive*. Ideally, a true positive from the object detector should transition to a *Tracked* state, while a false alarm should enter an *Inactive* state. A tracked target can stay tracked, or transition to a *Lost* state if the target is not visible due to some reason, e.g. occlusion, or disappearance from sensor range. Likewise, a lost target can stay *Lost*, or go back to a *Tracked* state if it appears again, or transition to an *Inactive* state if it has been lost for a sufficiently long time. Finally, *Inactive* is the terminal state for any target, i.e., an inactive target stays inactive forever.

Actions and Transition Function

Seven possible transitions are designed between the state subspaces, which correspond to seven actions in our target MDP. Figure A.5 illustrates these transitions and actions. In the MDP, all the actions are deterministic, i.e., given the current state and an action, we specify a new state for the target. For example, executing action a_6 on a *Lost* target would transfer the target into a *Tracked* state, i.e., $T(s_{Lost}, a_6) = s_{Tracked}$.

Reward Function

As in the original study [275], we learn the reward function from training data, i.e., an inverse reinforcement learning problem, where we use ground truth trajectories of the targets as supervision.

A.4.2 Policy

Policy in Active States

In an Active state s, the MDP makes the decision between transferring an object proposal into a Tracked or Inactive state based on whether the detection is true or noisy. To do this, we train a set of binary Support Vector Machines (SVM) offline, one for each camera view, to classify

a detection belonging to that view into Tracked or Inactive states using a normalized 5D feature vector $\phi_{Active}(s)$, i.e., 2D image plane coordinates, width, height and score of the detection, where training examples are collected from training video sequences.

This is equivalent to learning the reward function:

$$R_{Active}(s,a) = y(a)((\mathbf{w}^k_{Active})^T \cdot \phi_{Active}(s) + b^k_{Active}), \tag{A.8}$$

for an object proposal belonging to camera $k \in \{1,\cdots,K\}$. $(\mathbf{w}^k_{Active}, b^k_{Active})$ defines the learned weights and bias of the SVM for camera k, $y(a) = +1$ if action $a = a_1$, and $y(a) = -1$ if $a = a_2$ (see Figure A.5). Training a separate SVM for each camera view allows weights to be learned based on object dimensions and locations in that particular view, and thus works better than training a single SVM for all views. Since a single object can result in multiple proposals, we initialize a tracker for that object if any of the fused proposals result in a positive reward. Note that a false positive from the object detector can still be misclassified and transferred to a Tracked state, which we then leave to be handled by the MDP in Tracked and Lost states.

Policy in Tracked States

In a *Tracked* state, the MDP needs to decide whether to keep tracking the target or to transfer it to a *Lost* state. As long as the target is visible, we should keep tracking it. Else, it should be marked "lost". We build an appearance model for the target online and use it to track the target. If the appearance model is able to successfully track the target in the next video frame, the MDP leaves the target in a Tracked state. Otherwise, the target is transferred to a Lost state.

Template Representation

The appearance of the target is simply represented by a template that is an image patch of the target in a video frame. Whenever an object detection is transferred to a Tracked state, we initialize the target template with the detection bounding box. If the target is initialized with

multiple fused proposals, then each detection is stored as a template. We make note of detections obtained from different camera views, and use these to model the appearance of the target in that view. This is crucial to track objects across camera views under varying perspective changes. When the target is being tracked, the MDP collects its templates in the tracked frames to represent the history of the target, which will be used in the Lost state for decision making.

Template Tracking

Tracking of templates is carried out by performing dense optical flow as described in [275]. The stability of the tracking is measured using the median of the Forward-Backward (FB) errors [295] of all sampled points: $e_{medFB} = median(e(\mathbf{u}_i)_{i=1}^{n})$, where $\mathbf{u}_i$ denotes each sampled point, and n is the total number of points. If e_{medFB} is larger than some threshold, the tracking is considered to be unstable. Moreover, after filtering out unstable matches whose FB error is larger than the threshold, a new bounding box of the target is predicted using the remaining matches by measuring scale change between points before and after. This process is carried out for all camera views in which a target template has been initialized and tracking is in progress.

Similar to the original MDP framework, we use the optical flow information in addition to the object proposals history to prevent drifting of the tracker. To do this, we compute the bounding box overlap between the target box for l past frames, and the corresponding detections in each of those frames. Then we compute the mean bounding box overlap for the past L tracked frames o_{mean} as another cue to make the decision. Once again, this process is repeated for each camera view the target is being tracked in. In addition to the above features, we also *gate* the target track. This involves introducing a check to see if the current global position of the tracked target falls within a window (gate) of it's last known global position. This forbids the target track from latching onto objects that appear close on the image plane, yet are much farther away in the global coordinate frame. We denote the last know global position and the currently tracked global position of the target as $\mathbf{x}^G(t-1)$ and $\hat{\mathbf{x}}^G(t)$ respectively.

Finally, we define the reward function in a Tracked state s using the feature set $\phi_{Tracked}(s) = (\{e^{k'}_{medFB}\}^{K'}_{k'=1}, \{o^{k'}_{mean}\}^{K'}_{k'=1}, \mathbf{x}^G(t-1), \hat{\mathbf{x}}^G(t))$ as:

$$R_{Tracked}(s,a) = \begin{cases} y(a), & \text{if } \exists k' \in \{1,\cdots,K'\} \text{ s.t.} \\ & (e^{k'}_{medFB} < e_0) \wedge (o^{k'}_{mean} > o_0) \\ & \wedge (|\mathbf{x}^G(t-1) - \hat{\mathbf{x}}^G(t)| \leq \mathbf{t}_{gate}), \\ -y(a), & \text{otherwise,} \end{cases} \tag{A.9}$$

where e_0 and o_0 are fixed thresholds, $y(a) = +1$ if action $a = a_3$, and $y(a) = -1$ if $a = a_4$ (see Figure A.5). k' above indexes camera views in which the target is currently being tracked and $\mathbf{t}_{gate}$ denotes the gating threshold. So the MDP keeps the target in a Tracked state if e_{medFB} is smaller and o_{mean} is larger than their respective thresholds for any one of K' camera views in addition to satisfying the gating check. Otherwise, the target is transferred to a Lost state.

Template Updating

The appearance model of the target needs to be regularly updated in order to accommodate appearance changes. As in the original work, we adopt a"lazy" updating rule and resort to the object detector in preventing tracking drift. This is done so that we don't accumulate tracking errors, but rather rely on data association to handle appearance changes and continue tracking. In addition to this, templates are initialized in views where the target is yet to be tracked by using proposals that are fused with detections corresponding to the tracked location in an adjacent camera view. This helps track objects that move across adjacent camera views, by creating target templates in the new view as soon as they are made available.

Policy in Lost States

In a Lost state, the MDP needs to decide whether to keep the target in a Lost state, or transition it to a Tracked state, or mark it as Inactive. We simply mark a lost target as Inactive and

Algorithm A.1: Reinforcement learning of the binary classifier for data association.

input : Set of multi-video sequences $\mathcal{V} = \{(v_i^1, \cdots, v_i^K)\}_{i=1}^{N}$, ground truth trajectories $\mathcal{T}_i = \{t_{ij}\}_{j=1}^{N_i}$, object proposals $\mathcal{D}_i = \{d_{im}\}_{m=1}^{M_i}$ and their corresponding projections for each multi-video sequence $(v_i^1, \cdots, v_i^K)$

output : Binary classifiers $\{(\mathbf{w}_{Lost}^k, b_{Lost}^k)\}_{k=1}^{K}$ for data association

repeat
  foreach *multi-video sequence* $(v_i^1, \cdots, v_i^K)$ *in* $\mathcal{V}$ **do**
    foreach *target* t_{ij} **do**
      Initialize the MDP in an Active state $l \leftarrow$ index of the first frame in which t_{ij} is correctly detected
      Transfer the MDP to a Tracked state and initialize the target template for each camera view in which target is observed **while** $l \leq$ *index of last frame of* t_{ij} **do**
        Fuse object proposals as described in A.3 Follow the current policy and choose an action a
        Compute the action a_{gt} according to the ground truth **if** *current state is lost and* $a \neq a_{gt}$ **then**
          foreach *camera view k in which the target has been seen* **do**
            Decide the label $y_{m_k}^k$ of the pair $(t_{ij}^k, d_{im_k}^k)$ $\mathcal{S}^k \leftarrow \mathcal{S}^k \cup \{(\phi(t_{ij}^k, d_{im_k}^k), y_{m_k})\}$
            $(\mathbf{w}_{Lost}^k, b_{Lost}^k) \leftarrow$ solution of Eq.A.11 on $\mathcal{S}^k$
          end
          break
        else
          Execute action a $l \leftarrow l+1$
        end
        if $l >$ *index of last frame of* t_{ij} **then**
          Mark target t_{ij} as successfully tracked
        end
      end
    end
  end
until *all targets are successfully tracked*;

terminate the tracking if the target has been lost for more than L_{Lost} frames. The more challenging task is to make the decision between tracking the target and keeping it as lost. This is treated as a data association problem where, in order to transfer a lost target into a Tracked state, the target needs to be associated with an object proposal, else, the target retains its Lost state.

Data Association

Let t denote a lost target, and d be an object detection. The goal of data association is to predict the label $y \in \{+1, -1\}$ of the pair (t, d) indicating that the target is linked $(y = +1)$ or not linked $(y = -1)$ to the detection. Assuming that the detection d belongs to camera view k, this binary classification is performed using the real-valued linear function $f^k(t,d) = (\mathbf{w}_{Lost}^k)^T \cdot \phi_{Lost}(t,d) + b_{Lost}^k$, where $(\mathbf{w}_{Lost}^k, b_{Lost}^k)$ are the parameters that control the function (for

camera view k), and $\phi_{Lost}(t,d)$ is the feature vector which captures the similarity between the target and the detection. The decision rule is given by $y = +1$ if $f^k(t,d) \geq 0$, else $y = -1$. Consequently, the reward function for data association in a lost state s given the feature set $\{\phi_{Lost}(t,d_j)\}_{m=1}^{M}$ is defined as

$$R_{Lost}(s,a) = y(a)\left(\max_{m=1}^{M}((\mathbf{w}_{Lost}^{k_m})^T \cdot \phi_{Lost}(t,d_m) + b_{Lost}^{k_m})\right), \tag{A.10}$$

where $y(a) = +1$ if action $a = a_6$, $y(a) = -1$ if $a = a_5$ (see Figure A.5), and m indexes M potential detections for association. Potential detections for association with a target are simply obtained by applying a gating function around the last known location of the target in the global coordinate system. Note that based on which camera view each detection d_m originates from, the appropriate weights $(\mathbf{w}_{Lost}^{k_m}, b_{Lost}^{k_m})$ associated with that view are used. As a result, the task of policy learning in the Lost state reduces to learning the set of parameters $\{(\mathbf{w}_{Lost}^{k}, b_{Lost}^{k})\}_{k=1}^{K}$ for the decision functions $\{f^k(t,d)\}_{k=1}^{K}$.

