UNIVERSITY OF CALIFORNIA
RIVERSIDE

A Connected Automation Enabled Cooperative Management Framework for Mixed Traffic

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Electrical Engineering

by

Zhouqiao Zhao

June 2023

Dissertation Committee:
Dr. Matthew Barth, Co-Chairperson
Dr. Guoyuan Wu, Co-Chairperson
Dr. Hamed Mohsenian-Rad

The Dissertation of Zhouqiao Zhao is approved:

_____

_____

_____

Committee Chairperson

University of California, Riverside

ACKNOWLEDGMENT

Firstly, I extend my deepest gratitude to the Center for Environmental Research & Technology (CE-CERT) and the Transportation System Research (TSR) Group at UCR, along with the Department of Electrical Engineering. Throughout the five years of my PhD journey, they have offered me unparalleled support, steady financial backing, and a rich repository of research resources. I specifically want to thank my Ph.D. co-advisors, Dr. Matthew J. Barth and Dr. Guoyuan Wu. Dr. Barth provided me with a nurturing research environment, consistently proposing the most constructive suggestions during the critical stages of my study. Dr. Wu has been an invaluable guide throughout my research, helping me establish my research direction and mentoring my evolution from a beginner to an experienced researcher. I also wish to express my appreciation for Dr. Hamed Mohsenian-Rad for his service in the dissertation committee. His course on convex optimization has significantly contributed to my research throughout my PhD journey.

I am also thankful to the research faculties and technical staff within my research group, including Dr. Peng Hao, Dr. Kanok Boriboonsomsin, Mike Todd, Dr. George Scora, Alexander Vu, Henry Gomez, and Dr. Ji Luo. Their management and collaboration on the projects I were involved in have been instrumental to my growth. I owe a lot to my seniors and peers, such as Dr. Ziran Wang, Dr. Danyang Tian, Dr. Fei Ye, Dr. Nigel Williams, Dr. Chao Wang, and Xishun Liao, Zhensong Wei, Roland David Oswald, Abdullah Fuad Un-Noor, among others. Their guidance, akin to that of a mentor, has helped me quickly settle into my research projects. I also wish to acknowledge my other colleagues in our vibrant lab, including Zhenwei Bai, Saswat Priyadarshi Nayak, Yejia Liao, Jacqueline Garrido

Escobar, Xuanpeng Zhao, Haishan Liu, Dongbo Peng, and Chuheng Wei. Our collective discussions and shared growth have been a constant source of motivation.

Finally, I would like to acknowledge the Department of Electrical and Computer Engineering at UCR, College of Engineering-Center for Environmental Research and Technology (CE- CERT) at UCR, United States Department of Energy, United States Department of Transportation, National Center for Sustainable Transportation (NCST), Toyota Motor North America, Honda Research Institute, Volvo Group North America, and Esther F. Hays family, for their full or partial support towards my study and research encapsulated in this dissertation.

The major contents in this dissertation have been published in the Institute of Electrical and Electronics Engineers (IEEE) Internet of Things Journal, IEEE International Conference on Robotics and Automation, IEEE Intelligent Transportation Systems Conference, IEEE Intelligent Vehicles Symposium, IEEE Conference on Technologies for Sustainability, MDPI Sustainability, Transportation Research Record, SAE World Congress, and Transportation Research Board Annual Meeting, respectively.

DEDICATION

This dissertation is dedicated to the world I deeply love, and to the people whom I might not love as much. Special recognition goes to my partner and soulmate, Jun Zhou (周珺). I am truly grateful for our shared experiences and the growth we've achieved together. I would also like to dedicate this work to our cat, Hugo. Even though he might not comprehend it, his companionship was indispensable for me to smoothly navigate my Ph.D. journey, especially during the challenging times of the COVID-19 pandemic.

Lastly, but most importantly, I dedicate this dissertation to my parents, Haifeng Zhou (周海峰) and Hongtian Zhao (赵宏天). I may not be thankful that they brought me into this world, but I am genuinely grateful for their nurturing and education during the early stages of my life. Their care and cultivation of me have been my privilege. Without their contribution, I would not be who I am today.

ABSTRACT OF THE DISSERTATION

A Connected Automation Enabled Cooperative Management Framework for Mixed Traffic

by

Zhouqiao Zhao

Doctor of Philosophy, Graduate Program in Electrical Engineering
University of California, Riverside, June 2023
Dr. Matthew J. Barth, Co-Chairperson
Dr. Guoyuan Wu, Co-Chairperson

Safety, mobility, and environmental sustainability form the triad of challenges in modern transportation systems. To tackle these issues, there has been an increasing emphasis on Intelligent Transportation Systems (ITS) technology, which employs interdisciplinary approaches to provide effective solutions. Transportation systems are characterized by their large scale, non-linearity, time-varying behavior, interconnectivity, heterogeneity, and distributed nature, with various participants engaging in intensive interactions.

As the fields of sensing, communication, and control techniques advance, we expect a significant rise in connected automation applications in the transportation system. This development will likely lead to an increased presence of Autonomous Vehicles (AVs) and Connected and Automated Vehicles (CAVs). However, these emerging automated

systems will coexist with traditional human-driven traffic for an extended period, making cooperation in mixed traffic conditions a crucial yet challenging research topic.

In this dissertation, a cooperative framework for mixed traffic is proposed at both macroscopic and microscopic levels. Extend CDA definition for mixed traffic environment. At the macroscopic level, shared automated mobility applications are explored by formulate it as a mixed integer programming problem and consider demand-side cooperation, and dispatching and scheduling algorithms, based on ant copoline optimization, for battery electric truck fleets are investigated to improve operational efficiency. On a microscopic level, a corridor-wise ramp management framework is introduced to handle the unique challenges of mixed traffic. Moreover, with the inclusion of non-CAV agents such as traditional vehicles in mixed traffic, the research pursues two key directions. First, it examines infrastructure-side sensing to improve detection and monitoring capabilities, thereby enhancing decision-making and overall system management. Second, it undertakes the modeling of driver behavior to better understand and predict human actions in mixed traffic scenarios, which could lead to improved traffic flow management.

The key contributions of this dissertation are listed as follows:

- The definition of Cooperative Driving Automation (CDA) is extended to account for the unique characteristics of mixed traffic environments.
- The dissertation formulates shared automated mobility as a mixed integer programming problem, optimizing the utilization of shared mobility services considering demand-side cooperation.

- Dispatching and scheduling algorithms are developed for battery electric truck fleets using ant colony optimization and recursive algorithms, improving their operational efficiency.
- A corridor-wise ramp management framework based on model predictive control is introduced to address the challenges of mixed traffic at ramp areas.
- A dynamic background subtraction-based method is proposed to achieve lane-level infrastructure-side sensing using Fisheye cameras.
- An inverse reinforcement learning-based car-following model and digital twin framework are developed to understand and predict human behavior in mixed traffic scenarios.

TABLE OF CONTENTS

LIST OF TABLES xv

LIST OF FIGURES xvi

1 INTRODUCTION 1

1.1 Motivation 1

1.2 Trends and Opportunities 3

1.3 Challenges and Research Gaps 4

1.4 Research Objectives and Contribution of the Dissertation Research 6

1.5 Organization of the Dissertation 7

2 CATEGORIZATION AND DEFINITION TO COOPERATION IN TRANSPORTATION SYSTEMS 8

2.1 Macroscopic Level – Supply vs. Demand 8

2.2 Microscopic Level – Cooperative Driving Automation (CDA) 10

3 REVIEW OF VARIOUS FRAMEWORKS 16

3.1 Traffic State Estimation 16

3.1.1 Model-Based Approaches 18

3.1.2 Learning-Based Approaches 18

3.1.3 Streaming-Data-Driven Approaches 19

3.2 Infrastructure-Side Sensing 19

3.2.1 Vision-Based Perception 20

3.2.2 Sensor Selection and Challenge 23

3.2.3 Fisheye Camera Applications 23

3.3 Driving Behavior Modeling 24

3.3.1 Car-Following Model 25

3.4 Optimization and Control 27

3.4.1 Centralized Approaches 27
3.4.2 Distributed Approaches 29
4 MACROSCOPIC COOPERATION 31
4.1 Shared Automated Mobility With Demand-Side Cooperation 31
4.1.1 Introduction and Background 31
4.1.2 System Framework 35
4.1.2.1 System Framework in Simulation 35
4.1.2.2 Alternative PUDO Locations and Ride Matching 40
4.1.3 Methodology 41
4.1.3.1 Heuristic Model 41
4.1.3.2 Optimization Model With Demand-Side Cooperation (ODC) 42
4.1.3.3 Optimization Model Without Demand-Side Cooperation (ONDC) 46
4.1.4 Case Study 46
4.1.4.1 Determination of System Optimization Time Window 50
4.1.4.2 Comparison of Different Ride Matching Strategies 51
4.1.4.2 Comparison of Different SAM Service Demand 53
4.1.4.3 Comparison of Different Vehicle Capacity 54
4.1.5 Summary 55
4.2 Fleet Dispatching Strategy for Battery-Electric Trucks (BET) 56
4.2.1 Introduction and Background 56
4.2.2 Problem Formulation 59
4.2.3 Methodology 65
4.2.3.1 Coarse Scheduling 65

4.2.3.2 Fine Scheduling 67

4.2.3.3 Heuristic Algorithm 70

4.2.4 Case Study 71

4.2.5 Summary 78

5 MICROSCOPIC COOPERATION 80

5.1 Corridor-Wise Eco-Friendly Cooperative Ramp Management System for Connected and Automated Vehicles 80

5.1.1 Introduction and Background 80

5.1.2 Problem Formulation and System Architecture 82

5.1.2.1 Problem Formula Assumptions 82

5.1.2.2 Vehicle Dynamics 83

5.1.2.3 Optimization Problem Formulation 85

5.1.2.4 System Architecture 86

5.1.3 Methodology 88

5.1.3.1 Corridor-Level: Metering Rate Estimation 88

5.1.3.2 Ramp-Level: Movement Control and Rate Regulation 90

5.1.4 Simulation Study 96

5.1.4.1 Results Comparison for Gasoline Vehicles 97

5.1.4.2 Results Comparison for Electric Vehicles 100

5.1.5 Summary 102

6 SOLVING THE MIXED TRAFFIC PROBLEM 104

6.1 Infrastructure-Side Sensing 104

6.1.1 Introduction and Background 104

6.1.2 System Architecture 107
6.1.3 Methodology 109
6.1.3.1 Fisheye Camera Calibration 110
6.1.3.2 BS-Based Object Detection 113
6.1.3.3 Object Tracking 117
6.1.3.4 Localization 117
6.1.4 Experiments 119
6.1.4.1 BS-Based Detection Algorithm Validation 119
6.1.4.2 Simulation Experiment 122
6.1.4.3 Real-World Experiment 124
6.1.5 Summary 127
6.2 Driver Behavior Modeling 129
6.2.1 Introduction and Background 129
6.2.2 Problem Formulation and System Architecture 132
6.2.2.1 Digital Twin Framework 132
6.2.2.2 P-ACC With Online Adaptation 134
6.2.2.3 Assumptions and Specifications 135
6.2.3 Methodology 137
6.2.4 Numerical Experiment 150
6.2.5 Simulation Study 155
6.2.6 Summary 161
7 Conclusions and Future Work 162

7.1 Conclusions 162

7.2 Selected Publications Resulting from This Research 163

7.3 Future Work 165

Bibliography 167

LIST OF TABLES

Table 2-1 Relationship between classes of CDA cooperation and levels of automation [8] 11

Table 4-1 Key service performance metrics. 47

Table 4-2 Sensitivity analysis results on system optimization time window 51

Table 4-3 Simulation results for three strategies 53

Table 4-4 Simulation results for ODC (Optimization Model without Demand-Side Cooperation) scenarios with different demand levels 54

Table 4-5 Simulation results for ODC in different seat capacities 55

Table 4-6 Variable definitions 61

Table 4-7 Case study setup 72

Table 4-8 Cost comparison for different algorithms 78

Table 5-1 Simulation Results of Mobility and Energy Performance for Gasoline Vehicles 99

Table 5-2 Simulation Results for Electric Vehicles 101

Table 6-1 Dynamic background generation performance on CDNet2014 120

Table 6-2 BS-Based detection performance on simulation environment 121

Table 6-3 Vehicle tracking performance on simulation environment 121

Table 6-4 Quantitative results of real-world experiment 127

Table 6-5 Quantitative results of numerical simulation 154

Table 6-6 Results of Human-in-the-Loop simulation: Percentage-of-Interruption (PoI) during a 300-Sec trip 159

Table 6-7 Effectiveness of online adaptation 160

LIST OF FIGURES

Figure 1-1 Average hours lost to congestion per driver in major U.S. cities. [3] 2

Figure 1-2 Road traffic deaths by type of road user. [1] 2

Figure 1-3 US energy consumption by sector. [4] 4

Figure 2-1 Multi-Modal Cooperative for Transportation System. 9

Figure 2-2 3-Dimensional definition of cooperative driving automation. 12

Figure 4-1 The system framework of demand-side cooperative share automated mobility (DC-SAM) service. [136] 37

Figure 4-2 An example illustrating alternative pickup locations. 38

Figure 4-3 SUMO (Simulation of Urban Mobility) for the New York City network. 48

Figure 4-4 Graph model (the arcs within set V and set C are omitted, the arcs between set V and set C are simplified). 59

Figure 4-5 Illustration of fine scheduling process. 68

Figure 4-6 Convergence curve of ACO algorithm. 73

Figure 4-7 Shortest route giving by ACO algorithm. 75

Figure 4-8 Travel distance – time plot giving by ACO algorithm. 77

Figure 4-9 Travel distance – time plot giving by Fine Scheduling algorithm. 78

Figure 5-1 System Architecture of the Ramp Management System. 87

Figure 5-2 Flowchart of corridor-wise ramp metering rate determination. 88

Figure 5-3 Definition of the zones of the proposed ramp-level control. 90

Figure 5-4 Flowchart of the vehicle grouping. 91

Figure 5-5 Trajectories of the CAVs for the three cases. 97

Figure 5-6 Distributions of TTC. 102

Figure 6-1 System architecture of the infrastructure-side sensing. 105

Figure 6-2 Roadside Perception Unit (RSPU) and communication topology. 106

Figure 6-3 RSPU Graphic User Interface. 107

Figure 6-4 Illustration of auto-calibration process. (a) Extracted trajectories in raw image. (b) Extracted trajectories after undistortion with estimated calibration parameters. 109

Figure 6-5 The cascaded structure of the background maintenance. 113

Figure 6-6 Chassis Center Estimation. (a) Projection error. (b) Illustration of contour center and chassis center. 115

Figure 6-7 Illustration of virtual fisheye image synthesis. (a) The stitched cubic camera image with 90-degree of FOV from each perspective. (b) The projection relation from hemisphere to the image plane synthesized fisheye image. (c) Raw fisheye image. (d) Undistorted image using real-world calibration parameters. 116

Figure 6-8 Qualitative Results for Real-world experiment. (a) Detected bounding boxes. (b) Detected trajectories. (c) Trajectory of the ego vehicle, ground truth in blue and detection in red. 126

Figure 6-9 Result of customer survey regarding ACC usage. 129

Figure 6-10 System architecture: Digital twin framework. 130

Figure 6-11 System architecture: Offline learning (blue blocks), Online learning (orange blocks), Personalized controller (green blocks). 131

Figure 6-12 The training process of the maximum entropy IRL (Max-Ent IRL) used for static car-following preference modeling. 141

Figure 6-13 Illustration of drivers' dynamic preference. 143

Figure 6-14 Flowchart of online DGPT adaptation algorithm 144

Figure 6-15 Example trajectories of the naturalistic driving data collected by our 149

Figure 6-16 Simulated trajectories of numerical experiments. 150

Figure 6-17 Static driving preference from IRL modeling: (a) recovered reward function from naturalistic driving data using IRL, (b) smoothed DGPT. 151

Figure 6-18 Human driver vs. P-ACC. 152

Figure 6-19 Human-in-the-loop simulator. 153

Figure 6-20 Three simulated weather conditions in the game engine: Clear Sky (Day), Clear Sky (Night), and Foggy. 155

Figure 6-21 Driver's interruption during the trip. 156

Figure 6-22 Static preference of Driver A in three simulated weather conditions. 157

1 INTRODUCTION

1.1 Motivation

Traveling and freight shipping are basic needs in modern society. As a result, the size of the transportation network is rapidly enlarging, and the number of vehicles is also quickly growing. Take the United State as an example, the total length of the road network is nearly 7 million kilometers, and the number of vehicles is nearly 300 million [1]. In such a huge system, a mass of issues are threatening the health of the system operation which are focused on three key aspects — safety, mobility, and environmental sustainability. From the safety perspective, approximately 1.35 million people die each year due to the traffic crashes which ranks eighth in the top reason for death [2], as shown in Figure 1-1. On the mobility side, congestion costs on average each American 97 hours a year on the road as shown in Figure 1-2, and the 'last mile' speed of New York City is as low as 9 miles per hour [3]. With respect to environmental sustainability, the energy consumption for transportation accounts for 29% of the total energy consumption in the United States, and 91% of the transportation energy use is from petroleum products which are not renewable energy [4] as shown in Figure 1-3. To deal with such difficult situations, optimizing the current transportation system is imperative, and cooperation between participants in the system is a promising solution.

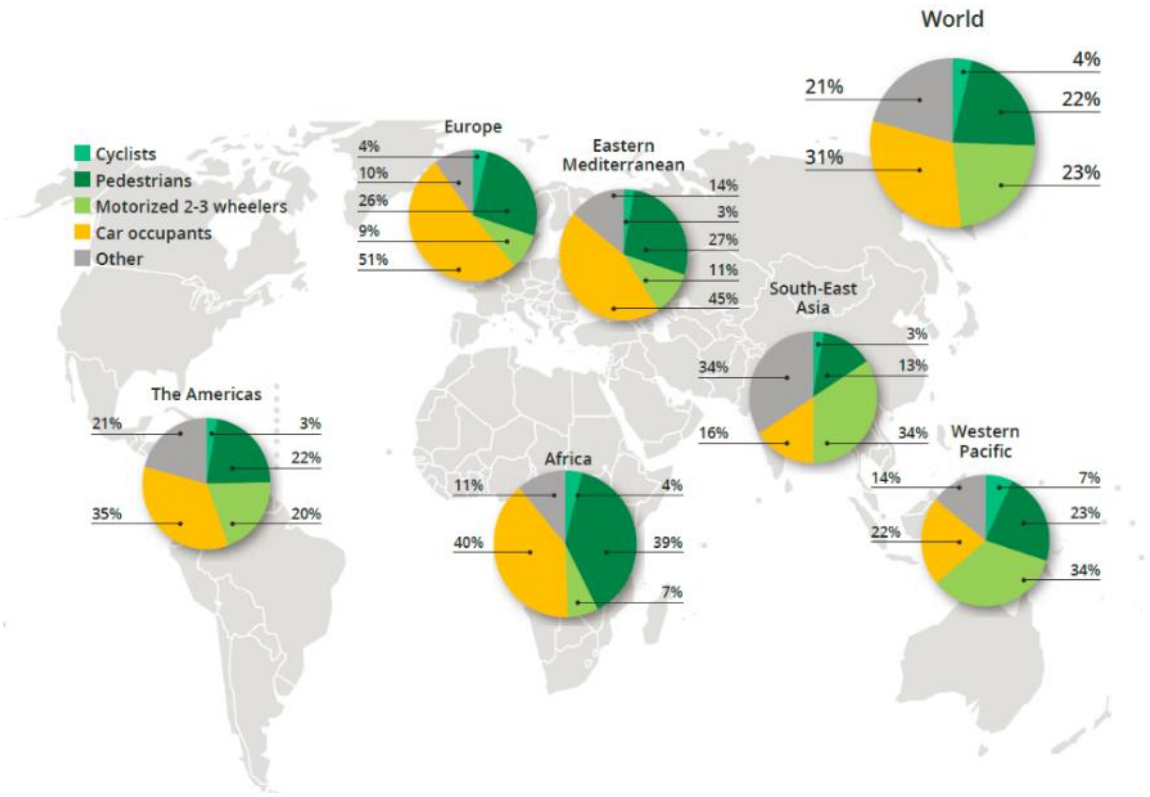


Figure 1-2 Road traffic deaths by type of road user. [1]

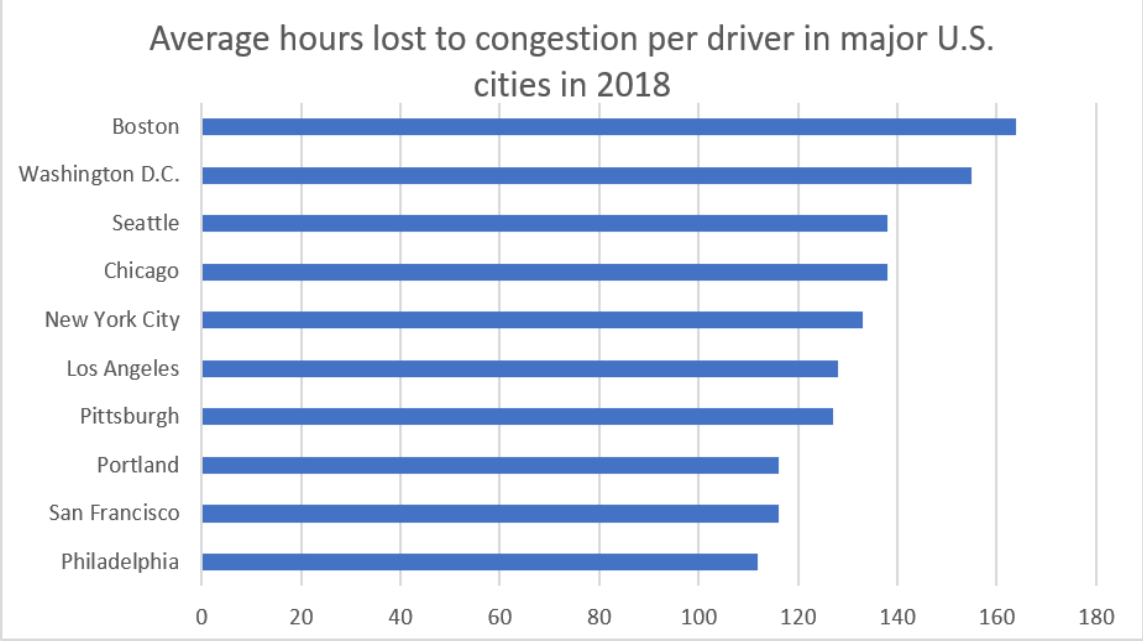


Figure 1-1 Average hours lost to congestion per driver in major U.S. cities. [3]

1.2 Trends and Opportunities

Traditionally, significant efforts have been made by expanding the existing transportation infrastructure or enacting regulations to solve the aforementioned problems. Current trends in sensing techniques, communication techniques, and control techniques give us an unprecedented opportunity to further improve the system into a new epoch. Such integrative research falls into the area of connected and automated vehicles (CAVs). Various sensing techniques, such as the perception based on high fidelity camera, LiDAR, or radar, provide accurate and robust real-time information flow, and according to the information flow, the communication between individual traffic participants and infrastructure expands the horizons even further. Cellular and Wi-Fi are two promising candidates of communication techniques which provide reliable and ubiquitous access services with low latency. The control techniques, including classic control, optimization, and learning, specify the management paradigm to the traffic system. The potential of CAV technologies lie in the introduction of not only the automation but also the cooperation features into the system. By sharing the understanding of the situation in real-time from various traffic participants or infrastructure, the system can be better managed. There are several examples of this: 1) transportation network companies (TNCs), such as Uber and Lyft, bridge the demands (customers) and supplies (mobility service providers) through innovative platforms and smartphone apps to facilitate the completion of mobility needs [5]; 2) ramp management systems reduce congestion and speed fluctuation by coordinating the mainline and on-ramp vehicles [6]; 3) connected eco-operation for transit bus integrates

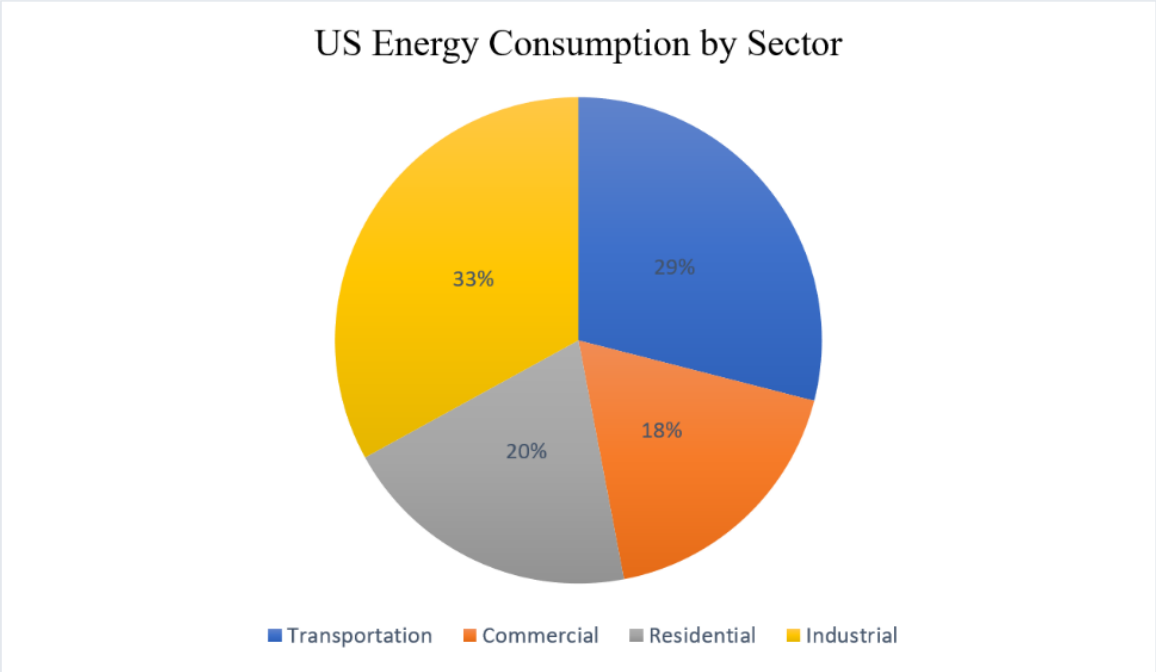


Figure 1-3 US energy consumption by sector. [4]

Eco-Stop, Eco-Cruise, and Eco-Approach and Departure to improve energy efficiency utilizing the information broadcasted from the roadside infrastructure such as look-ahead traffic, terrain conditions and signal phase and timing (SPaT) [7].

1.3 Challenges and Research Gaps

The cooperation problems in this dissertation can be separated into two levels based on their scopes. At the macroscopic level, cooperation considers the interaction between supplies and demands, among supplies, or among demands. While at the microscopic level, cooperative driving automation (CDA) is the research problem to be focused on.

For macroscopic level cooperation, the supply-demand relationship usually can be formed as an optimization problem. To make the mathematical formulation as close to the real-world scenario as possible, more constraints have to be added. As a result, the computation load can be exponentially high as the scale of the problem grows. Therefore, an essential goal of the algorithm design for these problems is to balance the computation

time and the optimality of the solution. The research in this area has been carried out for decades, and the metaheuristic algorithms and heuristic algorithms are most commonly used to reduce the computational load. However, as the process of vehicle electrification and the emergence of innovative Mobility as a Service (MaaS) concept, the formulation of the problem needs to be updated accordingly.

For microscopic level cooperation, the major challenge is that the mixture of automation and human driver will coexist for a long time in the future. In such a mixed traffic condition, the legacy vehicles could be an isolated island, where no advanced information is shared from them. Because the dynamics of the vehicles are controlled by the human driver, the behavior can be highly stochastic. Also, because of the minimum cooperation feature, it is hard to coordinate the legacy vehicles with other CAVs to achieve the desired overall performance. Therefore, it is imperative that we establish a systematic framework to facilitate the coordination between CAVs and non-CAVs. This framework would serve as the bridge, synchronizing the operations of intelligent and legacy vehicles. Simultaneously, the implementation of a robust sensing system becomes essential. Such a system would be responsible for detecting and tracking pertinent vehicle information, forming a comprehensive and continuously updated database that can aid in effective decision making. Finally, a behavior model comes into play. This model is tasked with predicting future behaviors based on historical data and current conditions, laying the foundation for cooperative optimization in forthcoming periods.

1.4 Research Objectives and Contribution of the Dissertation Research

The research objectives and contributions of this dissertation are anchored in the comprehensive examination and enhancement of cooperation in transportation systems in mixed traffic conditions.

Initially, a cooperative framework is established under a multi-modal transportation system. The current state of research within the constituent modules of this framework is reviewed, highlighting areas of development and potential for further exploration. Subsequently, the existing definition of Cooperative Driving Automation (CDA) [8] is broadened to increase its applicability in mixed traffic conditions. The redefined concept emphasizes the importance of harmonizing intelligent and legacy vehicles for effective operation within mixed traffic. At the macroscopic level, two significant applications are addressed to substantiate the utility of the proposed cooperative framework: 1) exploration of shared automated mobility applications, and 2) investigation of dispatching and scheduling algorithms for battery electric truck fleets. It is anticipated that these applications will not only enhance operational efficiency but also yield insights into large-scale coordination among diverse transportation agents. At the microscopic level, a specific application of the framework is illustrated through a corridor-wise ramp management system. This application serves to elucidate the complexities of microscopic cooperation, demonstrating how individual agents can interact and collaborate within the overarching system. To navigate the complexities associated with mixed traffic, the dissertation pursues two vital research directions. Firstly, roadside sensing is examined to bolster detection and monitoring capabilities, ultimately enhancing decision-making and overall system management. Secondly, driver behavior is modeled to gain a deeper understanding to and

predict human driving patterns in mixed traffic scenarios, thereby facilitating improved traffic flow management.

1.5 Organization of the Dissertation

The rest of the dissertation is organized as follows:

Chapter 2 expounds on the categorization and definition of cooperation in transportation systems. It focuses on both macroscopic and microscopic levels, examining the supply-demand balance and cooperative driving automation (CDA). Chapter 3 contains a comprehensive review of the key modules in the framework. It presents different traffic state estimation techniques, explores the role of infrastructure-based sensing, discusses driving behavior modeling, and analyzes various optimization and control methods. Chapter 4 is devoted to macroscopic cooperation. It begins by examining shared automated mobility with demand-side cooperation. The chapter then proceeds to discuss the fleet dispatching strategy for battery-electric trucks. Chapter 5 pertains to microscopic cooperation. Here, a corridor-wise eco-friendly cooperative ramp management system for connected and automated vehicles is introduced and examined in detail. Chapter 6 addresses the mixed traffic problem. Two key areas are investigated: infrastructure-based sensing for better detection and monitoring capabilities, and driver behavior modeling for improved understanding of human interactions in mixed traffic scenarios. Chapter 7 concludes the entire dissertation, summarizing the research outcomes, highlighting key publications resulting from this research, and suggesting directions for future work in the domain of intelligent transportation systems.