Reinforcement Learning

We train the binary classifiers described above using the reinforcement learning paradigm. Let $\mathcal{V} = \{(v_i^1, \cdots, v_i^K)\}_{i=1}^{N}$ denote a set of multi-video sequences for training, where N is the number of sequences and K is the total number of camera views. Suppose there are N_i ground truth targets $\mathcal{T}_i = \{t_{ij}\}_{j=1}^{N_i}$ in the i^{th} multi-video sequence $(v_i^1, \cdots, v_i^K)$. Our goal is to train the MDP to successfully track all these targets across all camera views they appear in. We start training with initial weights $(\mathbf{w}_0^k, b_0^k)$ and an empty training set $\mathcal{S}_0^k = \emptyset$ for the binary classifier corresponding to each camera view k. Note that when the weights of the binary classifiers are specified, we have a complete policy for the MDP to follow. So the training algorithm loops over all the multi-video sequences and all the targets, follows the current policy of the MDP to track the targets. The binary classifier or the policy is updated only when the MDP makes a mistake in data association. In this case, the MDP takes a different action than what is indicated by the

Table A.2: Features used for data association [275]. We introduce two new features (highlighted in bold) based on the global coordinate positions of targets and detections.

Type	Notation	Feature Description
FB error	$\phi_1, \cdots, \phi_5$	Mean of the median forward-backward errors from the entire, left half, right half, upper half and lower half of the templates obtained from optical flow
NCC	ϕ_6	Mean of the median Normalized Correlation Coefficients (NCC) between image patches around the matched points in optical flow
	ϕ_7	Mean of the median Normalized Correlation Coefficients (NCC) between image patches around the matched points obtained from optical flow
Height ratio	ϕ_8	Mean of ratios of the bounding box height of the detection to that of the predicted bounding boxes obtained from optical flow
	ϕ_9	Ratio of the bounding box height of the target to that of the detection
Overlap	ϕ_{10}	Mean of the bounding box overlaps between the detection and the predicted bounding boxes from optical flow
Score	ϕ_{11}	Normalized detection score
Distance	ϕ_{12}	Euclidean distance between the centers of the target and the detection after motion prediction of the target with a linear velocity model
	ϕ_{13}	**Lateral offset between last known global coordinate position of the target and that of the detection**
	ϕ_{14}	**Longitudinal offset between last known global coordinate position of the target and that of the detection**

ground truth trajectory. Suppose the MDP is tracking the j^{th} target t_{ij} in the video v_i^k, and on the l^{th} frame of the video, the MDP is in a lost state. Consider the two types of mistakes that can happen: i) The MDP associates the target $t_{ij}^k(l)$ to an object detection d_m^k which disagrees with the ground truth, i.e., the target is incorrectly associated to a detection. Then $\phi(t_{ij}^k(l), d_m^k)$ is added to the training set $\mathcal{S}^k$ of the binary classifier for camera k as a negative example. ii) The MDP decides to not associate the target to any detection, but the target is visible and correctly detected by a detection d_m^k based on the ground truth, i.e., the MDP missed the correct association. Then $\phi(t_{ij}^k(l), d_m^k)$ is added to the training set as a positive example. After the training set has been augmented, we update the binary classifier by re-training it on the new training set. Specifically, given the current training set $\mathcal{S}^k = \{(\phi(t_m^k, d_m^k), y_m^k)\}_{m=1}^M$, we solve the following soft-margin

optimization problem to obtain a max-margin classifier for data association in camera view k:

$$\min_{\mathbf{w}^k_{Lost}, b^k_{Lost}, \xi} \ \frac{1}{2}||\mathbf{w}^k_{Lost}||^2 + C\sum_{m=1}^{M}\xi_m$$
$$\text{s.t. } y^k_m\left((\mathbf{w}^k_{Lost})^T \cdot \phi(t^k_m, d^k_m) + b^k_{Lost}\right) \geq 1-\xi_m, \xi_m \geq 0, \forall m, \quad \text{(A.11)}$$

where $\xi_m, m = 1, \cdots, M$ are the slack variables, and C is a regularization parameter. Once the classifier has been updated, we obtain a new policy which is used in the next iteration of the training process. Note that based on which view the data association is carried out in, the weights of the classifier in that view are updated in each iteration. We keep iterating and updating the policy until all the targets are successfully tracked. Algorithm A.1 summarizes the policy learning algorithm.

Feature Representation

We retain the same feature representation described in [275], but add two features based on the lateral and longitudinal displacements of the last known target location and the object proposal location in the global coordinate system. This leverages 3D information that is otherwise unavailable in 2D MOT. Table A.2 summarizes our feature representation.

A.4.3 Multi-Object Tracking with MDPs

After learning the policy/reward of the MDP, we apply it to the multi-object tracking problem. We dedicate one MDP for each target, and the MDP follows the learned policy to track the object. Given a new input video frame, targets in tracked states are processed first to determine whether they should stay as tracked or transfer to lost states. Then we compute pairwise similarity between lost targets and object detections which are not covered by the tracked targets, where non-maximum suppression based on bounding box overlap is employed to suppress covered detections, and the similarity score is computed by the binary classifier for data association. After that, the similarity scores are used in the Hungarian algorithm [305] to obtain

Algorithm A.2: Multi-object tracking with MDPs.

input : A multi-video sequence $(v^1, \cdots, v^K)$, corresponding object proposals $\mathcal{D} = \{d_m\}_{m=1}^{M}$ and their projections, learned binary classifier weights $\{(\mathbf{w}^k_{Active}, b^k_{Active})\}_{k=1}^{K}$ and $\{(\mathbf{w}^k_{Lost}, b^k_{Lost})\}_{k=1}^{K}$
output : Trajectories of targets $\mathcal{T} = \{t_j\}_{j=1}^{N}$ in the sequence

foreach *frame l in* $(v^1, \cdots, v^K)$ **do**
  Fuse object proposals as described in A.3 `/* process targets in tracked states */`
  foreach *tracked target* t_j *in* $\mathcal{T}$ **do**
    Follow the policy, move the MDP of t_j to the next state
  end
  `/* process targets in lost states */`
  foreach *lost target* t_j *in* $\mathcal{T}$ **do**
    foreach *proposal* d_m *not covered by any tracked target* **do**
      Compute $f^{k_m}(t_j, d_m) = (\mathbf{w}^{k_m})^T_{Lost} \cdot \phi(t_j, d_m) + b^{k_m}_{Lost}$
    end
  end
  Data association with Hungarian algorithm for the lost targets Initialize target templates for uninitialized camera views using matched (fused) proposals **foreach** *lost target* t_j *in* $\mathcal{T}$ **do**
    Follow the assignment, move the MDP of t_j to the next state
  end
  `/* initialize new targets */`
  foreach *proposal* d_m *not covered by any tracked target in* $\mathcal{T}$ **do**
    Initialize a MDP for a new target t with proposal d_m **if** *action* a_1 *is taken following the policy* **then**
      Transfer t to the *tracked* state and initialize the target template for each camera view in which target is observed $\mathcal{T} \leftarrow \mathcal{T} \bigcup \{t\}$
    else
      Transfer t to the Inactive state
    end
  end
end

the assignment between detections and lost targets. According to the assignment, lost targets which are linked to some object detections are transferred to tracked states. Otherwise, they stay as lost. Finally, we initialize a MDP for each object detection which is not covered by any tracked target. Algorithm A.2 describes our 3D MOT algorithm using MDPs in detail. Note that, tracked targets have higher priority than lost targets in tracking, and detections covered by tracked targets are suppressed to reduce ambiguities in data association.

A.5 Experimental Analysis

Testbed

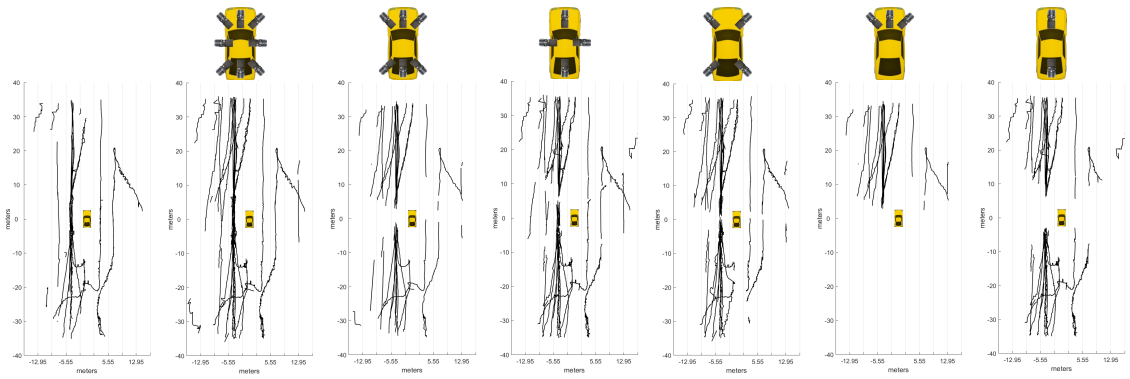
Table A.3: Quantitative results showing ablative analysis of our proposed tracker.

Criteria for Comparison	Tracker Variant	Sensor Configuration		MOT Metrics [111, 308]				
		# of Cameras	Range Sensors	MOTA (↑)	MOTP (↓)	MT (↑)	ML (↓)	IDS (↓)
Number of Cameras Used (Section A.5.1)	-	2	✓	73.38	0.03	71.36%	16.13%	16
	-	3	✓	77.26	0.03	77.34%	14.49%	38
	-	4	✓	72.81	0.05	72.48%	20.76%	49
	-	4†	✓	74.18	0.05	74.10%	18.18%	45
	-	6	✓	79.06	0.04	79.66%	11.93%	51
	-	8	✓	75.10	0.04	70.37%	14.07%	59
Projection Scheme (Section A.5.2)	Point cloud based projection	8	✓	**75.10**	**0.04**	**70.37%**	**14.07%**	**59**
	IPM projection	8	✓(for fusion)	47.45	0.39	53.70%	19.26%	152
Fusion Scheme (Section A.5.3)	Point cloud based fusion	8	✓	**75.10**	**0.04**	**70.37%**	14.07%	**59**
	Distance based fusion	8	✓(for projection)	72.20	**0.04**	68.23%	**12.23%**	65
Sensor Modality (Sections A.5.2,A.5.3)	Cameras+LiDAR	8	✓	**75.10**	**0.04**	**70.37%**	**14.07%**	**59**
	Cameras	8	✗	40.98	0.40	50.00%	27.40%	171
Vehicle Detector (Section A.5.4)	RefineNet [309]	8	✓	**75.10**	**0.04**	**70.37%**	**14.07%**	**59**
	RetinaNet [310]	8	✓	73.89	0.05	68.37%	17.07%	72
	SubCat [311]	8	✓	69.93	**0.04**	66.67%	22.22%	81
Global Position based Features (Section A.5.5)	with $\{\phi_{13}, \phi_{14}\}$	8	✓	**75.10**	**0.04**	**70.37%**	**14.07%**	**59**
	without $\{\phi_{13}, \phi_{14}\}$	8	✓	71.32	0.05	64.81%	17.78%	88

Since we propose full-surround MOT using vision sensors, we use a testbed comprising of 8 outside looking RGB cameras (seen in Figure A.1). This setup ensures full surround coverage of the scene around the vehicle, while retaining a sufficient overlap between adjacent camera views. Frames captured from these cameras along with annotated surround vehicles are shown in Figure A.1. In addition to full vision coverage, the testbed has full-surround Radar and LiDAR FoVs. Despite the final goal of this study being full-surround MOT, we additionally consider cases where only a subset of the vision sensors are used to illustrate the modularity of the approach. More details on the sensors, their synchronization and calibration, and the testbed can be found in [45].