2 CATEGORIZATION AND DEFINITION TO COOPERATION IN TRANSPORTATION SYSTEMS

Cooperation is the process of groups of agents working together for common, mutual, or some underlying benefits, as opposed to working in competition for selfish benefits [9]. In general, the mutual benefit can be defined as the single or the combination of the three fundamental problems, i.e., safety, mobility, and environmental sustainability, for different types of traffic scenarios.

As shown in Figure 2-1, the cooperative transportation system is constructed with multiple transportation participants such as road users and infrastructure, where the target cooperation agents vary from case to case. The road users including non-CAVs agents, who do not share detailed information through communication and are not controllable through outside commands, such as human-driven vehicles, pedestrians, and so on; and CAVs which contains at least sensing module, communication module, and vehicle control module. The infrastructure in the transportation system includes surveillance and sensing systems such as loop detectors, monitoring camera; and traffic control systems such as traffic light and ramp meter. The planner module, as the core of the cooperation, can be deployed in individual CAVs and in the infrastructure. If the planner is deployed in individual CAVs, it is a distribution-oriented cooperation where the planner makes decisions in a decentralized way. On the other hand, if the planner is deployed in the infrastructure, in general, a centralized cooperation can be executed.

2.1 Macroscopic Level – Supply vs. Demand

The many-to-many relationship between supply and demand is highly coupled and can be defined as the macroscopic level cooperation in transportation systems. This supply-

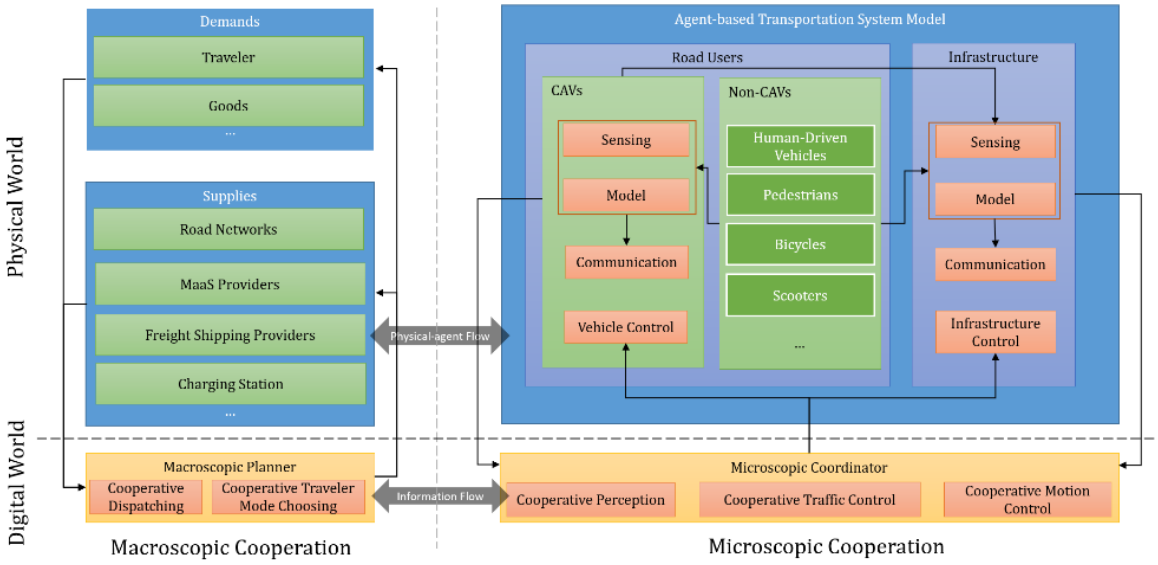


Figure 2-1 Multi-Modal Cooperative for Transportation System.

demand relationship adapts well in the transportation system context in many cases. For example, dispatching and scheduling according to the supply and demand has been explored in the field of operation research for years. Two typical scenarios discussed in this thesis are the truck fleet operating in regional dispatching application and the on-demand shared mobility paradigm for the next-generation urban transportation systems. Among these applications, cooperation plays an important role to improve overall system efficiency and reduce unnecessary costs such as detours or deadheading.

For the truck fleet dispatching application, the problem can be considered as an extension of the classic Vehicle Routing Problem (VRP) [10], which is a well-known combinatorial optimization problem in transportation logistics. The major goal of VRP is to find the optimal or near-optimal itinerary for the truck fleet given the pick-up and delivery order information, which considers the scheduling and routing of multiple vehicles cooperatively. Most of the real-world problems are often more complex than classical VRP. Therefore, studies of VRP usually try to extend the classic VRP by adding further constraints. For instance, the capacitated vehicle routing problem (CVRP) is defined by

adding the carrying capacity limitation of the vehicles [11], and the vehicle routing problem with time windows (VRPTW) is defined by adding the scheduled time of each customer [12]. Also, VRP with pickup and delivery (VRPPD) studies the scenario that customers may request services with pickup and delivery [13]. Up to now, the problem is still the cooperation of supplies only. With the electrification of the vehicle fleet, more and more research turn to the topic of the electric vehicle routing problem (E-VRP) and its extensions [14]–[16]. For E-VRPs, the possibility of recharging at available charging stations is considered. These opportunistic charging, requiring the coordination of scheduling charging time, introduces the cooperation from the demand-side.

For the shared automated mobility application, one example can be the new car-pooling services emerging in major urban areas that offer the most affordable ride price while in exchange for a little walk of customers to/from designated pick-up and drop-off (PUDO) locations with respect to their origins and destinations [17]. Such flexibility in PUDO locations can be considered as another demand-side cooperation strategy. On the supply-side, the service providers, shared automated vehicles (SAVs), decide their assignments and routes together to avoid necessary deadheading and detour. Sometimes, the demands are fixed or disoperative. For example, passengers may feel uncomfortable and refuse to walk. The on-demand shared mobility has been considered as a cost-effective strategy to fulfill transportation demand without compromising traffic congestion, fuel consumption, and air quality.

2.2 Microscopic Level – Cooperative Driving Automation (CDA)

Cooperative Driving Automation (CDA) builds upon the emerging CAV techniques in the transportation system, which looks at the problem from a microscopic

Table 2-1 Relationship between classes of CDA cooperation and levels of automation [8]

		SAE Driving Automation Levels					
		No Automation	Driving Automation System		Automated Driving System (ADS)		
		Level 0 *No Driving Automation (human does all driving)*	**Level 1** *Driver Assistance (longitudinal OR lateral vehicle motion control)*	**Level 2** *Partial Driving Automation (longitudinal AND lateral vehicle motion control)*	**Level 3** *Conditional Driving Automation*	**Level 4** *High Driving Automation*	**Level 5** *Full Driving Automation*
CDA Cooperation Classes	**No cooperative automation**	(e.g., Signage, TCD)	Relies on driver to complete the DDT and to supervise feature performance in real-time		Relies on ADS to perform complete DDT under defined conditions (fallback condition performance varies between levels)		
	Class A: Status-sharing *Here I am and what I see*	(e.g., Brake Lights, Traffic Signal)	Limited cooperation: Human is driving and must supervise CDA features (and may intervene at any time), and sensing capabilities may be limited compared to C-ADS		C-ADS has full authority to decide actions Improved C-ADS situational awareness beyond on-board sensing capabilities and increased awareness of C-ADS state by surrounding road users and road operators		
	Class B: Intent-sharing *This is what I plan to do*	(e.g., Turn Signal, Merge)	Limited cooperation (only longitudinal OR lateral intent that may be overridden by human)	Limited cooperation (both longitudinal AND lateral intent that may be overridden by human)	C-ADS has full authority to decide actions Improved C-ADS situational awareness through increased prediction reliability, and increased awareness of C-ADS plans by surrounding road users and road operators		
	Class C: Agreement-seeking *Let's do this together*	(e.g., Hand Signals, Merge)	N/A	N/A	C-ADS has full authority to decide actions Improved ability of C-ADS and transportation system to attain mutual goals by accepting or suggesting actions in coordination with surrounding road users and road operators		
	Class D: Prescriptive *I will do as directed*	(e.g., Hand Signals, Lane Assignment by Officials)	N/A	N/A	C-ADS has full authority to decide actions, except for very specific circumstances in which it is designed to accept and adhere to a prescriptive communication		

perspective. In this context, the detailed dynamics and decision making of the transportation participants such as vehicles and pedestrians are studied. Through vehicle-to-everything (V2X) communication, two or more participating entities (including communication devices) can cooperate with each other by supporting the movement of multiple vehicles in proximity to improve safety, mobility, and environmental sustainability. Such cooperation, constructed based on communications and predefined regulations, originates from the very beginning of the transportation system. For example, hand signals or turn signals can inform other road users of the intention of the current vehicle such that the surrounding road users can respond accordingly, e.g., slow down and compromise. Another example, from the infrastructure side, is the actuated traffic signals system with loop detectors who can change the light based on the detected real-time traffic condition, and the vehicles are therefore controlled based on the signal phases.

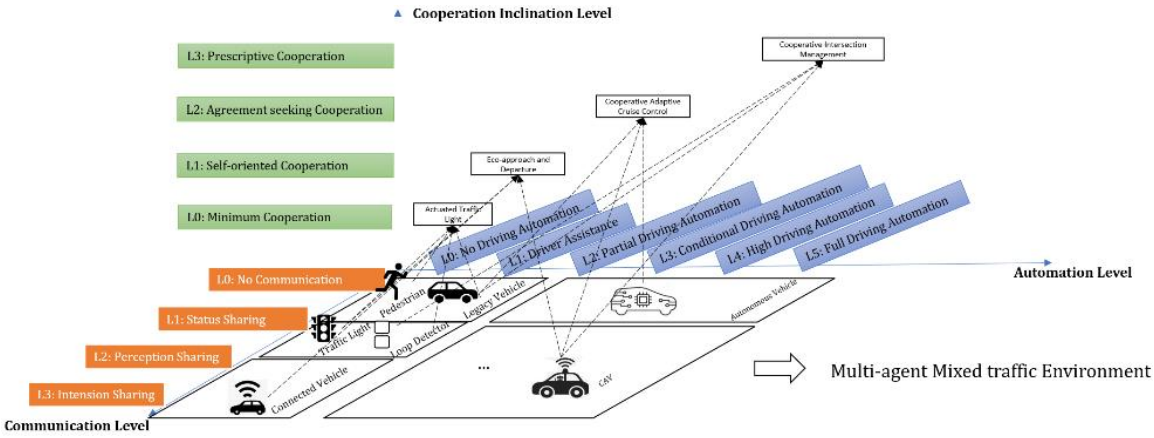


Figure 2-2 3-Dimensional definition of cooperative driving automation.

According to the SAE surface vehicle information report J3126 [8], the CDA cooperation defined as Classes A through D based on the increasing amount of cooperation entailed in each successive class (see Table 2-1). Class A is the status-sharing cooperation, where perception information about the traffic environment and information about the sending entity provided by the sending entity for potential utilization by receiving entities; Class B is intent-sharing cooperation, where information about planned future actions of the sending entity provided by that entity for potential utilization by receiving entities; Class C is agreement-seeking cooperation, where a sequence of collaborative messages among specific CDA devices intended to influence local planning of specific dynamic driving task (DDT)-related actions; Class D is prescriptive cooperation, where the direction of specific action(s) to specific traffic participants for imminent performance of the DDT or performance of a particular task by a road operator, provided by a prescribing CDA device agent(s) and adhered to by a receiving CDA device agent(s).

To further expatiate the features of an agent in the microscopic-level cooperative transportation system, a 3-dimentional coordinate system is defined, where the 3 axes are communication level, automation level, and cooperation inclination level, respectively.

It is not difficult to find that the information shared from the traditional traffic system is either the direct intentions (from vehicles or other road users) or simple control instructions (from infrastructure). More sophisticated information such as operation status and perception of the surrounding environment can boost the system efficiency to a new level. For instance, the Cooperative Adaptive Cruise Control (CACC) system allows vehicles to form platoons and drive at harmonized speeds with constant time headways between vehicles [18]. By sharing the speed or acceleration of the vehicles, the operation of the group can be optimized. In this way, without compromising safety, the roadway capacity can be significantly increased as the reduction of the intervehicle gaps. Also, energy consumption and pollutant emission can be reduced because of the reduction of unnecessary speed fluctuation and additional aerodynamic drag if driving individually. The benefit of sharing the perception of the surrounding environment is that it can enhance the situational awareness of all the traffic participants thus enhancing the understanding of the overall traffic system. The largely applied application nowadays is the eco-routing system where road uses can be assigned to different roads to relieve traffic congestion [19]. Another feature that improves safety by sharing perception information is the cooperative object tracking, which can eliminate the risk of the blind spots of human drivers [20]. The degree of communication considered in this thesis is in four levels: Level 0: no communication, where only direct intentions from vehicles or other road users (e.g. turning light, braking light) and simple control instructions from infrastructures (e.g. traffic light) are shared; Level 1: status sharing (who am I and what am I doing), where basic safety message (BSM) such as vehicle size, position, speed, and acceleration are shared; Level 2: perception sharing (what I see), where sensed information of the surrounding environment

is shared; and Level 3: intention sharing (what I plan to do), where planned actions such as routing, lane change indicator, and desired trajectory are shared.

Automation is another key feature of CDA applications. Human drivers can never concentrate continuously for a long time while driving as accurately as the automatic control system in terms of speed tracking, lane keeping, and so on. Borrowing from the well-known definition of the SAE J3016 [21], the levels of driving automation are defined as follows: Level 0: no driving automation; Level 1: driver assistance (longitudinal or lateral vehicle motion control); Level 2: partial driving automation (longitudinal and lateral vehicle motion control); Level 3: conditional driving automation; Level 4: high driving automation; Level 5: full driving automation. It can be observed from the traditional vehicle cooperation cases that automation is not the necessary condition. However, to control the dynamics of vehicle elaborately, automation is the fundamental module of the system. The requirement of the level of automation depends on the specific applications.

Other than the levels of communication and automation, another important dimension to think about CDA is the level of the cooperation inclination. First, the legacy vehicles are not purely disoperative but have the minimum cooperation inclination to comply with the traffic regulations or conventions. For example, human drivers would usually slow down when seeing turning light on preceding vehicles from adjacent lanes. Second, for the autonomous vehicles or CAVs with only decentralized decision-making logic, they would consider all the information received but plan their actions solely. Third, the CAVs, who are willing to make decision together with others or carry out instruction from a centralized planner, are considered as fully cooperative agents. Combining the definition from SAE [8], here we define the cooperation into 3 different levels: Level 0:

minimum cooperation (basic cooperation only to comply with the regulations or conventions); Level 1: self-centered cooperation (acquire and analyze the shared information from others while making decisions individually); Level 2: full cooperation (accept centralized management from the outside controller).

To conclude, a 3-dimensional definition of the microscopic level cooperation, CDA, is given in this subsection (see Figure 2-5.), which expatiates how cooperation formed by the traffic participants, including both the road users and infrastructure.

3 REVIEW OF VARIOUS FRAMEWORKS

The key modules in the CDA system include traffic state estimation, driving behavior modeling, and optimization and control [6]. As the human-driven vehicle only share the basic intension information, the surveillance of these vehicles is required to better understand the real-time traffic conditions. In general, surveillance can be achieved by surrounding CAVs equipped with sensing systems and roadside infrastructures like surveillance cameras, loop detectors, and so on. Then, the traffic state estimation module will further analyze the data combining from both the surveillance and the CAVs. The goal of traffic state estimation is to provide an accurate prediction of the current traffic information both microscopically and macroscopically based on apriori knowledge and partial observation. Surveillance data for individual human-driven vehicles is also valuable for better understanding and predicting the behavior of that vehicle. Therefore, the data will also be fed into the driving behavior model, which will conclude the driver behaviors into a mathematical form such as a utility function or a reward function. This kind of function can depict how the specific human driving interacts with the environment. Then, the optimization and control module will decide detailed instruction for the controllable agents like CAVs. The instruction can be the desired trajectories, powertrain control signals, and so on.

3.1 Traffic State Estimation

Accurate measurement and estimation of prevailing traffic conditions are the foundation of effective cooperation driving automation. Since it is difficult and expensive to obtain complete information on the traffic (e.g., 100% penetration rate of connected vehicles), estimation of traffic states, such as flow, density, and speed, from partially

observed traffic data plays an important role. Seo et al. performed a comprehensive survey about traffic state estimation which provides a guideline into this field [22]. The categories listed below are on the basis of their suggestion.

At the macroscopic scale, road networks are divided into several segments without further merging or diverging, which are called links. Inside each link, the traffic states can be considered to be homogeneous. Usually, flow (veh/hr), density (veh/km), and speed (km/hr) are three key variables used for traffic flow management.

Raw data available for estimation can be categorized as stationary and mobile. The stationary data is collected by fixed location sensors such as inductive loop detectors, micro-wave radars on the roadside, and surveillance cameras. The mobile data is collected by moving vehicles equipped with sensors such as GPS, onboard radar, and camera. The emergence of CAVs on roads can provide a significant amount of mobile sensor data.

Traffic flow model is widely used for traffic state estimation. The models are usually based on physical and empirical relations. Borrowing the idea of the hydrodynamic theory, the fundamental diagram depicts the relationship between density and speed, and/or density and flow [23], [24]. Another dynamic representation model with the hydrodynamic theory is cell transmission model (CTM) [25], which utilizes the discrete analog of the differential equations arising from a special case of the hydrodynamic model of traffic flow. Because of the conservation law for the traffic hydrodynamic, the aggregated behavior of traffic is depicted by partial differential equations. Lighthill-Whitham-Richard (LWR) model is the most commonly used first-order model [26], [27], and there are many higher-order models such as Payne-Whitham (PW) model [28], [29].

3.1.1 Model-Based Approaches

Model-based approaches are widely used in the field of traffic state estimation, where the aforementioned traffic flow models are applied. The parameters of the models are usually calibrated via historical data from the field. After the calibration, target data is fed into the model to estimate the traffic states. Coifman proposed a method of estimating microscopic vehicle information (i.e., travel times and trajectories) using LWR model and loop detector data [30]. Wang and Papageorgiou estimated macroscopic traffic via Extended Kalman Filter (EKF) [31]. However, these approaches may fail if inappropriate models are adopted. Furthermore, as the parameters are calibrated by specific data sets, the model-based approaches may not be so adaptive to the drastic changes in traffic conditions.

3.1.2 Learning-Based Approaches

Different from model-based approaches, no empirical traffic flow models are used in learning-based approaches. Learning-based models are trained through statistical methodologies or machine learning techniques given a large amount of historical data. A typical learning-based estimation approach was proposed by Tak et al. [32], in which k-nearest neighbors (KNN) algorithm was applied. Because of the challenge to well interpret the features of the learning scheme, it is difficult to analytically comprehend how the entire system works even though the results may be very attractive. Additionally, similar to the model-based approaches, the dependency of the trained data set makes the system not flexible enough to transfer to other untrained scenarios.

3.1.3 Streaming-Data-Driven Approaches

Without using neither empirical models nor historical data, streaming-date-driven approaches only rely on real-time data and "weak" assumptions. Therefore, it is more robust to different traffic conditions. The "weak" assumption such as Conservation Law is generally reasonable, considering the physical constraints. Recent research was performed by Florin et al. [33], who presented a mobile observer method with aggregated information on the number of overtaking maneuvers of vehicles. The increasing number of CAVs and connected vehicles on freeways can provide a large amount of streaming data for traffic state estimation, which makes it possible to consider mixed traffic scenarios. Bekiaris-Liberis el al. proposed a mixed traffic state estimation method with a streaming-data-driven approach utilizing only average speed measurements reported by connected vehicles and a minimum number (sufficient to guarantee observability) of spot-sensor-based total flow measurements [34].

3.2 Infrastructure-Side Sensing

As a crucial module in the cooperation system under mix-traffic conditions, infrastructure-side sensing plays a pivotal role. For those agents designated as non-CAV, such as human-driven vehicles and pedestrians, infrastructure-side sensing can conveniently supply their location information. This integration allows for a more seamless interaction between different traffic participants, serving to augment overall system efficiency and safety. This cooperative system is built on the robust data collected and distributed by the infrastructure-side sensing technology, fundamentally enhancing the situational awareness and decision-making capabilities of all participants in the mix-traffic environment.

3.2.1 Vision-Based Perception

Datondji et al. [35] give a comprehensive survey of the traffic monitoring of road intersections, where vision-based vehicle detection plays the most important role in the system. In general, vision-based detection can be categorized into five classes, including background subtraction [36], model-based detection [37]–[40], feature-based detection [41]–[45], motion-based detection [46], [47], and Artificial Neural Network (ANN)-based detection [48]–[55].

The approaches requiring historical data are model-based detection and ANN-based detection. Model-based object detection identifies possible foregrounds in images by fitting vehicles' 2D or 3D models to image regions. However, the methods require a large database of vehicle models from different perspectives and lighting conditions. Therefore, some studies introduced probabilistic frameworks or motion information to reduce the search space and improve the system robustness [39], [40]. With the fast development of deep learning techniques, ANN-based approaches have dominated object detection applications in recent years. In the specific dataset, the most recent network, such as Fast R-CNN [50], [54], [55], SSD [52], and YOLO [51], [53], can even outperform humans. However, it requires a large amount of labeled data for training to have good prediction accuracy. Although some networks are built for small object detection based on onboard fisheye cameras, there is no complete solution suitable for our scenarios: object perception (detection, localization, and tracking) with roadside fisheye cameras where moving objects range in size from small to large with distortion.

Considering the lack of roadside perception datasets, data-free approaches such as feature-based detection, motion-based detection, and background subtraction are more

suitable for the early-stage development of the roadside perception system. Feature-based detection methods combine, graft, and track the geometry or appearance features, such as SIFT [56], SURF [57], and Haar-like Wavelets [58], to select the vehicle candidate in the image. These human-selected features are intended to capture the unique characteristics of the objects while also being resistant to minor alterations in diverse environments. One of the significant challenges for feature-based detection is occlusion, where the feature of the occluded object would be lost. Motion-based detection approaches utilize optical-flow techniques [46] to extract the movement of the candidate objects from the background image. The traffic surveillance application usually has a good performance as the background is relatively consistent. However, these approaches are usually computationally heavy and not robust for slow-moving or stationary object detection.

Finally, one of the most common approaches for extracting foreground objects from target photos is background subtraction. Because a static background is susceptible to changes in the background environment, such as light, shadow, and weather, employing a predefined background image typically results in poor system performance and stability. As a result, the most important module is to generate real-time background images in response to changes in the surroundings, which can be defined as dynamic, statistical, or adaptive background estimation. The representative dynamic background estimation algorithms are Gaussian Mixture Model (GMM)-based method [59], Fuzzy models [60], [61], Local Binary Similarity segmenTER (LOBSTER) [62], and Visual Background extractor (ViBe) [63]. However, the existing methods still face numerous challenges. For example, the GMM-based method necessitates a proper initiation of the background model, which can be difficult to achieve in practice. Some methods, such as LOBSTER, are

computationally heavy, thus not capable of real-time perception. Furthermore, all the existing adaptive methods are sensitive to objects that are moving slowly or stationary. After numerous iterations, the vehicles idle at the intersection waiting for the red light may be erroneously classified as part of the background.

The localization process is necessary to get the location of the detected objects in the world coordinates. Localization from the onboard vehicle perspective has been studied for decades, which usually requires a high-definition map or additional sensing sources such as GPS and IMU [64]. As to the roadside perception use cases, because the cameras are fixed at certain locations, the localization process can be rather simplified. For 3D sensors like LiDAR or multi-camera, the location of the detected object can be calculated based on the extrinsic calibration parameters [65]. For 2D sensors like a single camera, it requires either a projection from 2D to 3D space or depth estimation [66].

Vision-based Multi-Object Tracking (MOT) algorithms process image sequences to create object correspondences over the images and assign consistent identification numbers to the detected objects [67]. Most of the methods utilize Kalman Filter (KF) to predict the object's motion from one frame to another. The most well-known MOT algorithm is SORT [68], which associates objects using bounding box detection. Then DeepSort [69] improved the bounding box association step by introducing Convolutional Neural Network (CNN). Instead of bounding box tracking, Zhou et al. proposed CenterTrack [70] to track objects as points, which is not only accurate but also fast.

3.2.2 Sensor Selection and Challenge

LiDAR is an active sensor that generates the range and reflection point cloud, whereas a camera is a passive sensor that projects the 3D space into 2D planes with three color channels. As a result, it is easier for the camera to recognize the objects but harder to obtain precise range information. Because the roadside sensors often shoot to the ground and all the objects (e.g., cars, pedestrians) are assumed to move on the ground surface, the ranging impacts may be minimal. In comparison to LiDAR, however, the camera normally has significantly greater resolution. Therefore, in this study, we choose the camera as the sensor of our RSPU. Compared to the regular perspective camera, the fisheye camera or omnidirectional cameras are becoming increasingly used in multiple areas such as traffic monitoring [71] and drone sensing [72] due to its wider detection view. Taking advantage of this feature, fewer cameras are required to cover the entire range of the target intersection. However, the images from the fisheye camera suffer from distortion and perspective effects, which necessitate additional processing. Furthermore, unlike the development of onboard detection algorithms, which already has a large amount of labeled data, the roadside camera dataset with positioning information, let alone fisheye camera dataset, are scarce. As a result, instead of deep learning-based approach, we have to rely on the image processing and conventional computer vision techniques to achieve the data-free real-time detection, localization and tracking tasks.

3.2.3 Fisheye Camera Applications

Due to the large angle of view, fisheye lens camera is becoming more and more popular in automated driving and roadside monitoring applications [73]. As the incident light cannot be projected onto a limited plane, fisheye lens modeling usually relies on

equisolidity projection, orthogonal projection, or stereographic projection [74], which makes the calibration of the fisheye camera more difficult. The calibration method for the fisheye intersection detection and tracking is discussed in the Methodology section. Due to the lack of 3D datasets of fisheye images, Plaut et al. offered a 3D object detection model that is trained only on the rectilinear images without using any fisheye training data [75]. Although the majority of the current image dataset is from a regular camera, some new dataset from a fisheye camera has emerged as a result of the increased interest in this. For example, Rashed et al. create an open-source dataset consisting of 10,000 fisheye images along with all the object representations of ground truth for autonomous driving [76]. The fisheye camera-based detection and tracking technique appears to be promising. Using the labeled fisheye data, Yahiaoui et al. proposed a FisheyeMODNet to detect moving objects on surround-view cameras for autonomous driving, which can run at 15 frames per second on an automotive embedded system at an accuracy of 40% [77]. To enhance the perception performance by adding a different data source, Zhu et al. introduced attitude data and fused them with raw fisheye images [72]. From the roadside perspective, Wang et al. presented a feature-based real-time detection approach to track and count vehicles using simple feature points tracking grouping and association [71].

3.3 Driving Behavior Modeling

Modeling and predicting the driving behavior of conventional human-driven vehicles are essential for designing the motion behavior of CAVs in mixed traffic conditions. In this section, I will first introduce car-following models, which are widely used in CDA applications such as CACC and ramp management system. Then, a more

detailed algorithm, inverse reinforcement learning, is reviewed as it potential of modeling human-driven behavior.

3.3.1 Car-Following Model

The task of P-ACC design can be regarded as both the driving behavior modeling and the personalized car-following controller adaptation. Therefore, in this section, we briefly review existing methods for driving behavior modeling and taxonomies of car-following models. Some of the methods tackle these two tasks separately, while others couple the two tasks and provide a control scheme in an end-to-end manner.

One of the most prevalent methods for vehicle longitudinal control is to design control policy based on physics laws, where the most common ways are by modeling the car following with an Ordinary Differential Equation (ODE) (i.e., $\dot{v} = f(s, v, u)$). The ODE equation explicitly depicts the interaction between the driver and his/her front vehicle considering the dynamics of the ego vehicle, which can be categorized as an end-to-end method. The car-following behavior is determined by the acceleration of the ego vehicle, which is derived based on the speed of the ego vehicle and the preceding vehicle and their distance gap using the ODE equation. The most commonly used models under this category include Gipps model [78], Intelligent Driver Model (IDM) [79], as well as Newell's car-following model [80]. Other than mapping the control input directly using the ODE, another type of physics-based controllers first model the car-following dynamics in the state space, and then use the Model Predictive Controller (MPC) to optimize the predefined objective functions in terms of the factors like comfort, safety, and fuel economy requirements [81], [82]. The major drawback of the physics-based approaches is that it is relatively difficult for them to capture personalized behaviors.

On the other hand, the data-driven methods model the car-following behaviors from the demonstrated trajectories directly. As a result, they are naturally suitable for modeling personalized driving styles. Most of the data-driven methods follow the end-to-end scheme, aiming to find the mapping function from current states (or previous sequential states) to control inputs. In general, data-driven methods can be categorized into two types. The first one is ***Imitation Learning***, where models clone behaviors of the demonstrations by learning the mapping from states to actions. Wang et al. used Gaussian Process Regression (GPR) to learn drivers' behaviors from naturalistic data, as it only requires a small amount of data and uses Bayesian treatment to avoid overfitting. However, the computational complexity of GPR is $O(n^3)$ for training and $O(n^3)$ for inferencing, which severely limits the scalability of the method [83]. Wang et al. and Pongtep et al. applied the Gaussian Mixture Model (GMM) for stochastic driver behavior modeling [84]. However, GMM treats the data as a set of scattered points and does not consider the dynamic of the system nor the transition preference of the driver. With the rapid development of deep learning, Deep Neural Network (DNN)-based approach has been used widely to model car-following behaviors such as [85], [86]. Furthermore, because driver's decision-making may be a sequential process, memory networks, such as Recurrent Neural Network (RNN) [87], Long Short-Term Memory (LSTM) [88], and Attention-Based Network [89], have also been successfully employed for car-following model. However, as the demonstration trajectories cannot traverse the entire state space, extrapolation is required at the inference stage. Therefore, the performance of the systems cannot be guaranteed. Applying a safety filter after the network output is necessary to ensure safety as described in [90].