Dataset

To train and test our 3D MOT system, we collect a set of four sequences, each 3-4 minutes long, comprising of multi-camera videos and LiDAR point clouds using our testbed described above. The sequences are chosen much longer than traditional MOT sequences so that long range maneuvers of surround vehicles can be tracked. This is very crucial to autonomous driving. We also annotate all vehicles in the 8 camera videos for each sequence with their bounding box, as



(a) GT (b) 8 (c) 6 (d) 4 (e) 4† (f) 3 (g) 2

Figure A.6: Tracking results with different number of cameras. The camera configuration used is depicted above each result.

well as track IDs. It should be noted that each unique vehicle in the scene is assigned the same ID in all camera views. With these sequences set up, we use one sequence for training our tracker, and reserve the rest for testing. All our results are reported on the entire test set.

Evaluation Metrics

We use multiple metrics to evaluate the multiple object tracking performance as suggested by the MOT Benchmark [111]. Specifically, we use the 3D MOT metrics described in [308]. These include Multiple Object Tracking Accuracy (MOTA), Multiple Object Tracking Precision (MOTP), Mostly Track targets (MT, percentage of ground truth objects who trajectories are covered by the tracking output for at least 80%), Mostly Lost targets (ML, percentage of ground truth objects who trajectories are covered by the tracking output less than 20%), and the total number of ID Switches (IDS). In addition to listing the metrics in Table A.3, we also draw arrows next to each of them indicating if a high (↑) or low (↓) value is desirable. Finally, we provide top-down visualizations of the tracking results in a global coordinate system centered on the ego-vehicle for qualitative evaluation.

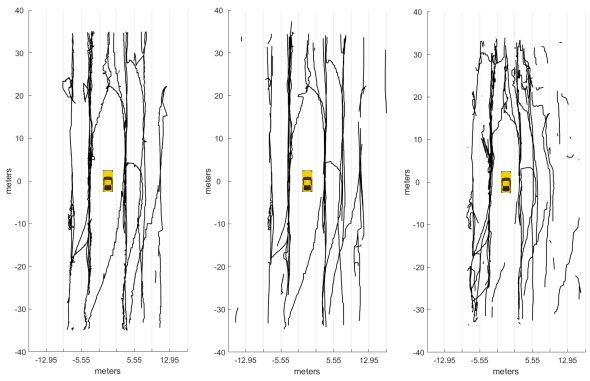
A.5.1 Experimenting with Number of Cameras

As our approach to tracking is designed to be extremely modular, we test our tracker with different camera configurations. We experiment with 2, 3, 4, 6 and 8 cameras respectively. Top-down visualizations of the generated tracks for a test sequence are depicted in Figure A.6. The ground truth tracks are provided for visual comparison. As can be seen, the tracker provides consistent results in its FoV irrespective of the camera configuration used, even if the cameras have no overlap between them.

The quantitative results on the test set for each camera configuration are listed in Table A.3. It must be noted that the tracker for each configuration is scored only based on the ground truth tracks visible in that camera configuration. The tracker is seen to score very well on each metric, irrespective of the number of cameras used. This illustrates the robustness of the M^3OT framework. More importantly, it is seen that our tracker performs exceptionally well in the MT and ML metrics, especially in camera configurations with overalapping FoVs. Even though our test sequences are about 3 minutes long in duration, the tracker mostly tracks more than 70% of the targets, while mostly losing only a few. This demonstrates that our M^3OT framework is capable of long-term target tracking.

A.5.2 Effect of Projection Scheme

Figure A.7 depicts the tracking results for a test sequence using the two projection schemes proposed. It is obvious that LiDAR based projection results in much better localization in 3D, which leads to more stable tracks and fewer fragments. The IPM based projection scheme is very sensitive to small changes in the input domain, and this leads to considerable errors during gating and data association. This phenomenon is verified by the high MOTP value obtained for IPM based projection as listed in Table A.3.



(a) GT (b) LiDAR-based projection (c) IPM-based projection

Figure A.7: Tracking results on a test sequence with different projection schemes.

A.5.3 Effect of Fusion Scheme

Once again, we see that the LiDAR point cloud based fusion scheme is more reliable in comparison to the distance based approach, albeit this difference is much less noticeable when proposals are projected using LiDAR point clouds. The LiDAR based fusion scheme results in objects being tracked longer (across camera views), and more accurately. The distance based fusion approach on the other hand fails to associate certain proposals, which results in templates not being stored for new camera views, thereby cutting short the track as soon as the target exits the current view. This superiority is reflected in the quantitative results shown in Table A.3. The drawbacks of the distance based fusion scheme are exacerbated when using IPM to project proposals, reflected by the large drop in MOTA for a purely vision based system. This drop in performance is to be expected in the absence of LiDAR sensors. However, it must be noted that half the targets are still tracked for most of their lifetime, while only a quarter of the targets are mostly lost.

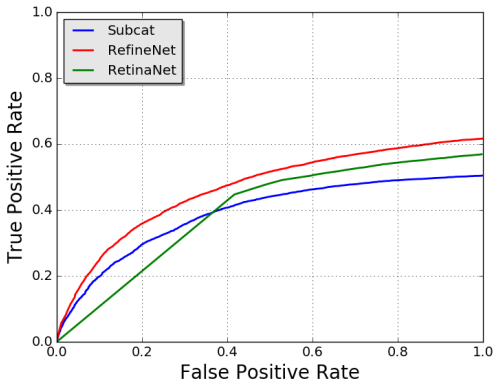


Figure A.8: ROC curves for different vehicle detectors on the 4 test sequences.

A.5.4 Effect of using Different Vehicle Detectors

Ideally, a tracking-by-detection approach should be detector agnostic. To observe how the tracking results change for different vehicle detectors, we ran the proposed tracker on vehicle detections obtained from three commonly used object detectors [309–311]. All three detectors were trained on the KITTI dataset [312] and have not seen examples from the proposed multi-camera dataset. The ROC (receiver operating characteristic) curves for the detectors on the proposed dataset are shown in Figure A.8. The corresponding tracking results for each detector are listed in Table A.3. Despite the sub-optimal performance of all three detectors in addition to significant differences in their ROC curves, the tracking results are seen to be relatively unaffected. This indicates that the tracker is less sensitive to errors made by the detector, and consistently manages to correct for it.

A.5.5 Effect of Global Position based Features

Table A.3 indicates a clear benefit in incorporating features $\{\phi_{13}, \phi_{14}\}$ for data association in Lost states. These features express how near/far a proposal is from the last know location of a target. This helps the tracker disregard proposals that are unreasonably far away from the latest target location. Introduction of these features leads to an improvement in all metrics and therefore

justifies their inclusion.

A.6 Chapter Summary

In this work, we have described a full-surround camera and LiDAR based approach to multi-object tracking for autonomous vehicles. To do so, we extend a 2D MOT approach based on the tracking-by-detection framework, and make it capable of tracking objects in the real world. The proposed M^{3}OT framework is also made highly modular so that it is capable of working with any camera configuration with varying FoVs, and also with or without LiDAR sensors. An efficient and fast early fusion scheme is adopted to handle object proposals from different sensors within a calibrated camera array. We conduct extensive testing on naturalistic full-surround vision and LiDAR data collected on highways, and illustrate the effects of different camera setups, fusion schemes and 2D-to-3D projection schemes, both qualitatively and quantitatively. Results obtained on the dataset support the modular nature of our framework, as well as its ability to track objects for a long duration. In addition to this, we believe that the M^{3}OT framework can be used to test the utility of any camera setup, and make suitable modifications thereof to ensure optimum coverage from vision and range sensors.

A.7 Acknowledgements

Appendix A, in full, is a reprint of the material as it appears in the IEEE Transactions on Intelligent Vehicles (2019), by Akshay Rangesh, and Mohan M. Trivedi. The dissertation author was the primary investigator and author of this paper.

Bibliography

[1] S. M. Casner, E. L. Hutchins, and D. Norman, "The challenges of partially automated driving," *Communications of the ACM*, vol. 59, no. 5, pp. 70–77, 2016.

[2] M. Kyriakidis, J. C. de Winter, N. Stanton, T. Bellet, B. van Arem, K. Brookhuis, M. H. Martens, K. Bengler, J. Andersson, N. Merat *et al.*, "A human factors perspective on automated driving," *Theoretical Issues in Ergonomics Science*, pp. 1–27, 2017.

[3] M. M. Trivedi, T. Gandhi, and J. McCall, "Looking-in and looking-out of a vehicle: Computer-vision-based enhanced vehicle safety," *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, no. 1, pp. 108–120, 2007.

[4] A. Tawari, S. Sivaraman, M. Trivedi, T. Shannon, and M. Tippelhofer, "Looking-in and looking-out vision for urban intelligent assistance: Estimation of driver attentive state and dynamic surround for safe merging and braking," in *IEEE Intelligent Vehicles Symposium*, 2014.

[5] S. D. Pendleton, H. Andersen, X. Du, X. Shen, M. Meghjani, Y. H. Eng, D. Rus, and M. H. Ang, "Perception, planning, control, and coordination for autonomous vehicles," *Machines*, vol. 5, no. 1, p. 6, 2017.

[6] A. Rangesh and M. M. Trivedi, "No blind spots: Full-surround multi-object tracking for autonomous vehicles using cameras & lidars," *arXiv preprint arXiv:1802.08755*, 2018.

[7] N. Deo, A. Rangesh, and M. M. Trivedi, "How would surround vehicles move? a unified framework for maneuver classification and motion prediction," *IEEE Transactions on Intelligent Vehicles*, vol. 3, no. 2, pp. 129–140, 2018.

[8] J. V. Dueholm, M. S. Kristoffersen, R. K. Satzoda, T. B. Moeslund, and M. M. Trivedi, "Trajectories and maneuvers of surrounding vehicles with panoramic camera arrays," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 2, pp. 203–214, 2016.

[9] N. Deo and M. M. Trivedi, "Convolutional social pooling for vehicle trajectory prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 1468–1476.

[10] N. Deo, N. Meoli, A. Rangesh, and M. Trivedi, "On control transitions in autonomous driving: A framework and analysis for characterizing scene complexity," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2019, pp. 0–0.

[11] W. Zimmer, A. Rangesh, and M. Trivedi, "3d bat: A semi-automatic, web-based 3d annotation toolbox for full-surround, multi-modal data streams," *arXiv preprint arXiv:1905.00525*, 2019.

[12] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[13] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[14] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[15] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[16] T. Kong, A. Yao, Y. Chen, and F. Sun, "Hypernet: Towards accurate region proposal generation and joint object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 845–853.

[17] F. Yang, W. Choi, and Y. Lin, "Exploit all the layers: Fast and accurate cnn object detector with scale dependent pooling and cascaded rejection classifiers," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2129–2137.

[18] D. G. Lowe *et al.*, "Object recognition from local scale-invariant features." in *iccv*, vol. 99, no. 2, 1999, pp. 1150–1157.

[19] F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce, "3d object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints," *International Journal of Computer Vision*, vol. 66, no. 3, pp. 231–259, 2006.

[20] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International conference on computer vision*. Ieee, 2011, pp. 2564–2571.

[21] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *European conference on computer vision*. Springer, 2006, pp. 404–417.

[22] Y. Li, L. Gu, and T. Kanade, "Robustly aligning a shape model and its application to car alignment of unknown pose," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 9, pp. 1860–1876, 2011.

[23] D. G. Lowe, "Fitting parameterized three-dimensional models to images," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 5, pp. 441–450, 1991.

[24] D. P. Huttenlocher, W. J. Rucklidge, and G. A. Klanderman, "Comparing images using the hausdorff distance under translation," in *Proceedings 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 1992, pp. 654–656.

[25] M.-Y. Liu, O. Tuzel, A. Veeraraghavan, and R. Chellappa, "Fast directional chamfer matching," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 2010, pp. 1696–1703.

[26] K. Ramnath, S. N. Sinha, R. Szeliski, and E. Hsiao, "Car make and model recognition using 3d curve alignment," in *IEEE Winter Conference on Applications of Computer Vision*. IEEE, 2014, pp. 285–292.

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

[28] Y. Xiang, R. Mottaghi, and S. Savarese, "Beyond pascal: A benchmark for 3d object detection in the wild," in *IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2014, pp. 75–82.

[29] K. Matzen and N. Snavely, "Nyc3dcars: A dataset of 3d vehicles in geographic context," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 761–768.

[30] Y. Xiang, W. Choi, Y. Lin, and S. Savarese, "Data-driven 3d voxel patterns for object category recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1903–1911.

[31] ——, "Subcategory-aware convolutional neural networks for object proposals and detection," in *IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2017, pp. 924–933.

[32] F. Chabot, M. Chaouch, J. Rabarisoa, C. Teulière, and T. Chateau, "Deep manta: A coarse-to-fine many-task network for joint 2d and 3d vehicle analysis from monocular image," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.(CVPR)*, 2017, pp. 2040–2049.

[33] X. Chen, K. Kundu, Y. Zhu, A. G. Berneshawi, H. Ma, S. Fidler, and R. Urtasun, "3d object proposals for accurate object class detection," in *Advances in Neural Information Processing Systems*, 2015, pp. 424–432.

[34] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun, "Monocular 3d object detection for autonomous driving," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2147–2156.

[35] A. Mousavian, D. Anguelov, J. Flynn, and J. Košecká, "3d bounding box estimation using deep learning and geometry," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017, pp. 5632–5640.

[36] B. Tekin, S. N. Sinha, and P. Fua, "Real-time seamless single shot 6d object pose prediction," *arXiv preprint arXiv:1711.08848*, 2017.

[37] T.-Y. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, "Feature pyramid networks for object detection." in *CVPR*, vol. 1, no. 2, 2017, p. 4.

[38] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[39] H. Alhaija, S. Mustikovela, L. Mescheder, A. Geiger, and C. Rother, "Augmented reality meets computer vision: Efficient data generation for urban driving scenes," *International Journal of Computer Vision (IJCV)*, 2018.

[40] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 3354–3361.

[41] A. Naiden, V. Paunescu, G. Kim, B. Jeon, and M. Leordeanu, "Shift r-cnn: Deep monocular 3d object detection with closed-form geometric constraints," *arXiv preprint arXiv:1905.09970*, 2019.

[42] J. Ku, A. D. Pon, and S. L. Waslander, "Monocular 3d object detection leveraging accurate proposals and shape reconstruction," *arXiv preprint arXiv:1904.01690*, 2019.

[43] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[44] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," *arXiv preprint arXiv:1903.11027*, 2019.

[45] A. Rangesh, K. Yuen, R. K. Satzoda, R. N. Rajaram, P. Gunaratne, and M. M. Trivedi, "A multimodal, full-surround vehicular testbed for naturalistic studies and benchmarking: Design, calibration and deployment," *arXiv preprint arXiv:1709.07502*, 2017.

[46] M. Daily, S. Medasani, R. Behringer, and M. Trivedi, "Self-driving cars," *Computer*, vol. 50, no. 12, pp. 18–23, 2017.

[47] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7291–7299.

[48] K. Yuen and M. M. Trivedi, "Looking at hands in autonomous vehicles: A convnet approach using part affinity fields," *IEEE Transactions on Intelligent Vehicles*, vol. 5, no. 3, pp. 361–371, 2019.

[49] F. Yu, D. Wang, E. Shelhamer, and T. Darrell, "Deep layer aggregation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2403–2412.

[50] R. K. Satzoda and M. M. Trivedi, "On enhancing lane estimation using contextual cues," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 11, pp. 1870–1881, 2015.

[51] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "Enet: A deep neural network architecture for real-time semantic segmentation," *arXiv preprint arXiv:1606.02147*, 2016.

[52] R. K. Satzoda and M. M. Trivedi, "Drive analysis using vehicle dynamics and vision-based lane semantics," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 1, pp. 9–18, 2014.

[53] Q. Zou, H. Jiang, Q. Dai, Y. Yue, L. Chen, and Q. Wang, "Robust lane detection from continuous driving scenes using deep neural networks," *IEEE transactions on vehicular technology*, vol. 69, no. 1, pp. 41–54, 2019.

[54] M. Ghafoorian, C. Nugteren, N. Baka, O. Booij, and M. Hofmann, "El-gan: Embedding loss driven generative adversarial networks for lane detection," in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 2018, pp. 0–0.

[55] X. Pan, J. Shi, P. Luo, X. Wang, and X. Tang, "Spatial as deep: Spatial cnn for traffic scene understanding," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[56] J. Philion, "Fastdraw: Addressing the long tail of lane detection by adapting a sequential prediction network," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 582–11 591.

[57] W. Van Gansbeke, B. De Brabandere, D. Neven, M. Proesmans, and L. Van Gool, "End-to-end lane detection through differentiable least-squares fitting," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019, pp. 0–0.

[58] Y. Hou, Z. Ma, C. Liu, and C. C. Loy, "Learning lightweight lane detection cnns by self attention distillation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1013–1021.

[59] D. Neven, B. De Brabandere, S. Georgoulis, M. Proesmans, and L. Van Gool, "Towards end-to-end lane detection: an instance segmentation approach," in *2018 IEEE intelligent vehicles symposium (IV)*. IEEE, 2018, pp. 286–291.

[60] F. Pizzati, M. Allodi, A. Barrera, and F. García, "Lane detection and classification using cascaded cnns," in *International Conference on Computer Aided Systems Theory*. Springer, 2019, pp. 95–103.

[61] Y. Ko, J. Jun, D. Ko, and M. Jeon, "Key points estimation and point instance segmentation approach for lane detection," *arXiv preprint arXiv:2002.06604*, 2020.

[62] M. Bai and R. Urtasun, "Deep watershed transform for instance segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5221–5229.

[63] J. Uhrig, M. Cordts, U. Franke, and T. Brox, "Pixel-level encoding and depth layering for instance-level semantic labeling," in *German Conference on Pattern Recognition*. Springer, 2016, pp. 14–25.

[64] T. Gupta, H. S. Sikchi, and D. Charkravarty, "Robust lane detection using multiple features," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1470–1475.

[65] N. Garnett, R. Cohen, T. Pe'er, R. Lahav, and D. Levi, "3d-lanenet: end-to-end 3d multiple lane detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 2921–2930.

[66] L. Tabelini, R. Berriel, T. M. Paixão, C. Badue, A. F. De Souza, and T. Olivera-Santos, "Keep your eyes on the lane: Attention-guided lane detection," *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2021.

[67] Y. Huang, S. Chen, Y. Chen, Z. Jian, and N. Zheng, "Spatial-temproal based lane detection using deep learning," in *IFIP International conference on artificial Intelligence applications and innovations*. Springer, 2018, pp. 143–154.

[68] M. Bai, G. Mattyus, N. Homayounfar, S. Wang, S. K. Lakshmikanth, and R. Urtasun, "Deep multi-sensor lane detection," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3102–3109.

[69] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[70] E. Romera, J. M. Alvarez, L. M. Bergasa, and R. Arroyo, "Erfnet: Efficient residual factorized convnet for real-time semantic segmentation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 1, pp. 263–272, 2017.

[71] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[72] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 764–773.

[73] K. Behrendt and R. Soussan, "Unsupervised labeled lane markers using maps," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019, pp. 0–0.

[74] L. Tabelini, R. Berriel, T. M. Paixao, C. Badue, A. F. De Souza, and T. Oliveira-Santos, "Polylanenet: Lane estimation via deep polynomial regression," *arXiv preprint arXiv:2004.10924*, 2020.

[75] Y. Hou, Z. Ma, C. Liu, T.-W. Hui, and C. C. Loy, "Inter-region affinity distillation for road marking segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 12 486–12 495.

[76] Z. Li, "Curvelane-nas: Unifying lane-sensitive architecture search and adaptive point blending," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2020.

[77] T. Liu, Z. Chen, Y. Yang, Z. Wu, and H. Li, "Lane detection in low-light conditions using an efficient data enhancement: Light conditions style transfer," in *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2020, pp. 1394–1399.

[78] S. Yoo, H. S. Lee, H. Myeong, S. Yun, H. Park, J. Cho, and D. H. Kim, "End-to-end lane marker detection via row-wise classification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 1006–1007.

[79] T. Zheng, H. Fang, Y. Zhang, W. Tang, Z. Yang, H. Liu, and D. Cai, "Resa: Recurrent feature-shift aggregator for lane detection," *arXiv preprint arXiv:2008.13719*, 2020.

[80] N. Deo and M. M. Trivedi, "Trajectory forecasts in unknown environments conditioned on grid-based plans," *arXiv preprint arXiv:2001.00735*, 2020.