The second type of data-driven approach is ***Apprenticeship Learning***, which decouples driving behavior modeling and car-following controller adaptation. The models first reason driver's behaviors by approximate rewards/objectives of demonstration trajectories using IRL [91]–[95] or Inverse Optimal Control (IOC) [96]. Then, MPC, RL, or other controllers are applied based on the recovered rewards/objectives. Rather than directly reproducing the motion, apprenticeship learning method apprehends the preference of the target agent before making decisions. As a result, it is expected to have better performances for unseen scenarios. In addition, because the controller design is decoupled with behavior modeling, both stability and safety can be guaranteed mathematically based on the choice of the controller. Some researchers also tried to use RL (or deep RL) directly to train a human-like car-following policy [97], [98]. As handcrafted reward functions consider different factors, they usually have good performance in specific environments. However, they fail to depict drivers' personalized behaviors.

3.4 Optimization and Control

The optimization and control module is the core of the cooperation applications as it determines the behaviors of the controlled CAVs and will highly influence the system performance. In this section, we strategically categorize the optimization and control into two types: centralized approaches and distributed approaches.

3.4.1 Centralized Approaches

We define a CAV coordination approach as centralized if the tasks and control commands of the system are globally conducted by the roadside infrastructure and/or the transportation management center (TMC) for all CAVs. Some of the major reasons that

such coordination is conducted in a centralized manner are that, each CAV in the system might not have global information of the system, nor can they conduct computations locally that consume large computational power and long computational time.

In some centralized approaches, tasks and control commands were executed by different layers of the centralized controller. A two-layer CAV coordination system at ramp was proposed by Schimidt et al. in 1983, which was based on non-linear system dynamics behavior [99]. In their system, the higher layer is in charge of the sequence control, while the lower layer is in charge of the motion control of vehicles. Ran et al. proposed a similar centralized multi-layer automated ramp system and also built a microscopic simulation model to validate its characteristics, as well as the designing requirements of the minimum ramp length [100].

Other than the aforementioned approaches, coordination of CAVs at ramp can also be modeled as an optimization problem to be solved by the centralized controller, with the aim of minimizing travel time or fuel consumption. Awal et al. proposed an optimization problem with the objective of reducing merging time at ramps and thus reducing merging bottlenecks [101]. Raravi et al. formulated an optimization problem that aims at minimizing the Driving-Time-To-Intersection (DTTI) of vehicles, subject to certain constraints for ensuring safety [58]. Rios-Torres et al. presented an optimization framework and an analytical closed-form solution that allowed the online coordination of CAVs at on-ramp merging zones [102]. Xie et al. formulated this case as a nonlinear optimization problem and conducted a simulation evaluation with MATLAB and Car2X module in VISSIM [103].

3.4.2 Distributed Approaches

Different from centralized approaches that rely on the roadside infrastructure and/or the TMC with infrastructure-to-vehicle (I2V) communication, distributed approaches of CAV coordination at ramp control conduct coordination decisions locally among different CAVs through vehicle-to-vehicle (V2V) communication. Compared to the centralized approaches, the distributed approaches bring several benefits, including reducing communication requirements and improving scalability.

The concept of virtual vehicle coordination at ramps was originated from Uno et al. [104]. The proposed approach maps a virtual vehicle onto the highway mainline before the actual merging happens, allowing vehicles to perform safer and smoother merging maneuvers. Lu et al. applied a similar idea in their proposed systems, where they first formulated the merging problems differently with respect to two different geometric layouts of the road (i.e., either with or without a parallel lane), and then proposed a speed based closed-loop adaptive control method to control the longitudinal speed of merging CAVs [105].

Besides the virtual vehicle ideas, other distributed approaches were also proposed to control the longitudinal motion of CAVs at ramps. Dao et al. proposed a distributed control protocol to assign vehicles into vehicle strings in the merging scenario [106]. Zhou et al. developed a vehicle trajectory planning method for CAV coordination at ramp, formulating the planning tasks of the ramp vehicle and the mainline vehicle as two related distributed optimal problems [101]. Wang et al. proposed a distributed consensus-based CAV coordination system [107]. Furthermore, agent-based modeling and simulation of the proposed CAV coordination system were conducted in game engine Unity3D, and it was

compared to human-in-the-loop simulation to evaluate its benefits in terms of mobility and sustainability [108], [109].

4 MACROSCOPIC COOPERATION

4.1 Shared Automated Mobility With Demand-Side Cooperation

4.1.1 Introduction and Background

Shared and automated mobility has been prevailing and changing the paradigm of next-generation urban transportation systems, leading to disruptive concepts such as Mobility-as-a-Service (MaaS) and transportation network companies (TNCs), such as Uber and Lyft. TNCs have been efficiently identifying the missing links between demands (customers) and supplies (mobility service providers) and bridging them through innovative platforms and smartphone apps to facilitate the completion of mobility needs. In spite of never-ending criticisms to TNCs such as avoiding government regulations and inducing excess traffic demands [110], [111], they keep evolving by providing feasible solutions, such as ride-hailing, pooled TNCs, and different tiers of transportation needs [112], [113].

Recently, new car-pooling services emerging in major U.S. cities [5] offer the most affordable ride price in exchange for a little walk of customers to/from designated pickup and drop-off (PUDO) locations with respect to their origins and destinations. Such flexibility in PUDO locations can be considered as a demand-side cooperative strategy. It is similar to the travelling salesman problems (TSP) with moving targets, which have been explored in the field of operation research for years [114], [115]. Such flexibility in travel behaviors of customers may impact overall system efficiency and sustainability [116], such as vehicle miles traveled (VMT), emissions, and energy consumption, although the emerging on-demand mobility services rely on many other types of studies, such as studies

of existing services, stated preference studies, and policy studies [117]. However, whether the anticipation holds and how much SAM (shared automated mobility) service will be impacted due to the demand-side cooperation is still unknown. To address all the challenges mentioned above, or in other words to assess the mobility and sustainability impacts of SAM services with demand-side cooperation, this dissertation proposes a demand-side cooperative (DC) SAM service optimization model and an open-sourced microscopic simulation platform. The DC ride matching is formulated as a capacitated vehicle routing problem with repositioning (CVRPR) and is solved by a commercial solver (Gurobi). Under the operational constraints, such as SAV seat capacities and maximum walking distances, the DC ride matching strategies aims to optimize the overall profit of the proposed SAM service, which considers both maximizing the serving rate (to obtain more revenue) and minimizing the travel distance, travel time, and energy consumption (to reduce the fleet operational cost). The proposed service can potentially benefit the customers and the entire transportation system by reducing the detoured portions and dead-heading time of SAV trips. The proposed DC-SAM is framed in the Simulation of Urban MObility (SUMO), an open-source and multi-modal microscopic traffic simulation tool. It is capable of modeling not only vehicular traffic dynamics in detail but also customer behaviors (including customer–vehicle interactions) via its unique application programming interfaces (APIs), i.e., "TraCI" [118]. This enables the proof-of-concept study of the proposed DC-SAM service in a dynamic environment where the ride matching and repositioning (i.e., re-optimization) are performed continuously as the system evolves (e.g., new on-demand ride requests pop up). In addition, a real-world network of New York

City (NYC) is coded, and ride demands as well as background traffic are synthesized to evaluate the performance of the proposed DC-SAM service.

On-demand shared mobility has been considered as a cost-effective strategy to fulfill transportation demand without compromising traffic congestion, fuel consumption and air quality [119]. In particular, ridesharing refers to the rides in a vehicle among individual travelers (a driver or customer) whose itinerary is in the proximity of both space and time, although the system in which customers may not share the vehicle at the same time can also increase congestion. With the emergence of smartphones and the Internet, for-hiring pooled services research and development has focused on online ride-matching programs as well as real-time traveler information delivery. Thanks to both the rapid advances in information and communication technologies and increased concerns for contemporary transportation issues (e.g., congestion, environment, and parking), more affordable, secure, and accessible Mobility on Demand (MOD) [120], shared ride [121], and pooled TNC services have been provided continuously by transportation network companies (TNCs) via smartphone apps, such as Uber and Lyft [122]. Based on positional elements, Furuhata et al. proposed a systematic classification scheme over the ridesharing patterns and discussed some significant challenges and future directions, mainly from the perspective of matching agencies [17].

From a mathematical perspective, the dynamic ridesharing (DRS) problem can be categorized into the well-known vehicle routing problem (VRP), or more specifically dynamic VRP [123]–[125]. Due to the computational complexity of VRP, a myriad of studies has been focused on developing efficient heuristic approaches to solve DRS problems under different scenarios [126], [127]. Furthermore, with the introduction of

mobile apps and improved services from TNCs, variants of DRS problems have emerged. Wang considered a DRS problem where drivers or riders may accept or reject the ride-matching assignment provided by the system [128]. Simonetto et al. proposed a computationally efficient dynamic ridesharing algorithm based on a linear assignment problem and federated optimization architecture [129]. In a follow-up study, they examined the impacts of cooperation and competition between ridesharing companies through the Mobility-as-a-Service (MaaS) platform, and showed that the competition could worsen the on-demand mobility service, especially in the presence of customer preferences [130]. To improve system efficiency, Coltin and Veloso proposed a heuristic algorithm to coordinate ridesharing routes and matching, which may smoothly transfer customers between different vehicles [131].

Most of the aforementioned DRS studies, however, assumed door-to-door services. Only a few consider more flexible pickup and drop-off (PUDO) locations, which may potentially provide system-wide benefits for the ridesharing service due to the demand agglomeration effects [132]. Li et al. developed an enhanced ridesharing system where the users may be collectively picked up or dropped off, and the preliminary numerical study showed that the proposed system could improve the overall travel time [133]. Zhao et al. also relaxed the PUDO location constraints in the ridesharing problem and performed a case study in MATLAB [134]. Although the results from these studies were promising, their validation was limited to numerical analyses only without considering the dynamic nature of the system. Therefore, modeling and evaluation of the proposed DRS system in a microscopic traffic simulation environment would be very valuable, which is a focus of this dissertation.

4.1.2 System Framework

In transportation research, the performance of emerging mobility technologies and services has been evaluated by transportation demand models and traffic simulation tools. Macroscopic or mesoscopic simulation models, focusing on the network or link level traffic dynamics, may not provide detailed behavior on an individual vehicle or customer basis. Agent-based models such as MATSim are able to describe the activities of every agent in a large scale, but not the delicate interactions between them (vehicles and customers) or the traffic dynamics in a realistic manner. Most of the microscopic simulation tools primarily use an old-fashioned vehicle-based paradigm where customers' behavior in a SAM service cannot be well modeled. Some commercial software, such as PTV VISSIM, has attempted to extend the capability of its products with MaaS features [135]. However, it is very challenging to integrate enough flexibility for demand-side behaviors such as movements of pedestrians or customers. To the best of our knowledge, a demand-side cooperative shared automated mobility service has never been modeled and evaluated in a simulation platform with a realistic roadway network and sophisticated operational settings. The proposed demand-side cooperative shared automated mobility (DC-SAM) service includes the framework in microscopic traffic simulation and dynamic ride-matching algorithm.

4.1.2.1 System Framework in Simulation

The proposed system [136], built upon a microscopic simulation architecture of SUMO with background traffic, consists of a group of customers, a fleet of shared automated vehicles (SAVs), and a service coordinator. SUMO has been used for describing emerging on-demand shared mobility in several studies [137], [138]. The customer's

demand is generated randomly over the simulation network. The request information is sent to the service coordinator, including time stamp, location (with privacy consideration), group size, trip origin (if different from the location upon request), and trip destination. As the core of the DC-SAM system, the service coordinator keeps collecting the riding requests from customers and monitoring the states of SAVs (e.g., location, seat availability) as well as network traffic in real time. Then, it determines the optimal ride-matching for each customer–SAV pair and the alternative pickup and drop-off (PUDO) locations and communicates all this information with designated customers. Once the customers confirm the matched SAVs and PUDO locations (which may be mandated by customers or suggested by the system and may be different from their trip origins and destinations), the service coordinator will deliver walking guidance related to PUDO locations (if applicable)

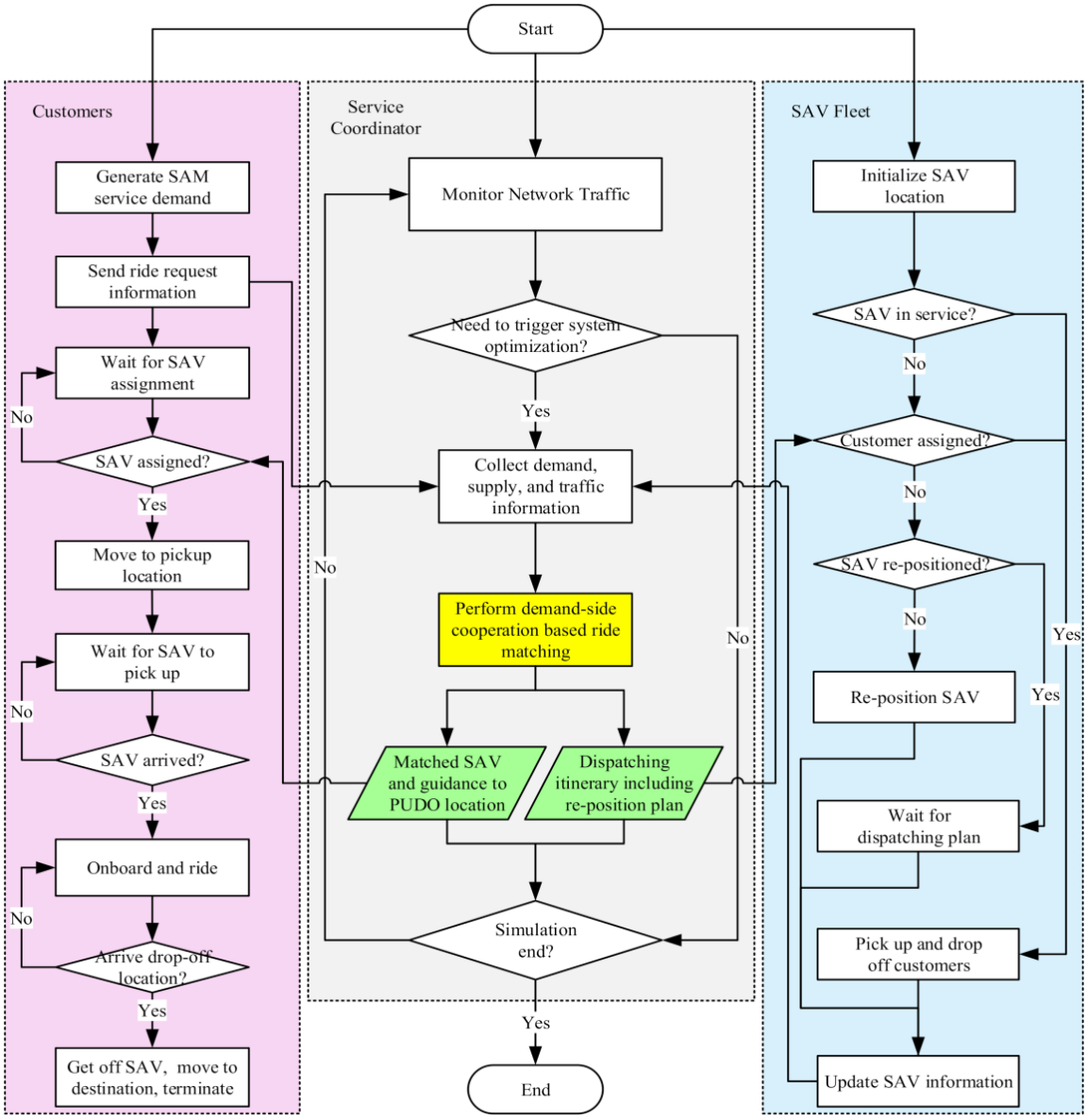


Figure 4-1 The system framework of demand-side cooperative share automated mobility (DC-SAM) service. [136]

to customers, and itineraries as well as suggested routes to SAVs. The customers follow the shortest distance (walkable) paths and the travel times of walking are calculated by the lengths of walking paths divided by the constant walking speed (5 kph), which is set in SUMO. To this end, customers will proceed until the completion of their trips, and SAVs will follow the system's suggestion (or commands) to provide service. If any SAV completes its service round without receiving further requests, it can be re-positioned to