[81] E. Ohn-Bar and M. M. Trivedi, "Looking at humans in the age of self-driving and highly automated vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 90–104, 2016.

[82] D. Ridel, E. Rehder, M. Lauer, C. Stiller, and D. Wolf, "A literature review on the prediction of pedestrian behavior in urban scenarios," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 3105–3112.

[83] H. Swathi, G. Shivakumar, and H. Mohana, "Crowd behavior analysis: A survey," in *2017 international conference on recent advances in electronics and communication technology (ICRAECT)*. IEEE, 2017, pp. 169–178.

[84] S. Sivaraman and M. M. Trivedi, "Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior analysis," *IEEE transactions on intelligent transportation systems*, vol. 14, no. 4, pp. 1773–1795, 2013.

[85] T. He, H. Mao, J. Guo, and Z. Yi, "Cell tracking using deep neural networks with multi-task learning," *Image and Vision Computing*, vol. 60, pp. 142–153, 2017.

[86] I. J. Cox and S. L. Hingorani, "An efficient implementation of reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1996, pp. 138 – 150.

[87] Y. Zhang, C. Wang, X. Wang, W. Zeng, and W. Liu, "Fairmot: On the fairness of detection and re-identification in multiple object tracking," *arXiv: 2004.01888*, 2020.

[88] H.-N. Hu, Q.-Z. Cai, D. Wang, J. Lin, M. Sun, P. Krähenbühl, T. Darrell, and F. Yu, "Joint monocular 3d vehicle detection and tracking," in *International Conference on Computer Vision (ICCV)*, 2019.

[89] X. Zhou, V. Koltun, and P. Krähenbühl, "Tracking objects as points," *ECCV*, 2020.

[90] X. Zeng, R. Liao, L. Gu, Y. Xiong, S. Fidler, and R. Urtasun, "Dmm-net: Differentiable mask-matching network for video object segmentation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 3929–3938.

[91] P. Voigtlaender, M. Krause, A. Osep, J. Luiten, B. B. G. Sekar, A. Geiger, and B. Leibe, "Mots: Multi-object tracking and segmentation," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[92] J. Luiten, T. Fischer, and B. Leibe, "Track to reconstruct and reconstruct to track," *arXiv preprint arXiv:1910.00130*, 2019.

[93] S. Hamid Rezatofighi, A. Milan, Z. Zhang, Q. Shi, A. Dick, and I. Reid, "Joint probabilistic data association revisited," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 3047–3055.

[94] C. Kim, F. Li, A. Ciptadi, and J. M. Rehg, "Multiple hypothesis tracking revisited," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 4696–4704.

[95] S. Sharma, J. A. Ansari, J. Krishna Murthy, and K. Madhava Krishna, "Beyond pixels: Leveraging geometry and shape cues for online multi-object tracking," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[96] H. Karunasekera, H. Wang, and H. Zhang, "Multiple object tracking with attention to appearance, structure, motion and size," *IEEE Access*, 2019.

[97] J. Pöschmann, T. Pfeifer, and P. Protzel, "Factor graph based 3d multi-object tracking in point clouds," *arXiv preprint arXiv:2008.05309*, 2020.

[98] A. Rangesh and M. M. Trivedi, "No blind spots: Full-surround multi-object tracking for autonomous vehicles using cameras and lidars," *IEEE Transactions on Intelligent Vehicles*, vol. 4, no. 4, pp. 588–599, 2019.

[99] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020.

[100] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," in *arXiv preprint arXiv:1511.08458*, 2015.

[101] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *arXiv:1704.01212*, 2017.

[102] R. L. Sergio Casas, Cole Gulino and R. Urtasun, "SpAGNN spatially-aware graph neural networks for relational behavior forecasting from sensor data," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[103] X. Weng, Y. Yuan, and K. Kitani, "Joint 3d tracking and forecasting with graph neural network and diversity sampling," in *arXiv:2003.07847*, 2020.

[104] X. W. Y. Wang, K. Kitani, "Joint object detection and multi-object tracking with graph neural networks," in *arXiv:2006.13164 [cs.CV]*, 2020.

[105] J. Li, X. Gao, and T. Jiang, "Graph networks for multiple object tracking," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2020, pp. 719–728.

[106] G. Brasó and L. Leal-Taixé, "Learning a neural solver for multiple object tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6247–6257.

[107] D. Reid, "An algorithm for tracking multiple targets," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1979, pp. 843–854.

[108] W. Choi, "Near-online multi-target tracking with aggregated local flow descriptor," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 3029–3037.

[109] J. Ren, X. Chen, J. Liu, W. Sun, J. Pang, Q. Yan, Y.-W. Tai, and L. Xu, "Accurate single stage detector using recurrent rolling convolution," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5420–5428.

[110] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[111] K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: the clear mot metrics," *EURASIP Journal on Image and Video Processing*, vol. 2008, pp. 1–10, 2008.

[112] Y. Li, C. Huang, and R. Nevatia, "Learning to associate: Hybridboosted multi-target tracker for crowded scene," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2009, pp. 2953–2960.

[113] A. Shenoi, M. Patel, J. Gwak, P. Goebel, A. Sadeghian, H. Rezatofighi, R. Martín-Martín, and S. Savarese, "Jrmot: A real-time 3d multi-object tracker and a new large-scale dataset," in *The IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020. [Online]. Available: https://arxiv.org/abs/2002.08397

[114] N. F. Gonzalez, A. Ospina, and P. Calvez, "Smat: Smart multiple affinity metrics for multiple object tracking," in *Image Analysis and Recognition*, A. Campilho, F. Karray, and Z. Wang, Eds. Cham: Springer International Publishing, 2020, pp. 48–62.

[115] W. Zhang, H. Zhou, S. Sun, Z. Wang, J. Shi, and C. C. Loy, "Robust multi-modality multi-object tracking," in *International Conference on Computer Vision (ICCV)*, October 2019.

[116] S. Ulbrich and M. Maurer, "Towards tactical lane change behavior planning for automated vehicles," in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*. IEEE, 2015, pp. 989–995.

[117] J. Nilsson, J. Silvlin, M. Brannstrom, E. Coelingh, and J. Fredriksson, "If, when, and how to perform lane change maneuvers on highways," *IEEE Intelligent Transportation Systems Magazine*, vol. 8, no. 4, pp. 68–78, 2016.

[118] S. Sivaraman and M. M. Trivedi, "Dynamic probabilistic drivability maps for lane change and merge driver assistance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 5, pp. 2063–2073, 2014.

[119] H.-S. Tan and J. Huang, "Dgps-based vehicle-to-vehicle cooperative collision warning: Engineering feasibility viewpoints," *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, no. 4, pp. 415–428, 2006.

[120] M. R. Hafner, D. Cunningham, L. Caminiti, and D. Del Vecchio, "Cooperative collision avoidance at intersections: Algorithms and experiments," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 3, pp. 1162–1175, 2013.

[121] S. Sivaraman and M. M. Trivedi, "Towards cooperative, predictive driver assistance," in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. IEEE, 2013, pp. 1719–1724.

[122] S. Ammoun and F. Nashashibi, "Real time trajectory prediction for collision risk estimation between vehicles," in *2009 IEEE 5th International Conference on Intelligent Computer Communication and Processing*. IEEE, 2009, pp. 417–422.

[123] N. Kaempchen, K. Weiss, M. Schaefer, and K. C. Dietmayer, "Imm object tracking for high dynamic driving maneuvers," in *IEEE Intelligent Vehicles Symposium, 2004.* IEEE, 2004, pp. 825–830.

[124] J. Hillenbrand, A. M. Spieker, and K. Kroschel, "A multilevel collision mitigation approach—its situation assessment, decision making, and performance tradeoffs," *IEEE Transactions on intelligent transportation systems*, vol. 7, no. 4, pp. 528–540, 2006.

[125] A. Polychronopoulos, M. Tsogas, A. J. Amditis, and L. Andreone, "Sensor fusion for predicting vehicles' path for collision avoidance systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, no. 3, pp. 549–562, 2007.

[126] A. Barth and U. Franke, "Where will the oncoming vehicle be the next second?" in *2008 IEEE Intelligent Vehicles Symposium.* IEEE, 2008, pp. 1068–1073.

[127] R. Schubert, E. Richter, and G. Wanielik, "Comparison and evaluation of advanced motion models for vehicle tracking," in *2008 11th international conference on information fusion*. IEEE, 2008, pp. 1–6.

[128] D. Huang and H. Leung, "Em-imm based land-vehicle navigation with gps/ins," in *Proceedings. The 7th International IEEE Conference on Intelligent Transportation Systems (IEEE Cat. No. 04TH8749).* IEEE, 2004, pp. 624–629.

[129] R. Toledo-Moreo and M. A. Zamora-Izquierdo, "Imm-based lane-change prediction in highways with low-cost gps/ins," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 1, pp. 180–185, 2009.

[130] J. Wiest, M. Höffken, U. Kreßel, and K. Dietmayer, "Probabilistic trajectory prediction with gaussian mixture models," in *2012 IEEE Intelligent Vehicles Symposium.* IEEE, 2012, pp. 141–146.

[131] J. Joseph, F. Doshi-Velez, A. S. Huang, and N. Roy, "A bayesian nonparametric approach to modeling motion patterns," *Autonomous Robots*, vol. 31, no. 4, pp. 383–400, 2011.

[132] Q. Tran and J. Firl, "Online maneuver recognition and multimodal trajectory prediction for intersection assistance using non-parametric regression," in *2014 IEEE Intelligent Vehicles Symposium Proceedings.* IEEE, 2014, pp. 918–923.

[133] C. Laugier, I. E. Paromtchik, M. Perrollaz, M. Yong, J.-D. Yoder, C. Tay, K. Mekhnacha, and A. Nègre, "Probabilistic analysis of dynamic scenes and collision risks assessment to improve driving safety," *IEEE Intelligent Transportation Systems Magazine*, vol. 3, no. 4, pp. 4–19, 2011.

[134] J. Schlechtriemen, F. Wirthmueller, A. Wedel, G. Breuel, and K.-D. Kuhnert, "When will it change the lane? a probabilistic regression approach for rarely occurring events," in *2015 IEEE Intelligent Vehicles Symposium (IV).* IEEE, 2015, pp. 1373–1379.

[135] J. Schlechtriemen, A. Wedel, G. Breuel, and K.-D. Kuhnert, "A probabilistic long term prediction approach for highway scenarios," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2014, pp. 732–738.

[136] B. T. Morris and M. M. Trivedi, "Trajectory learning for activity understanding: Unsupervised, multilevel, and long-term adaptive approach," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 11, pp. 2287–2301, 2011.

[137] H. Berndt, J. Emmert, and K. Dietmayer, "Continuous driver intention recognition with hidden markov models," in *2008 11th International IEEE Conference on Intelligent Transportation Systems*. IEEE, 2008, pp. 1189–1194.