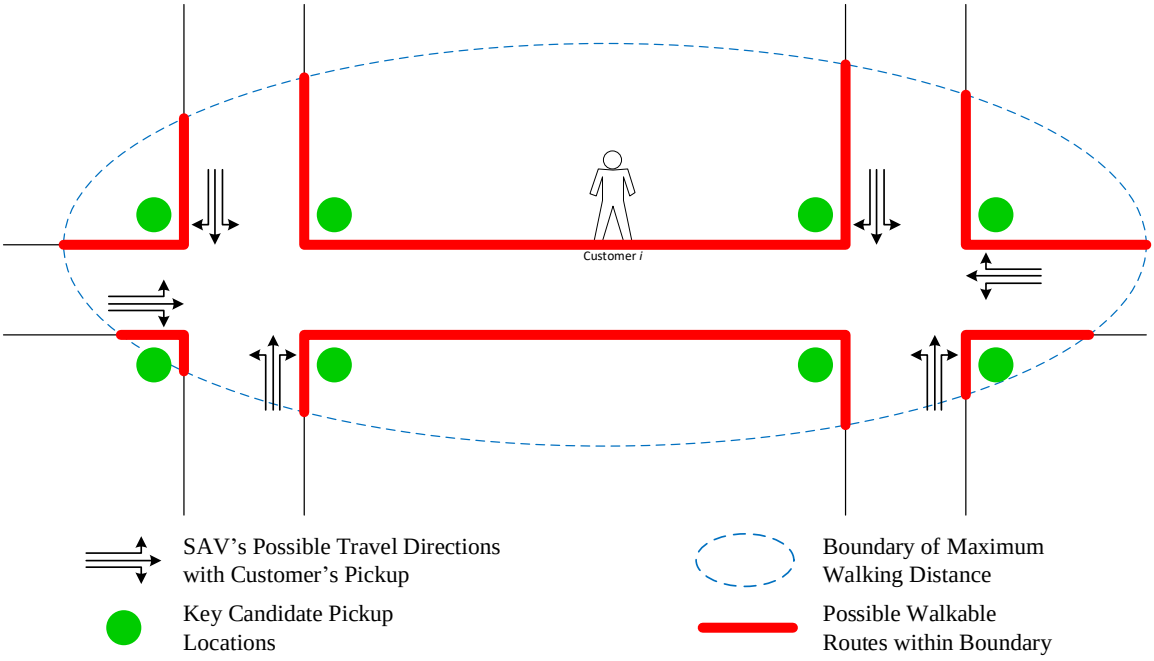


Figure 4-2 An example illustrating alternative pickup locations.

the suggested location by the service coordinator. The system framework, key components (i.e., customers, SAVs, and service coordinator), and associated flowcharts are illustrated in Figure 4-1.

The proposed DC-SAM service operates in an "online" manner in the microscopic simulation environment. Upon the start of simulation, the service coordinator collects the information from both demand (i.e., customers) and supply (i.e., SAVs) sides. At a certain frequency or within a System Optimization Time Window (e.g., every 120 s), the system performs cooperative (involving multiple customers and multiple SAVs) optimal ride matching, based on up-to-date information on unserved requests and available SAVs. A SAV is available for new customer(s) only if the vehicle has delivered to all customers and a new system optimization time window reaches. The procedure continues until all demands are satisfied, or the simulation ends. From the perspective of a customer (demand-side), a random number generator (RNG) is coded to reflect the customer's compliance

(cooperation) to accept alternative PUDO locations. For example, if the generated random number is greater than a threshold, the customer will be cooperative and follow the guidance about alternative PUDO locations suggested by the service coordinator. Otherwise, the customer will stick to door-to-door service without demand-side cooperation. In a more sophisticated mode choice model, many factors, such as walking distance penalty, could be considered as one of the future steps in modeling. Once the trip itinerary gets confirmed, the customer will move to the pickup location or stay at the origin to wait for riding on the matched SAV. When the SAV arrives at the drop-off location, the customer will finish the trip instantaneously or walk to his/her own destination. From the perspective of a SAV (supply-side), it follows the ride matching plans and recommended routes as well as repositioning guidance by the service coordinator, throughout the simulation run to provide the proposed DC-SAM service. Re-positioning of SAV to wait for potential customers is an interesting research topic, and researchers have investigated different strategies and evaluated the energy and mobility impacts [139]. For simplicity, the re-positioned locations in this study are chosen based on the information of last round service (e.g., the destinations of customers). For a more complicated system, these locations may be identified from spatiotemporal predictive analytics of historical SAM service demands [140]. It is also noted that the service fleet is considered "automated" herein because: (1) parameters of SAVs in simulation have been adjusted to model autonomous vehicles (AVs), which are different from background traffic (non-AVs); and (2) all SAVs are assumed to perfectly follow all the guidance provided by the system, including the re-positioning.

4.1.2.2 Alternative PUDO Locations and Ride Matching

As a critical feature of the proposed DC-SAM service, the alternative PUDO locations of the customer's origin and destination may have multiple candidates depending on the maximum walking distance and surrounding network topology. As shown in Figure 4-2, a customer i is located at the origin and is able to walk from the origin to any place along with all directions on the road network within the maximum walking distance (e.g., 0.5 mile), enclosed by the blue ellipse. Nearby walkable routes are indicated as red solid lines along the blocks. Theoretically, any place on the red lines could be considered as an alternative location for picking up the customer. However, given all the potential travel directions (arrows shown in Figure 4-2) of a SAV while completing the customer's pickup, only a limited number of key candidate locations within the maximum walking distance need to be considered. The alternative locations for dropping off in the simulation are identified in a similar way. In practice, other factors such as parking restrictions and unsafe streets could be considered for determining alternative PUDO locations. In addition, to facilitate the modeling of cooperation levels by customers (demand-side), origin and destination locations are also included in the candidate PUDO location set.

The ride matching procedure generates a dispatching plan for SAVs and determines the PUDO locations for customers, which is a critical component of the proposed system. Optimization models are proposed and implemented in the simulation framework, while a heuristic model of ride matching is also introduced in the following section as the baseline scenario for comparison.

4.1.3 Methodology

4.1.3.1 Heuristic Model

This model calculates a ride matching plan according to the spatiotemporal travel information of customers and SAVs in a heuristic manner, which has been used in the early "door-to-door" deployment of SAM services and supply chains [141]. In this model, a spatiotemporal incremental matching algorithm assigns each customer to a SAV and forms the service sequence. First, the potential customers are sorted by the ride request time ascendingly. The earlier the customer's request time is, the higher the priority is to be served. Then, SAVs are ordered by the route distance to Customer r ascendingly. For the nearest SAV v, if it is available, Customer r will be matched to SAV v and both of them are recorded in a customer–SAV mapping dictionary as $M = \{v: [r]\}$. Otherwise, SAV v cannot serve Customer r and the second nearest available SAV v' will be checked until either all potential customers are assigned, or no SAVs are available.

From a SAV viewpoint, it could be assigned with several customers (up to its seat capacity), e.g., $M = \{v: [r_1, r_2, r_3]\}$ for a 3-seat SAV, where the assigned customers are ascendingly ordered by the request time. All customers are delivered in the order of pickup. The algorithm outputs the mapping dictionary M for vehicle routing. With that, a First-Come-First-Served (FCFS) logic is applied to determine the PUDO sequence of the SAV. In addition, all customers have to be picked up first and then delivered in the order listed in the customer-SAV dictionary. For instance, for $M = \{v: [r_1, r_2, r_3]\}$, the service sequence of SAV v is $p(r_1), p(r_2), p(r_3), d(r_1), d(r_2)$, and $d(r_3)$, where $p(\cdot)$ and $d(\cdot)$ denote the SAV's pickup and drop-off actions, respectively. Based on the sequence, the

SAV takes the time-dependent shortest paths (TDSP), which consider the real-time network traffic conditions (e.g., link travel times), to connect PUDO locations.

The main purpose/scope of this heuristic model is to avoid the long customer waiting times for all requests (which is considered as one of the major concerns for pooled TNCs) rather than to maximize the system profit. The heuristic model is severed as a benchmark for comparing with other optimal ride matching models (ODC, Optimization Model with Demand-side Cooperation, and ONDC, Optimization Model without Demand-side Cooperation) proposed in this study. In SUMO, the real-time network traffic condition can be accessed via an application programming interface (API) for shortest path finding. In the real world, such information can be estimated if a large-scale traffic surveillance system is deployed.

4.1.3.2 Optimization Model With Demand-Side Cooperation (ODC)

The ride matching optimization with demand-side cooperation is modeled as a 0–1 binary integer programming problem with a directed graphic structure. In this study, an API is developed in Python for the SUMO simulation to solve this ride matching optimization problem online using the Gurobi Optimizer, which is an efficient solver for integer programming [46]. Before elaborating the model details, parameters and decision variables in the optimization are listed in Table 1. Note that $n_i(j)$ may include $O_{n_i(j)}$ or $D_{n_i(j)}$. Furthermore, D_r^{SAV} is a dummy node for the completeness of the network, i.e., to connect the final drop-off location of last service round with the origin of new service round. Depending on the re-positioning strategy, the cost from $D_{n_i(j)}$ to D_r^{SAV} or from O_r^{SAV} to $O_{n_i(j)}$ may vary.

The objective of the ride matching problem is to maximize the profit of the proposed DC-SAM service, considering both the revenue (positive) and the travel cost (negative) of the SAV fleet. Depending on the SAV availability, each ride request may or may not be served within the instant system optimization time window right after the request generation. For those ride requests that cannot be served instantly, they will be logged in the request list for the ride matching in the future system optimization time window. In the simulation, no waiting time tolerance is set for each request, so all the requests would be served eventually if the simulation time is long enough.

The problem is formulated as follows:

$$\max \sum_{r=1}^{R} \sum_{i=1}^{M} \sum_{j=1}^{N_i} p_{r,n_i(j)} \cdot y_{r,n_i(j)} \tag{4-1}$$

$$- \left[\sum_{r=1}^{R} \sum_{i=1}^{M} \sum_{j=1}^{N_i} c_{r,O_r^{SAV},O_{n_i(j)}} \cdot x_{r,O_r^{SAV},O_{n_i(j)}} \right.$$

$$+ \sum_{r=1}^{R} \sum_{i,k:i\neq k}^{M} \sum_{j=1}^{N_i} \sum_{l=1}^{N_k} c_{r,O_{n_i(j)},O_{n_k(l)}}$$

$$\cdot x_{r,O_{n_i(j)},O_{n_k(l)}}$$

$$+ \sum_{r=1}^{R} \sum_{i,k}^{M} \sum_{j=1}^{N_i} \sum_{l=1}^{N_k} c_{r,O_{n_i(j)},D_{n_k(l)}}$$

$$\cdot x_{r,O_{n_i(j)},D_{n_k(l)}}$$

$$+ \sum_{r=1}^{R} \sum_{i,k:i\neq k}^{M} \sum_{j=1}^{N_i} \sum_{l=1}^{N_k} c_{r,D_{n_i(j)},D_{n_k(l)}}$$

$$\cdot x_{r,D_{n_i(j)},D_{n_k(l)}}$$

$$\left. + \sum_{r=1}^{R} \sum_{i=1}^{M} \sum_{j=1}^{N_i} c_{r,D_{n_i(j)},D_r^{SAV}} \cdot x_{r,D_{n_i(j)},D_r^{SAV}} \right]$$

Subject to

$$\sum_{r=1}^{R} y_{r,O_{n_i(j)}} \leq 1, \quad \forall i,j \tag{4-2}$$

$$\sum_{r=1}^{R} y_{r,D_{n_i(j)}} \leq 1, \quad \forall i,j$$

Each alternative location node, e.g., the jth alternative location for the ith request, in either Origin (for pickup) set or Destination (for drop-off) set, is visited at most once by whichever SAV.

$$\sum_{i}^{M} \sum_{j=1}^{N_i} s_{n_i(j)} \cdot y_{r,O_{n_i(j)}} \leq C_r^{SAV}, \quad \forall r \tag{4-3}$$

For the rth SAV, the number of pickup nodes visited within the same service round (or system optimization time window) should not exceed its associated capacity, C_r^{SAV}

$$\sum_{i}^{M} \sum_{j=1}^{N_i} x_{r,O_r^{SAV},O_{n_i(j)}} \leq 1, \quad \forall r \tag{4-4}$$

(3) From its origin, the rth SAV will visit at most one pickup location.

$$\sum_{i:i\neq k}^{M} \sum_{j=1}^{N_i} x_{r,O_{n_i(j)},O_{n_k(l)}} + x_{r,O_r^{SAV},O_{n_k(l)}} = y_{r,O_{n_k(l)}}, \quad \forall k,l,r \tag{4-5}$$

(4) For any SAV, each pickup node has at most one incoming link, which equals to $y_{r,O_{n_k(l)}}$.

$$\sum_{k:\, k\neq i}^{M} \sum_{l=1}^{N_k} x_{r,O_{n_i(j)},O_{n_k(l)}} + \sum_{k}^{M} \sum_{l=1}^{N_k} x_{r,O_{n_i(j)},D_{n_k(l)}} = y_{r,O_{n_i(j)}}, \quad \forall i,j,r \tag{4-6}$$

(5) For any SAV, each pickup node has at most one outgoing link, which equals to $y_{r,O_{n_i(j)}}$.

$$\sum_{i,k}^{M}\sum_{j=1}^{N_i}\sum_{l=1}^{N_k} x_{r,O_{n_i(j)},D_{n_k(l)}} \leq 1, \quad \forall r \tag{4-7}$$

(6) After the rth SAV picks up all the customers in the origin node set, it will go to the destination node set. In other words, at most, one link will be set up between the origin node set and destination node set.

$$\sum_{i:\, i\neq k}^{M}\sum_{j=1}^{N_i} x_{r,D_{n_i(j)},D_{n_k(l)}} + \sum_{i}^{M}\sum_{j=1}^{N_i} x_{r,O_{n_i(j)},D_{n_k(l)}} = y_{r,D_{n_k(l)}}, \quad \forall k,l,r \tag{4-8}$$

(7) For any SAV, each drop-off node has at most one incoming link, which equals to $y_{r,D_{n_i(j)}}$

$$\sum_{k:\, k\neq i}^{M}\sum_{l=1}^{N_k} x_{r,D_{n_i(j)},D_{n_k(l)}} + x_{r,D_{n_i(j)},D_r^{SAV}} = y_{r,D_{n_i(j)}}, \quad \forall i,j,r \tag{4-9}$$

(8) For any SAV, each drop-off node has at most one outgoing link, which equals to $y_{r,D_{n_i(j)}}$.

$$\sum_{j=1}^{N_i} y_{r,O_{n_i(j)}} \leq 1, \quad \forall i,r \tag{4-10}$$

(9) For any SAV and any request, there is at most one alternative pickup location selected.

$$\sum_{j=1}^{N_i} y_{r,D_{n_i(j)}} \leq 1 \quad \forall i, r \tag{4-11}$$

(10) For any SAV and any request, there is at most one alternative drop-off location selected.