[138] H. M. Mandalia and M. D. D. Salvucci, "Using support vector machines for lane-change detection," in *Proceedings of the human factors and ergonomics society annual meeting*, vol. 49, no. 22. SAGE Publications Sage CA: Los Angeles, CA, 2005, pp. 1965–1969.

[139] G. S. Aoude, B. D. Luders, K. K. Lee, D. S. Levine, and J. P. How, "Threat assessment design for driver assistance system at intersections," in *13th International IEEE Conference on Intelligent Transportation Systems*. IEEE, 2010, pp. 1855–1862.

[140] A. Khosroshahi, E. Ohn-Bar, and M. M. Trivedi, "Surround vehicles trajectory analysis with recurrent neural networks," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2016, pp. 2267–2272.

[141] A. Houenou, P. Bonnifait, V. Cherfaoui, and W. Yao, "Vehicle trajectory prediction based on motion model and maneuver recognition," in *2013 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2013, pp. 4363–4369.

[142] M. Schreier, V. Willert, and J. Adamy, "Bayesian, maneuver-based, long-term trajectory prediction and criticality assessment for driver assistance systems," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2014, pp. 334–341.

[143] A. Doshi and M. M. Trivedi, "Tactical driver behavior prediction and intent inference: A review," in *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2011, pp. 1892–1897.

[144] A. Lawitzky, D. Althoff, C. F. Passenberg, G. Tanzmeister, D. Wollherr, and M. Buss, "Interactive scene prediction for automotive applications," in *2013 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2013, pp. 1028–1033.

[145] C. Hermes, C. Wohler, K. Schenk, and F. Kummert, "Long-term vehicle motion prediction," in *2009 IEEE intelligent vehicles symposium*. IEEE, 2009, pp. 652–657.

[146] D. Vasquez and T. Fraichard, "Motion prediction for moving objects: a statistical approach," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, vol. 4. IEEE, 2004, pp. 3931–3936.

[147] D. Vasquez, T. Fraichard, and C. Laugier, "Growing hidden markov models: An incremental tool for learning and predicting human and vehicle motion," *The International Journal of Robotics Research*, vol. 28, no. 11-12, pp. 1486–1506, 2009.

[148] N. Deo and M. M. Trivedi, "Learning and predicting on-road pedestrian behavior around vehicles," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2017, pp. 1–6.

[149] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains," *The annals of mathematical statistics*, vol. 41, no. 1, pp. 164–171, 1970.

[150] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[151] V. Beanland, M. Fitzharris, K. Young, and M. Lenné, "Erratum: Driver inattention and driver distraction in serious casualty crashes: Data from the australian national crash in-depth study (accident analysis and prevention 54c (2013) 99-107)," *Accident Analysis and Prevention*, vol. 59, p. 626, 2013.

[152] A. Rangesh, N. Deo, K. Yuen, K. Pirozhenko, P. Gunaratne, H. Toyoda, and M. Trivedi, "Exploring the situational awareness of humans inside autonomous vehicles," in *IEEE International Conference on Intelligent Transportation Systems*, 2018.

[153] M. M. Trivedi *et al.*, "Attention monitoring and hazard assessment with bio-sensing and vision: Empirical analysis utilizing cnns on the kitti dataset," in *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2019, pp. 1673–1678.

[154] S. Jha and C. Busso, "Probabilistic estimation of the gaze region of the driver using dense classification," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, Nov 2018, pp. 697–702.

[155] S. Martin, S. Vora, K. Yuen, and M. M. Trivedi, "Dynamics of driver's gaze: Explorations in behavior modeling and maneuver prediction," *IEEE Transactions on Intelligent Vehicles*, vol. 3, no. 2, p. 141–150, Jun 2018. [Online]. Available: http://dx.doi.org/10.1109/TIV.2018.2804160

[156] S. Vora, A. Rangesh, and M. M. Trivedi, "Driver gaze zone estimation using convolutional neural networks: A general framework and ablative analysis," *IEEE Transactions on Intelligent Vehicles*, vol. 3, no. 3, p. 254–265, Sep 2018. [Online]. Available: http://dx.doi.org/10.1109/TIV.2018.2843120

[157] H. S. Yoon, N. R. Baek, N. Q. Truong, and K. R. Park, "Driver gaze detection based on deep residual networks using the combined single image of dual near-infrared cameras," *IEEE Access*, vol. 7, pp. 93 448–93 461, 2019.

[158] N. Deo and M. M. Trivedi, "Looking at the driver/rider in autonomous vehicles to predict take-over readiness," *IEEE Transactions on Intelligent Vehicles*, vol. 5, no. 1, pp. 41–52, 2019.

[159] A. Roitberg, C. Ma, M. Haurilet, and R. Stiefelhagen, "Open set driver activity recognition," in *Intelligent Vehicles Symposium (IV)*. IEEE, 2020.

[160] M. Martin, A. Roitberg, M. Haurilet, M. Horne, S. Reiß, M. Voit, and R. Stiefelhagen, "Drive&act: A multi-modal dataset for fine-grained driver behavior recognition in autonomous vehicles," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2019.

[161] A. Rangesh, N. Deo, R. Greer, P. Gunaratne, and M. M. Trivedi, "Autonomous vehicles that alert humans to take-over controls: Modeling with real-world data," *arXiv preprint arXiv:2104.11489*, 2021.

[162] "Vision impact institute releases study on corrective lens wearers in the u.s." [Online]. Available: https://www.essilorusa.com/newsroom/vision-impact-institute-releases-study-on-corrective-lens-wearers-in-the-u-s

[163] "Vision impact institute stats." [Online]. Available: https://visionimpactinstitute.org/download-stats/

[164] "The most dangerous time to drive." [Online]. Available: https://www.nsc.org/road-safety/safety-topics/night-driving

[165] M. Gamadia, N. Kehtarnavaz, and K. Roberts-Hoffman, "Low-light auto-focus enhancement for digital and cell-phone camera image pipelines," *IEEE Transactions on Consumer Electronics*, vol. 53, no. 2, pp. 249–257, May 2007.

[166] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.

[167] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2017.19

[168] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2019. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2019.00453

[169] X. Huang, M.-Y. Liu, S. Belongie, and J. Kautz, "Multimodal unsupervised image-to-image translation," *Lecture Notes in Computer Science*, p. 179–196, 2018. [Online]. Available: http://dx.doi.org/10.1007/978-3-030-01219-9_11

[170] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. [Online]. Available: http://dx.doi.org/10.1109/ICCV.2017.244

[171] R. A. Naqvi, M. Arsalan, G. Batchuluun, H. S. Yoon, and K. R. Park, "Deep learning-based gaze detection system for automobile drivers using a nir camera sensor," *Sensors*, vol. 18, no. 2, p. 456, 2018.

[172] Y. Wang, G. Yuan, Z. Mi, J. Peng, X. Ding, Z. Liang, and X. Fu, "Continuous driver's gaze zone estimation using rgb-d camera," *Sensors*, vol. 19, no. 6, p. 1287, Mar 2019. [Online]. Available: http://dx.doi.org/10.3390/s19061287

[173] B. Hu, W. Yang, and M. Ren, "Unsupervised eyeglasses removal in the wild," 2019.

[174] Y. Wang, X. Ou, L. Tu, and L. Liu, "Effective facial obstructions removal with enhanced cycle-consistent generative adversarial networks," in *Artificial Intelligence and Mobile Services – AIMS 2018*, M. Aiello, Y. Yang, Y. Zou, and L.-J. Zhang, Eds. Cham: Springer International Publishing, 2018, pp. 210–220.

[175] M. LIANG, Y. XUE, K. XUE, and A. YANG, "Deep convolution neural networks for automatic eyeglasses removal," *DEStech Transactions on Computer Science and Engineering*, no. aiea, 2017.

[176] M. Li, W. Zuo, and D. Zhang, "Deep identity-aware transfer of facial attributes," 2016.

[177] W. Shen and R. Liu, "Learning residual images for face attribute manipulation," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2017.135

[178] A. Rangesh, B. Zhang, and M. M. Trivedi, "Driver gaze estimation in the real world: Overcoming the eyeglass challenge," *arXiv preprint arXiv:2002.02077*, 2020.

[179] A. Kar and P. Corcoran, "A review and analysis of eye-gaze estimation systems, algorithms and performance evaluation methods in consumer platforms," *IEEE Access*, vol. 5, p. 16495–16519, 2017. [Online]. Available: http://dx.doi.org/10.1109/ACCESS.2017.2735633

[180] S. Kaplan, M. A. Guvensan, A. G. Yavuz, and Y. Karalurt, "Driver behavior analysis for safe driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 6, pp. 3017–3032, Dec 2015.

[181] Y. Dong, Z. Hu, K. Uchimura, and N. Murayama, "Driver inattention monitoring system for intelligent vehicles: A review," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 2, pp. 596–614, June 2011.

[182] E. Murphy-Chutorian and M. M. Trivedi, "Head pose estimation in computer vision: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 4, p. 607–626, Apr. 2009. [Online]. Available: https://doi.org/10.1109/TPAMI.2008.106

[183] L. Fridman, P. Langhans, J. Lee, and B. Reimer, "Driver gaze region estimation without use of eye movement," *IEEE Intelligent Systems*, vol. 31, no. 3, p. 49–56, May 2016. [Online]. Available: http://dx.doi.org/10.1109/MIS.2016.47

[184] A. Tawari, K. Chen, and M. Trivedi, "Where is the driver looking: Analysis of head, eye and iris for robust gaze zone estimation," in *IEEE International Conference on Intelligent Transportation Systems*, 2014.

[185] E. Ohn-Bar, S. Martin, A. Tawari, and M. Trivedi, "Head, eye, and hand patterns for driver activity recognition," in *International Conference on Pattern Recognition*, 2014.

[186] W. Chinsatit and T. Saitoh, "Cnn-based pupil center detection for wearable gaze estimation system," *Applied Computational Intelligence and Soft Computing*, vol. 2017, 2017.

[187] A. Tawari and M. M. Trivedi, "Robust and continuous estimation of driver gaze zone by dynamic analysis of multiple face videos," in *2014 IEEE Intelligent Vehicles Symposium Proceedings*, June 2014, pp. 344–349.

[188] A. Tsukada, M. Shino, M. Devyver, and T. Kanade, "Illumination-free gaze estimation method for first-person vision wearable device," in *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, Nov 2011, pp. 2084–2091.

[189] S. J. Lee, J. Jo, H. G. Jung, K. R. Park, and J. Kim, "Real-time gaze estimator based on driver's head orientation for forward collision warning system," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 1, pp. 254–267, March 2011.

[190] D. Yi and S. Z. Li, "Learning sparse feature for eyeglasses problem in face recognition," in *Face and Gesture 2011*. IEEE, 2011, pp. 430–435.

[191] W. Wong and H. Zhao, "Eyeglasses removal of thermal image based on visible information," *Information Fusion*, vol. 14, no. 2, pp. 163 – 176, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1566253511000595

[192] N. U. Din, K. Javed, S. Bae, and J. Yi, "Effective removal of user-selected foreground object from facial images using a novel gan-based network," *IEEE Access*, vol. 8, pp. 109 648–109 661, 2020.

[193] Y.-H. Lee and S.-H. Lai, "Byeglassesgan: Identity preserving eyeglasses removal for face images," *Lecture Notes in Computer Science*, p. 243–258, 2020. [Online]. Available: http://dx.doi.org/10.1007/978-3-030-58526-6_15

[194] M. Amodio and S. Krishnaswamy, "Travelgan: Image-to-image translation by transformation vector learning," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2019. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2019.00919

[195] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, "OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields," in *arXiv preprint arXiv:1812.08008*, 2018.

[196] Y. Taigman, A. Polyak, and L. Wolf, "Unsupervised cross-domain image generation," 2016.

[197] B. A. Smith, Q. Yin, S. K. Feiner, and S. K. Nayar, "Gaze locking: Passive eye contact detection for human-object interaction," in *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 271–280. [Online]. Available: https://doi.org/10.1145/2501988.2501994

[198] J. Guo, X. Zhu, Z. Lei, and S. Z. Li, "Face synthesis for eyeglass-robust face recognition," *Lecture Notes in Computer Science*, p. 275–284, 2018. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-97909-0_30

[199] B. Hu, Z. Zheng, P. Liu, W. Yang, and M. Ren, "Unsupervised eyeglasses removal in the wild," *IEEE Transactions on Cybernetics*, pp. 1–13, 2020.

[200] N. Das, E. Ohn-Bar, and M. M. Trivedi, "On performance evaluation of driver hand detection algorithms: Challenges, dataset, and metrics," in *IEEE 18th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2015, pp. 2953–2958.

[201] P. Dollár, R. Appel, S. Belongie, and P. Perona, "Fast feature pyramids for object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 8, pp. 1532–1545, 2014.

[202] A. Rangesh, E. Ohn-Bar, and M. M. Trivedi, "Hidden hands: Tracking hands with an occlusion aware tracker," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2016, pp. 19–26.

[203] P. Molchanov, S. Gupta, K. Kim, and J. Kautz, "Hand gesture recognition with 3d convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2015, pp. 1–7.

[204] E. Ohn-Bar and M. M. Trivedi, "Hand gesture recognition in real time for automotive interfaces: A multimodal vision-based approach and evaluations," *IEEE transactions on intelligent transportation systems*, vol. 15, no. 6, pp. 2368–2377, 2014.

[205] N. Deo, A. Rangesh, and M. Trivedi, "In-vehicle hand gesture recognition using hidden markov models," in *IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2016, pp. 2179–2184.

[206] J. S. Supančič, G. Rogez, Y. Yang, J. Shotton, and D. Ramanan, "Depth-based hand pose estimation: methods, data, and challenges," *International Journal of Computer Vision*, vol. 126, no. 11, pp. 1180–1198, 2018.

[207] E. Ohn-Bar and M. Trivedi, "In-vehicle hand activity recognition using integration of regions," in *Intelligent Vehicles Symposium (IV), 2013 IEEE*. IEEE, 2013, pp. 1034–1039.

[208] E. Ohn-Bar and M. M. Trivedi, "Beyond just keeping hands on the wheel: Towards visual interpretation of driver hand motion patterns," in *IEEE 17th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2014, pp. 1245–1250.

[209] G. Borghi, F. Elia, R. Vezzani, and R. Cucchiara, "Hands on the wheel: a dataset for driver hand detection and tracking," in *8th International Workshop on Human Behavior Understanding (HBU)*, 2018.

[210] "libfreenect2," doi: https://doi.org/10.5281/zenodo.50641.

[211] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *European Conference on Computer Vision*. Springer, 2012, pp. 746–760.

[212] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2017, pp. 2980–2988.

[213] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[214] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[215] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[216] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[217] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1492–1500.

[218] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8697–8710.

[219] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," *arXiv preprint arXiv:1802.01548*, 2018.

[220] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," *arXiv preprint arXiv:1905.11946*, 2019.

[221] A. Rangesh, N. Deo, K. Yuen, K. Pirozhenko, P. Gunaratne, H. Toyoda, and M. M. Trivedi, "Exploring the situational awareness of humans inside autonomous vehicles," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 190–197.

[222] N. Deo and M. M. Trivedi, "Looking at the driver/rider in autonomous vehicles to predict take-over readiness," *arXiv preprint arXiv:1811.06047*, 2018.

[223] Ç. Gülçehre and Y. Bengio, "Knowledge matters: Importance of prior information for optimization," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 226–257, 2016.

[224] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2921–2929.

[225] C. Tran, A. Doshi, and M. M. Trivedi, "Pedal error prediction by driver foot gesture analysis: A vision-based inquiry," in *2011 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2011, pp. 577–582.

[226] ——, "Modeling and prediction of driver behavior by foot gesture analysis," *Computer Vision and Image Understanding*, vol. 116, no. 3, pp. 435–445, 2012.

[227] Y. Wu, L. N. Boyle, D. McGehee, C. A. Roe, K. Ebe, and J. Foley, "Foot placement during error and pedal applications in naturalistic driving," *Accident Analysis & Prevention*, vol. 99, pp. 102–109, 2017.

[228] Y. Wu, L. N. Boyle, and D. V. McGehee, "Evaluating variability in foot to pedal movements using functional principal components analysis," *Accident Analysis & Prevention*, vol. 118, pp. 146–153, 2018.

[229] X. Zeng and J. Wang, "A stochastic driver pedal behavior model incorporating road information," *IEEE Transactions on Human-Machine Systems*, vol. 47, no. 5, pp. 614–624, 2017.

[230] S. Frank and A. Kuijper, "Robust driver foot tracking and foot gesture recognition using capacitive proximity sensing," *Journal of Ambient Intelligence and Smart Environments*, vol. 11, no. 3, pp. 221–235, 2019.

[231] D. V. McGehee, C. A. Roe, L. N. Boyle, Y. Wu, K. Ebe, J. Foley, and L. Angell, "The wagging foot of uncertainty: data collection and reduction methods for examining foot pedal behavior in naturalistic driving," *SAE International journal of transportation safety*, vol. 4, no. 2, pp. 289–294, 2016.

[232] D.-Y. D. Wang, F. D. Richard, C. R. Cino, T. Blount, and J. Schmuller, "Bipedal vs. unipedal: a comparison between one-foot and two-foot driving in a driving simulator," *Ergonomics*, vol. 60, no. 4, pp. 553–562, 2017.

[233] E. Velloso, D. Schmidt, J. Alexander, H. Gellersen, and A. Bulling, "The feet in human–computer interaction: A survey of foot-based interaction," *ACM Computing Surveys (CSUR)*, vol. 48, no. 2, p. 21, 2015.

[234] M. Leo, G. Medioni, M. Trivedi, T. Kanade, and G. M. Farinella, "Computer vision for assistive technologies," *Computer Vision and Image Understanding*, vol. 154, pp. 1–15, 2017.

[235] G. M. Farinella, T. Kanade, M. Leo, G. G. Medioni, and M. Trivedi, "Special issue on assistive computer vision and robotics-part i," *Computer Vision and Image Understanding*, vol. 100, no. 148, pp. 1–2, 2016.

[236] K. K. Singh and Y. J. Lee, "Hide-and-seek: Forcing a network to be meticulous for weakly-supervised object and action localization," in *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2017, pp. 3544–3553.

[237] Y. Wei, J. Feng, X. Liang, M.-M. Cheng, Y. Zhao, and S. Yan, "Object region mining with adversarial erasing: A simple classification to semantic segmentation approach," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1568–1576.

[238] D. Kim, D. Cho, D. Yoo, and I. So Kweon, "Two-phase learning for weakly supervised object localization," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 3534–3543.

[239] K. Li, Z. Wu, K.-C. Peng, J. Ernst, and Y. Fu, "Tell me where to look: Guided attention inference network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9215–9223.

[240] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5mb model size," *arXiv:1602.07360*, 2016.

[241] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 618–626.

[242] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[243] I. Kotseruba and J. K. Tsotsos, "Behavioral research and practical models of drivers' attention," *arXiv preprint arXiv:2104.05677*, 2021.

[244] A. Eriksson and N. A. Stanton, "Takeover time in highly automated vehicles: noncritical transitions to and from manual control," *Human factors*, vol. 59, no. 4, pp. 689–705, 2017.

[245] E. Dogan, M.-C. Rahal, R. Deborne, P. Delhomme, A. Kemeny, and J. Perrin, "Transition of control in a partially automated vehicle: Effects of anticipation and non-driving-related task involvement," *Transportation research part F: traffic psychology and behaviour*, vol. 46, pp. 205–215, 2017.

[246] N. Du, J. Kim, F. Zhou, E. Pulver, D. M. Tilbury, L. P. Robert, A. K. Pradhan, and X. J. Yang, "Evaluating effects of cognitive load, takeover request lead time, and traffic density on drivers' takeover performance in conditionally automated driving," in *12th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, 2020, pp. 66–73.

[247] S. Ma, W. Zhang, Z. Yang, C. Kang, C. Wu, C. Chai, J. Shi, and H. Li, "Promote or inhibit: An inverted u-shaped effect of workload on driver takeover performance," *Traffic injury prevention*, vol. 21, no. 7, pp. 482–487, 2020.

[248] S.-A. Kaye, S. Demmel, O. Oviedo-Trespalacios, W. Griffin, and I. Lewis, "Young drivers' takeover time in a conditional automated vehicle: The effects of hand-held mobile phone use and future intentions to use automated vehicles," *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 78, pp. 16–29, 2021.

[249] H. Clark and J. Feng, "Age differences in the takeover of vehicle control and engagement in non-driving-related activities in simulated driving with conditional automation," *Accident Analysis & Prevention*, vol. 106, pp. 468–479, 2017.

[250] A. Roitberg, M. Haurilet, S. Reiß, and R. Stiefelhagen, "Cnn-based driver activity understanding: Shedding light on deep spatiotemporal representations," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2020, pp. 1–6.

[251] E. Ohn-Bar, A. Tawari, S. Martin, and M. M. Trivedi, "On surveillance for safety critical events: In-vehicle video networks for predictive driver assistance systems," *Computer Vision and Image Understanding*, vol. 134, pp. 130–140, 2015.

[252] S. Y. Cheng, S. Park, and M. M. Trivedi, "Multi-spectral and multi-perspective video arrays for driver body tracking and activity analysis," *Computer Vision and Image Understanding*, vol. 106, no. 2, pp. 245–257, 2007.

[253] A. Tawari, K. H. Chen, and M. M. Trivedi, "Where is the driver looking: Analysis of head, eye and iris for robust gaze zone estimation," in *17th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2014, pp. 988–994.