4.1.3.3 Optimization Model Without Demand-Side Cooperation (ONDC)

To demonstrate the benefits of demand-side cooperation, a similar ride matching optimization problem to ODC is formulated without considering any alternative PUDO locations besides the origin and destination specified by the customer (i.e., "door-to-door" service). In other words, the Optimization Model without Demand-side Cooperation (ONDC) can be considered as a special case of ODC where both j's and l's in Equations (4-1)–(4-10) are reduced to 1.

4.1.4 Case Study

The service performance metrics defined in Table 4-1 are directly computed from simulation results, such as trace data, vehicle stops, customer loading data, and service operation plans. They may describe the level of service, mobility efficiency of SAV fleet, and customers' cooperation efforts, which encompass vehicle miles traveled (VMT), vehicle hour traveled (VHT), trip detour factor (TDF), customer waiting time (CWT), customer walking time (WKT), and customer walking distance (WKM). It is noted that TDF can be considered as a surrogate metric to evaluate the customer's loss (in terms of travel distance) due to the shift from a dedicated service to a ridesharing service. Besides that, vehicle energy consumption (VEC) indicates the energy and/or fuel consumed by the SAV fleets serving all the shared riders or customers in the system. In this study, the fuel consumption and tailpipe emissions are estimated by SUMO, based on the Handbook

Emission Factors for Road Transport (HBEFA) where a typical gasoline-powered light-duty vehicle model is adopted.

Table 4-1 Key service performance metrics.

Metrics	Unit	Description
VMT	Vehicle-mile	Vehicle miles traveled
VHT	Vehicle-hour	Vehicle time traveled in hour
TDF	-	Trip detour factor: customer's actual trip distance under the pooled TNC service divided by the trip distance with dedicated service (based on the time-dependent shortest path).
CWT	Second	Average customer's waiting time; waiting time for the matched vehicle moving to the pickup location and picking the customer up.
WKT	Second	Customer's time spent on walking to/from alternative PUDO locations with respect to the origin and destination.
WKM	Mile	Customer's walking distance to/from alternative PUDO locations with respect to the origin and destination.
VEC	Liter (gasoline)	Vehicle energy/fuel consumption for serving all customers.

The proposed demand-side cooperative shared automated mobility service (DC-SAM) simulation was implemented and studied in SUMO with the New York City (NYC) network (see Figure 4-3). The Open Street Map (OSM) provides the detailed roadway network and default traffic signal plans in the region, which is imported in the SUMO platform. Shared automated vehicles (SAVs), SAV routes, background traffic, and movements of customers (either waiting or on-board) are illustrated in Figure 4-3.

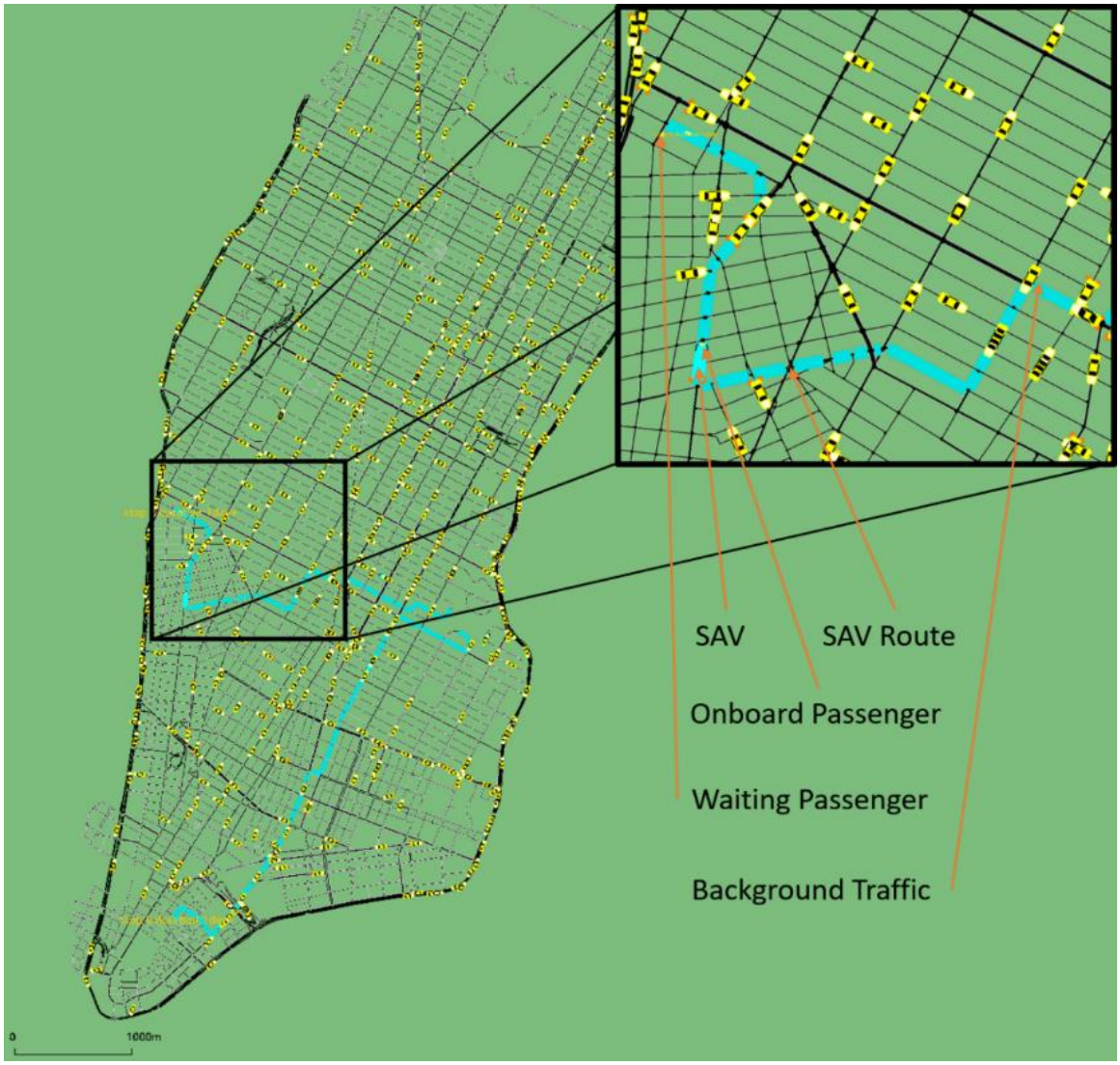


Figure 4-3 SUMO (Simulation of Urban Mobility) for the New York City network.

The SAM demand was extracted from a New York City Taxi and Uber Trips study [142], which provided a vast amount of individual taxi trips in the city from January 2009 to June 2015. In this section, a small sample from one taxi company's trip data on 1 June 2015 was selected. The original information of each trip includes pickup and drop-off (PUDO) locations, PUDO times, customer counts, vendor information, and transaction data (i.e., fare). In the simulation, a total of 140 trips were selected as the baseline SAM demand and synthesized according to the PUDO locations (as the surrogates of origins/destinations) and pickup time (as the surrogate of request time).

Four vehicles with three available seats (i.e., the maximum occupancy per vehicle is 3) were set as the SAV fleet to serve the SAM demand over the network. These SAV behaviors were finetuned in the SUMO simulation by adjusting the driver imperfection indicator to be 0 (i.e., perfect driving), and setting the desired time headway to be 1.5 s, which is different from the background traffic of human-driven vehicles (i.e., 2 s). At the beginning of the simulation, these SAVs were assigned to random locations and got ready for the execution of different ride matching models: 1) heuristic matching; 2) optimal matching without demand-side cooperation (ONDC); and 3) optimal matching with demand-side cooperation (ODC). Within every system optimization time window, the service coordinator monitored available SAVs and unserved demands, based on which a designated pickup/drop-off plan was calculated depending on the selected ride matching models. For the optimization models (ODC and ONDC), a high enough revenue (10,000 units, a unit = 1 dollar or mile) was set to incentivize SAVs to serve as many demands as possible. The travel cost from one place to another is proportional to the route distance of the least-duration path, which depends on the time-varying traffic conditions. After ride matching, the designated SAV would move to pick up and drop off customers according to the assigned itinerary. To guarantee that the demand can be served exhaustedly, a long enough simulation horizon (60,000 steps) was used. In addition, background traffic randomly generated at the rate of one trip per second was introduced into the simulation network uniformly over time. The computing platform to conduct the simulation is set up as follows: CPU—Intel i7 8700; GPU—Nvidia 1660 ti; OS—Windows 10 version 1909; SUMO 1.2.0; and Gurobi 8.1.1.

It is noted that a small-scale (in terms of optimization problem) test was presented in this section to prove the concept (i.e., to demonstrate the proposed DC-SAM service). A major concern is computational efficiency. It is well known that off-the-shelf optimization solvers are not able to address instances with roughly more than 10 vehicles for pooled TNC services with floating targets. In this study, the seat capacity was selected as 3 to be consistent with the setting of a small vehicle. Our trial and error tests showed that four vehicles with capacity of 3 seemed to reach the computational limitation that the Gurobi optimization solver could handle. In addition, the number of alternative PUDO locations would impact the computational time. In this study, each origin or destination of the request in the simulation study has up to 4 alternative locations, including itself. The computational efficiency problem may be solved by more efficient algorithms or more powerful high-performance computing (HPC), which is out of the scope of this work and will be another important future research direction.

4.1.4.1 Determination of System Optimization Time Window

System optimization time window refers to the time interval when all the updated information about riding requests and SAV statuses would be collected for the service coordinator to perform the ride matching. It is considered as one of the most critical parameters that governs the tradeoff between SAM performance and computational efficiency. Sensitivity analyses on this parameter have been conducted to evaluate its impacts. As shown in Table 3, if the time window is short (e.g., 1 s), the service coordinator can respond to the ride request instantly as long as there is any SAV available. This may result in higher overhead in computational time and sub-optimality in terms of system performance because there are less opportunities for SAVs to coordinate with each other

for serving the customers. On the other hand, if the time window is too long (e.g., 500 s), many more customers and vehicles will be considered in the optimization which may lead to significant computational burden for the Gurobi Optimizer and unsatisfactory customer experience. In addition, due to the change in traffic dynamics, a longer time window might not guarantee better system performance.

It turns out that for the test scenarios (i.e., 140 SAM trips, 4 SAVs with 3 seats capacity per vehicle, and the given background traffic), the "best" system optimization time window is 120 s in terms of the majority of performance metrics listed in Table 4-2, such as VMT, VHT, TDF, VEC, and CPU time (in second). For other parameters, e.g., CWT, WKT, and WKM, the values for the 120 s case are comparable to the others. Therefore, in the following simulation studies, the system optimization time window is set as 120 s.

Table 4-2 Sensitivity analysis results on system optimization time window

	500 s	300 s	240 s	180 s	120 s	60 s	1 s
VMT (vehicle-mile)	302	309	293	289	283	304	293
VHT (vehicle-hour)	31.2	30.1	28.8	28.2	27.9	28.1	28.4
TDF	4.5	4.69	4.61	4.55	4.09	4.64	4.46
CWT (s)	869	836	806	745	817	779	846
WKT (s)	468	479	414	474	476	429	513
WKM (mile)	0.48	0.49	0.45	0.49	0.49	0.46	0.52
VEC	101.1	96.3	90.0	91.6	82.2	103.8	88.7
CPU Time (10^3)	18.3	12.0	11.4	14.1	9.0	11.7	12.1

4.1.4.2 Comparison of Different Ride Matching Strategies

The comparative simulation results across all different ride matching strategies, i.e., heuristic, ONDC, and ODC are shown in Table 4-3. From the SAV (or supply-side) perspective, both optimal strategies (ONDC and ODC) can remarkably reduce VMT, VHT,

VEC, and tailpipe emissions in the range of 43.4 to 53.3%, which indicates that the optimization algorithms are much more efficient and sustainable in terms of serving SAM demands with respect to the heuristic strategy. In addition, the proposed ODC strategy can further improve the shared mobility performance compared to the ONDC strategy. For example, the scenario with ODC can reduce VMT and VHT by 4.3 and 4.5%, respectively, compared to the scenario with ONDC. In terms of environmental sustainability, demand-side cooperation can help further drop down the fuel consumption and pollutant emissions in the range of 2.2–5.0%.

From the customer (demand-side) perspective, the results show that even without demand-side cooperation, the optimal ride matching algorithm can significantly decrease both TDF (by up to 55.7%) and CWT (by up to 32.2%), compared to the heuristic model. The ODC strategy can further reduce TDF by 8.1% compared to the ONDC strategy. It is hypothesized that the scenario with ODC strategy may further reduce the possibility of SAV route detour due to the demand-side cooperation. The average CWTs for both optimization scenarios are comparable (about 13 min), which are a bit higher than those from TNC waiting time studies due to the sparsity of both demands and supplies in the large urban network in this proof-of-concept study.

Table 4-3 Simulation results for three strategies

Strategy	Performance Metrics									
	VMT	VHT	TDF	CWT	VEC (L)	CO_2(kg)	CO(kg)	HC(g)	NO_x(g)	PM_x(g)
Heuristic	605.4	51.6	9.24	1159	155.3	361.2	9.33	50.4	150.6	7.06
ONDC	295.6	29.2	4.45	786	86.5	201.1	6.00	31.9	85.7	4.14
ODC	283.0	27.9	4.09	817	82.2	191.1	5.87	31.1	81.5	3.96
ONDC vs. Heur.	−51.2%	−43.4%	−51.8%	−32.2%	−44.3%	−44.3%	−35.7%	−36.7%	−43.1%	−41.4%
ODC vs. Heur.	−53.3%	−45.9%	−55.7%	−29.5%	−47.1%	−47.1%	−37.1%	−38.3%	−45.9%	−43.9%
ODC vs. ONDC	−4.3%	−4.5%	−8.1%	3.9%	−5.0%	−5.0%	−2.2%	−2.5%	−4.9%	−4.3%

4.1.4.2 Comparison of Different SAM Service Demand

To inspect the sensitivity of system performance with respect to SAM service demands, simulation runs with different numbers of requests (where the seat capacity is 3), i.e., 20 trips, 60 trips, and 140 trips (benchmark), were tested and the results are shown in Table 4-4. It can be observed that the performance metrics fluctuate within an acceptable range, which provides some evidence for the system robustness.

As SAM service demand increases, VMT, VHT, and environment-related metrics (such as VEC and tailpipe emissions) increase correspondingly under the same supply capability as expected. For TDF, an apparent decline trend can be seen as the demand level increases. A hypothesis is that higher chances to coordinate the PUDO demands would be anticipated in system optimization with the increase of requests. For other metrics, including CWT, WKT, and WKM, no monotonic patterns (either decrease or increase) are observed, which may be caused by random trip OD locations in such a sparse demand–supply scenario.