[254] F. Naujoks, C. Purucker, K. Wiedemann, and C. Marberger, "Noncritical state transitions during conditionally automated driving on german freeways: Effects of non–driving related tasks on takeover time and takeover quality," *Human factors*, vol. 61, no. 4, pp. 596–613, 2019.

[255] M. Martin, A. Roitberg, M. Haurilet, M. Horne, S. Reiß, M. Voit, and R. Stiefelhagen, "Drive&act: A multi-modal dataset for fine-grained driver behavior recognition in autonomous vehicles," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 2801–2810.

[256] S. Vora, A. Rangesh, and M. M. Trivedi, "On generalizing driver gaze zone estimation using convolutional neural networks," in *Intelligent Vehicles Symposium (IV), 2017 IEEE*. IEEE, 2017, pp. 849–854.

[257] A. Rangesh, B. Zhang, and M. M. Trivedi, "Gaze preserving cyclegans for eyeglass removal & persistent gaze estimation," *arXiv preprint arXiv:2002.02077*, 2020.

[258] K. Yuen and M. M. Trivedi, "Looking at hands in autonomous vehicles: A convnet approach using part affinity fields," *arXiv preprint arXiv:1804.01176*, 2018.

[259] A. Rangesh and M. Trivedi, "Forced spatial attention for driver foot activity classification," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019, pp. 0–0.

[260] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. Li, "Pv-rcnn: Point-voxel feature set abstraction for 3d object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10 529–10 538.

[261] J. Deng, S. Shi, P. Li, W. Zhou, Y. Zhang, and H. Li, "Voxel r-cnn: Towards high performance voxel-based 3d object detection," *arXiv preprint arXiv:2012.15712*, 2020.

[262] W. Zheng, W. Tang, L. Jiang, and C.-W. Fu, "Se-ssd: Self-ensembling single-stage object detector from point cloud," *arXiv preprint arXiv:2104.09804*, 2021.

[263] Y. Chen, S. Liu, X. Shen, and J. Jia, "Dsgn: Deep stereo geometry network for 3d object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 12 536–12 545.

[264] Y. You, Y. Wang, W.-L. Chao, D. Garg, G. Pleiss, B. Hariharan, M. Campbell, and K. Q. Weinberger, "Pseudo-lidar++: Accurate depth for 3d object detection in autonomous driving," *arXiv preprint arXiv:1906.06310*, 2019.

[265] Y. Wang, W.-L. Chao, D. Garg, B. Hariharan, M. Campbell, and K. Q. Weinberger, "Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8445–8453.

[266] H.-N. Hu, Q.-Z. Cai, D. Wang, J. Lin, M. Sun, P. Krahenbuhl, T. Darrell, and F. Yu, "Joint monocular 3d vehicle detection and tracking," in *Proceedings of the IEEE international conference on computer vision*, 2019, pp. 5390–5399.

[267] D. Mykheievskyi, D. Borysenko, and V. Porokhonskyy, "Learning local feature descriptors for multiple object tracking," in *Proceedings of the Asian Conference on Computer Vision*, 2020.

[268] J. Luiten, A. Osep, P. Dendorfer, P. Torr, A. Geiger, L. Leal-Taixé, and B. Leibe, "Hota: A higher order metric for evaluating multi-object tracking," *International journal of computer vision*, vol. 129, no. 2, pp. 548–578, 2021.

[269] J. Gao, C. Sun, H. Zhao, Y. Shen, D. Anguelov, C. Li, and C. Schmid, "Vectornet: Encoding hd maps and agent dynamics from vectorized representation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 525–11 533.

[270] M. Liang, B. Yang, R. Hu, Y. Chen, R. Liao, S. Feng, and R. Urtasun, "Learning lane graph representations for motion forecasting," in *European Conference on Computer Vision*. Springer, 2020, pp. 541–556.

[271] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social gan: Socially acceptable trajectories with generative adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2255–2264.

[272] N. Rhinehart, R. McAllister, K. Kitani, and S. Levine, "Precog: Prediction conditioned on goals in visual multi-agent settings," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 2821–2830.

[273] S. Martin, A. Tawari, and M. M. Trivedi, "Toward privacy-protecting safety systems for naturalistic driving videos," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 4, pp. 1811–1822, 2014.

[274] A. Roitberg, M. Haurilet, M. Martinez, and R. Stiefelhagen, "Uncertainty-sensitive activity recognition: A reliability benchmark and the caring models," in *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021, pp. 3814–3821.

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

[276] J. Choi, S. Ulbrich, B. Lichte, and M. Maurer, "Multi-target tracking using a 3d-lidar sensor for autonomous vehicles," in *16th International IEEE Conference on Intelligent Transportation Systems-(ITSC)*. IEEE, 2013, pp. 881–886.

[277] S. Song, Z. Xiang, and J. Liu, "Object tracking with 3d lidar via multi-task sparse learning," in *IEEE International Conference on Mechatronics and Automation (ICMA)*. IEEE, 2015, pp. 2603–2608.

[278] A. Asvadi, P. Peixoto, and U. Nunes, "Detection and tracking of moving objects using 2.5 d motion grids," in *IEEE 18th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2015, pp. 788–793.

[279] A. Asvadi, C. Premebida, P. Peixoto, and U. Nunes, "3d lidar-based static and moving obstacle detection in driving environments: An approach based on voxels and multi-region ground planes," *Robotics and Autonomous Systems*, vol. 83, pp. 299–311, 2016.

[280] D. Pfeiffer and U. Franke, "Efficient representation of traffic scenes by means of dynamic stixels," in *Intelligent Vehicles Symposium (IV), 2010 IEEE*. IEEE, 2010, pp. 217–224.

[281] A. Broggi, S. Cattani, M. Patander, M. Sabbatelli, and P. Zani, "A full-3d voxel-based dynamic obstacle detection for urban scenario using stereo vision," in *16th International IEEE Conference on Intelligent Transportation Systems-(ITSC)*. IEEE, 2013, pp. 71–76.

[282] A. Vatavu, R. Danescu, and S. Nedevschi, "Stereovision-based multiple object tracking in traffic scenarios using free-form obstacle delimiters and particle filters," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 1, pp. 498–511, 2015.

[283] A. Ošep, A. Hermans, F. Engelmann, D. Klostermann, M. Mathias, and B. Leibe, "Multi-scale object candidates for generic object tracking in street scenes," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 3180–3187.

[284] R. K. Satzoda, S. Lee, F. Lu, and M. M. Trivedi, "Vision based front & rear surround understanding using embedded processors," *IEEE Transactions on Intelligent Vehicles*, 2017.

[285] H. Cho, Y.-W. Seo, B. V. Kumar, and R. R. Rajkumar, "A multi-sensor fusion system for moving object detection and tracking in urban driving environments," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 1836–1843.

[286] A. Asvadi, P. Girão, P. Peixoto, and U. Nunes, "3d object tracking using rgb and lidar data," in *IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2016, pp. 1255–1260.

[287] M. Allodi, A. Broggi, D. Giaquinto, M. Patander, and A. Prioletti, "Machine learning in tracking associations with stereo vision and lidar observations for an autonomous vehicle," in *Intelligent Vehicles Symposium (IV), 2016 IEEE*. IEEE, 2016, pp. 648–653.

[288] S.-H. Bae and K.-J. Yoon, "Robust online multi-object tracking based on tracklet confidence and online discriminative appearance learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1218–1225.

[289] S. Kim, S. Kwak, J. Feyereisl, and B. Han, "Online multi-target tracking by large margin structured learning," in *Asian Conference on Computer Vision*. Springer, 2012, pp. 98–111.

[290] C.-H. Kuo, C. Huang, and R. Nevatia, "Multi-target tracking by on-line learned discriminative appearance models," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2010, pp. 685–692.

[291] X. Song, J. Cui, H. Zha, and H. Zhao, "Vision-based multiple interacting targets tracking via on-line supervised learning," in *European Conference on Computer Vision*. Springer, 2008, pp. 642–655.

[292] B. Babenko, M.-H. Yang, and S. Belongie, "Robust object tracking with online multiple instance learning," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 8, pp. 1619–1632, 2011.

[293] C. Bao, Y. Wu, H. Ling, and H. Ji, "Real time robust l1 tracker using accelerated proximal gradient approach," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2012, pp. 1830–1837.

[294] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks, and P. H. Torr, "Struck: Structured output tracking with kernels," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 10, pp. 2096–2109, 2016.

[295] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, pp. 1409–1422, 2012.

[296] J. Berclaz, F. Fleuret, E. Turetken, and P. Fua, "Multiple object tracking using k-shortest paths optimization," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 9, pp. 1806–1819, 2011.

[297] A. A. Butt and R. T. Collins, "Multi-target tracking by lagrangian relaxation to min-cost network flow," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 1846–1853.

[298] A. Milan, S. Roth, and K. Schindler, "Continuous energy minimization for multitarget tracking," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 1, pp. 58–72, 2014.

[299] J. C. Niebles, B. Han, and L. Fei-Fei, "Efficient extraction of human motion volumes by tracking," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2010, pp. 655–662.

[300] H. Pirsiavash, D. Ramanan, and C. C. Fowlkes, "Globally-optimal greedy algorithms for tracking a variable number of objects," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2011, pp. 1201–1208.

[301] L. Zhang, Y. Li, and R. Nevatia, "Global data association for multi-object tracking using network flows," in *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2008, pp. 1–8.

[302] Z. Khan, T. Balch, and F. Dellaert, "Mcmc-based particle filtering for tracking a variable number of interacting targets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 11, pp. 1805–1819, 2005.

[303] S. Oh, S. Russell, and S. Sastry, "Markov chain monte carlo data association for multi-target tracking," *IEEE Transactions on Automatic Control*, vol. 54, no. 3, pp. 481–497, 2009.

[304] K. Okuma, A. Taleghani, N. d. Freitas, J. J. Little, and D. G. Lowe, "A boosted particle filter: Multitarget detection and tracking," *Computer Vision-ECCV 2004*, pp. 28–39, 2004.

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

[306] M. D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. Van Gool, "Online multiperson tracking-by-detection from a single, uncalibrated camera," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 9, pp. 1820–1833, 2011.

[307] A. Rangesh and M. M. Trivedi, "Ground plane polling for 6dof pose estimation of objects on the road," *arXiv preprint arXiv:1811.06666*, 2018.

[308] A. Milan, K. Schindler, and S. Roth, "Challenges of ground truth evaluation of multi-target tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2013, pp. 735–742.

[309] R. N. Rajaram, E. Ohn-Bar, and M. M. Trivedi, "Refinenet: Refining object detectors for autonomous driving," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 4, pp. 358–368, 2016.

[310] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *arXiv preprint arXiv:1708.02002*, 2017.

[311] E. Ohn-Bar and M. M. Trivedi, "Fast and robust object detection using visual subcategories," in *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, 2014, pp. 179–184.

[312] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.