Table 4-4 Simulation results for ODC (Optimization Model without Demand-Side Cooperation) scenarios with different demand levels

Metrics	20 trips	60 trips	140 trips (Benchmark)
VMT (vehicle-mile)	58.34	133.74	283.0
VHT (vehicle-hour)	8.30	15.25	27.9
TDF	4.53	4.26	4.09
CWT (s)	800	854	817
WKT (s)	480	464	476
WKM (mile)	0.48	0.47	0.49
VEC (L)	27.1	51.1	82.2
CO_2 (kg)	63.0	118.9	191.1
CO (kg)	2.54	4.43	5.87
HC (g)	13.0	22.8	31.1
NO_x (g)	27.4	51.3	81.5
PM_x (g)	1.41	2.59	3.96

4.1.4.3 Comparison of Different Vehicle Capacity

Vehicle capacity is another vital operational parameter that can impact the service performance. The sensitivity analysis may provide some insight for early deployment of the proposed DC-SAM service with suitable vehicle size. Different seat capacities of SAVs (i.e., 1, 2, and 3 seats) were examined and the results are shown in Table 4-5. Please note that all simulation scenarios here assume 140 trips, 120 s system optimization time window, and 4 SAVs.

According to the simulation results, as the seat capacity grows, most performance measures, such as VMT, VHT, VEC, and pollutant emissions, decrease due to the improvement of supply-side capability and potential system efficiency with optimal ride-matching. Others, e.g., TDF, CWT, WKT, and WKM, increase due to more cooperative efforts being required from the customer side. In particular, for those scenarios with “1 seat”, the situation can be considered as the automated “car-sharing” service dedicated to single origin-destination pair. When comparing “1-seat” scenarios with the benchmark

pooled TNC services (i.e., “3-seat” scenarios), the experiment results indicate that VMT, VHT, and environment-related metrics increase by the range of 48.4–71.4%, but those demand-side related metrics (including TDF, CWT, WKT and WKM) get reduced by the range of 12.2–45.5% due to more dedication to the service.

Table 4-5 Simulation results for ODC in different seat capacities

Metrics.	1 seat	2 seats	3 seats (Benchmark)
VMT (vehicle-mile)	427.5	345.8	283.0
VHT (vehicle-hour)	43.4	33.4	27.9
TDF	2.53	3.49	4.09
CWT (second)	445	627	817
WKT (second)	384	399	476
WKM (mile)	0.43	0.44	0.49
VEC (liter)	122.0	103.2	82.2
CO_2 (kg)	283.7	240.0	191.1
CO (kg)	10.06	7.94	5.87
HC (g)	52.3	41.5	31.1
NO_x (g)	122.3	102.6	81.5
PM_x (g)	6.12	5.05	3.96

4.1.5 Summary

In this study, a demand-side cooperative shared automated mobility (DC-SAM) service framework was developed to allow the customers (i.e., demand-side) to relax their pickup and drop-off (PUDO) locations for improving the overall system efficiency (e.g., reducing the detouring effects of SAVs at the cost of very limited walking loads from customers). The problem was formulated as a binary integer programming and solved by using Gurobi, a commercial optimization solver. The model was implemented in an innovative SUMO-based SAM simulation platform which enables optimal ride matching in an online manner via application programming interfaces (APIs). Results from the preliminary simulation study indicated that the proposed system can significantly reduce the SAV’s operating costs in terms of vehicle-miles traveled (VMT), vehicle-hours traveled (VHT), vehicle energy consumption (VEC), and other pollutant emissions, and

improve the quality of service by reducing the customer waiting time (CWT) and trip detour factor (TDF), compared to the heuristic algorithm. For example, according to Table 4-5, VMT, VHT, and VEC can be reduced by 53.3, 45.9 and 47.1%, respectively, and CWT and TDF decrease by 29.5 and 55.7%, respectively, when using the proposed ODC strategy. In addition, the simulation study showed that more benefits can be obtained by enabling the cooperative efforts from customers under the optimal ride matching strategies with demand-side cooperation. The range of mobility and environmental benefits may vary from 2.2 to 8.1%, depending on the specific metrics. Based on the unique microscopic traffic simulation platform built in this study, we extensively evaluated the proposed system under a variety of settings, such as the number of service requests and SAV's maximum occupancy. It should be noted that the developed microscopic platform can lay a good foundation for further pursing research related to multi-modal operation (e.g., curbside management) and applications of emerging transportation technologies (e.g., connected and automated vehicles)

4.2 Fleet Dispatching Strategy for Battery-Electric Trucks (BET)

4.2.1 Introduction and Background

Over the last decade, significant progress has been made on vehicle fleet electrification, especially for light-duty vehicles. Recently, there have been increasing interest in electrifying medium-duty and heavy-duty vehicles. For instance, heavy-duty battery-electric trucks (BETs) have been successfully demonstrated in drayage application. Now, there are efforts to demonstrate the use of these trucks in regional distribution and other applications.

BETs operating in the regional distribution application will return to home base daily and can be charged overnight. However, due to their limited range and long charging time, care must be taken when planning the routes and schedules of these trucks. The fleet must ensure that the route distance does not exceed the expected range of the BET. If it is necessary to assign a BET to serve a route with distance longer than the expected range, an en-route charging station needs to be identified and the required charging time needs to be included in the schedule. Similarly, if it is necessary to assign a BET to serve multiple routes in the day, required charging times between consecutive routes need to be included in the schedule to ensure that the BET can return to the depot before its battery is fully depleted.

Originated from the traveling salesman problem (TSP), the problem of dispatching vehicles for delivery of items has been investigated for decades in the field of operation research. TSP, an NP-hard problem in combinatorial optimization, aims to find the shortest distance loop among multiple locations. Despite the complexity, many existing methods are able to solve the problem efficiently, such as [115], [143]. Later, as an extension of TSP, the vehicle routing problem (VRP) is also well studied. Instead of a single salesman searching an optimal route, VRP considers the routing of multiple vehicles cooperatively, which is very close to our routing and scheduling problem. Therefore, in this section, we briefly review the studies by firstly combing the variations of VRP, and then introducing the major methods of solving VRP.

The majority of the real-world logistics problems are often more complex than the classical VRP [10]. Therefore, studies of VRP usually try to extend the classical VRP by adding further constraints. For instance, the capacitated vehicle routing problem (CVRP)

is defined by adding the carrying capacity limitation of the vehicles [11], and the vehicle routing problem with time windows (VRPTW) is defined by adding the scheduled time window of each customer [12]. Also, VRP with pickup and delivery (VRPPD) studies the scenario that customers may request services with pickup and delivery [13]. With the electrification of the vehicle fleet, more and more research turns to the topic of the "electric vehicle routing problem" (E-VRP) and its extensions. For E-VRPs, the possibility of recharging at available charging stations is considered such as [14]–[16]. However, the charging strategy enormously increases the complexity of the problem, and a certain level of simplification is conducted in the existing works.

Considering the complexity of the VRP, an essential goal of the algorithm design is to balance the computation time and the optimality of the solution. The metaheuristic algorithms and heuristic algorithms are most commonly used to save the computational load. For example, Bard et al. proposed a branch and cut algorithm [144] to solve the VRP formulated with a mixed-integer linear programming problem. An adaptive large neighborhood search (ALNS) algorithm was proposed by Keskin et al. for the E-VRP with partial recharging strategies [145]. Simulated annealing (SA) and genetic algorithm (GA) approaches were applied by Omidwar et al. to minimize travel distances, time, and emissions [146]. Rizzoli et al. utilized ant colony optimization (ACO) and validated the performance with real-world data [147]. Combining GA, large neighborhood search

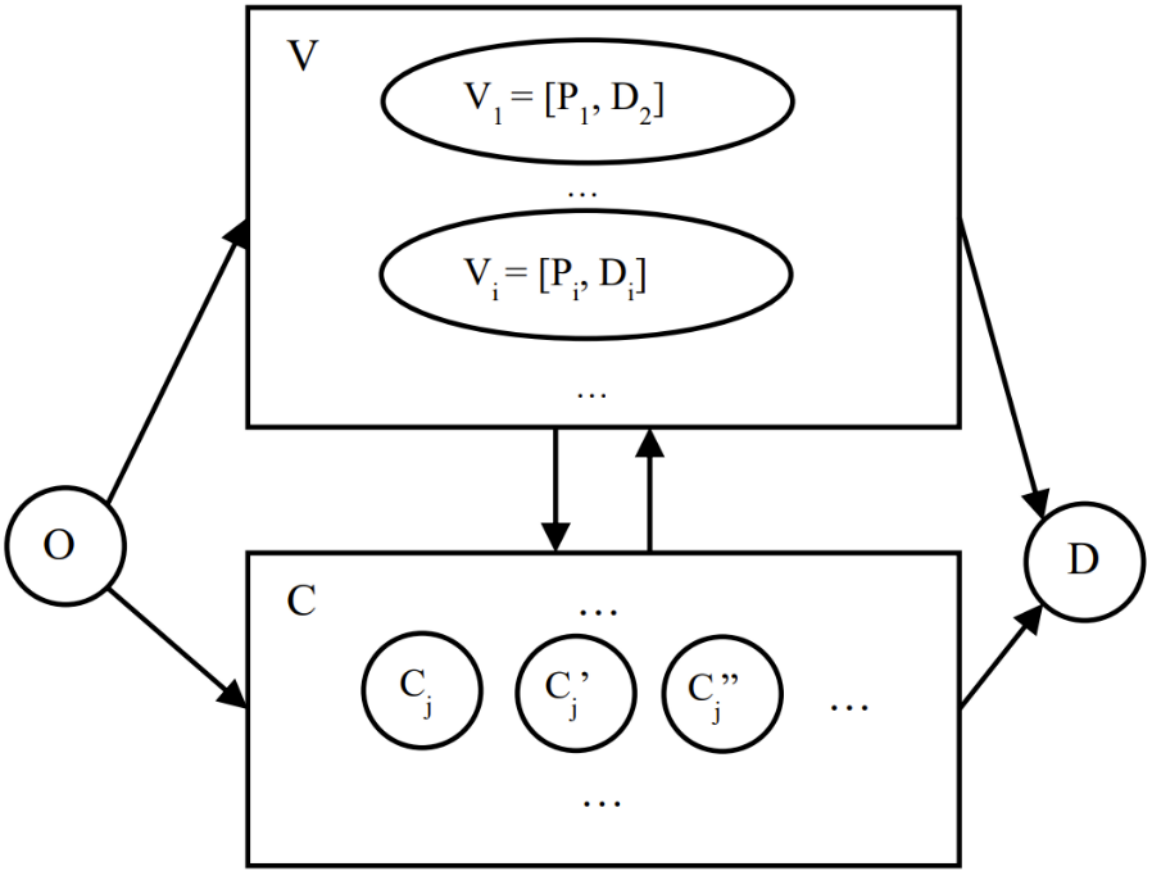


Figure 4-4 Graph model (the arcs within set V and set C are omitted, the arcs between set V and set C are simplified).

(LNS), local search and dynamic programming (DP), Hiermann et al. proposed an algorithm for the problem with the mix of internal-combustion engine vehicles (ICEVs), electric vehicles (EVs), and plug-in hybrid electric vehicles (PHEVs) [148].

4.2.2 Problem Formulation

On each operation day, a set of customers would schedule services for pickup and delivery as shown in blue circles and triangles. The same indices of the locations denote the request from an identical customer. The green box denotes the home base of a fleet of BETs, where a charging station is deployed at the home base to ensure the overnight charging. The BETs should start from and return to the home base within the working

hours. Also, there could be a few charging stations located around the operation region. To extend the range of BETs, opportunistic charging is allowed. Also, the opportunistic charging is set to be flexible to support the partial charging strategy. It should be noted that with the change of route and schedule, the average speed, travel time, and energy consumption from one location to another will change accordingly because of the weight of the goods and the real-time traffic condition. The proposed problem setup is defined as the "electric vehicle routing problem with pickup and delivery, time windows, and partial recharge" (EVRP-PD-TW-PR). If we consider all the key locations as nodes and the routes between nodes as bidirectional links, we can construct a graph model [149], [150] as shown in Figure 4-4.

We define node O to be the home base where the BETs departure from, while node D denotes the same home base where the BETS return. i is an index of a customer, and Vi is a set that contains the pickup and delivery nodes of customer i. The union of all customers set is named set V. j is an index of a charging station, and Cj denotes the charging station j. Also, the charging stations have multiple dummy notes marked by prime as each charging station can be visited multiple times. The union of all charging states with their dummy notes is named set C. The goal of the study is to generate itineraries for the BETs including the specific routes and schedules that could make the whole fleet of vehicles operating in an optimal or near-optimal manner.

Next, to formulate the optimization problem mathematically, the variables are defined in Table 4-6. The variables can be categorized into three types including system input, intermediate variables, and decision variables. The system inputs are the variables that are given before solving the problem. The intermediate variables are those who would

be updated while searching for the solution. The decision variables determine the final itineraries for the fleet of the BETs.

Table 4-6 Variable definitions

Type	Variable	Name	Description
System Input	M	Number of BETs	Number of BETs
	k_i	Node type	1 if charging station, 0 if customer
	q_i	Cargo weight	Positive if pickup, negative if delivery
	L_i	Loading/ unloading time	Time spend at node i
	$TW_i = [e_i, l_i]$	Time window	Scheduled time by customer *i*
	W_m	BET capacity	The maximum carrying capacity for BET m
	B_m	Maximum SOC	The maximum battery capacity for BET m
	T_O	Earliest departure time	The earliest time the BETs can leave from the home base
	T_D	Latest return time	The latest time the BETs should return to the home base
	R_c	Charging cost rate	Cost for recharging in $/kWh
	R_h	Labor cost rate	Salary for the drivers, $/hr.

	R_{ei}	Early penalty rate	Cost rate in $/hr., if vehicle arrive earlier than the scheduled time by customer *i*
	R_{li}	Late penalty rate	Cost rate in $/hr., if vehicle arriver later than the scheduled time by customer *i*
Intermediate Variable	τ_{im}	Arrival time	Arrival time for BET *m* at node *i*
	w_{im}	Current load	Current load for BET *m* at node *i*
	y_{im}	Current SOC	Current SOC for BET m at node i
	t_{ijm}	Travel time	Estimated travel time for BET *m* from node *i* to node *j*
	E_{ijm}	Energy consumption	Estimated electricity consumption for BET *m* from node *i* to node *j*
Decision Variable	τ_{0m}	Actual departure time	The actual time the BET *m* leaves from the home base
	x_{ijm}	Node level route	0 if the route from *i* to *j* is not visited by BET, 1 otherwise
	p_{im}	Charging Rate	Three values for slow, regular, and super charging
	Y_{im}	Finish charging SOC	The SOC when BET *m* leaving charging station *i*

The optimization problem is defined by the equations (4-13)-(4-31), aiming to minimize the overall operation cost of one day. The objective function has three parts,

including energy consumption cost, labor cost, and time window penalties. Though, the time window constraints are formulated as soft constraints. Constraint (4-14) enforces the connectivity of the costumers and constraint (4-15) assures the connectivity of the charging stations. Constraints (4-16) and (4-16) make sure all the BETs should depart from and return to the home base. Constraint (4-17) defines the conservation law that guarantees the number of incoming arcs to each node equal to the number of outgoing arcs. Constraint (4-18) makes sure that one customer's pickup and delivery requests are served by an identical BET, and the pickup should finish before the delivery for the same customer. (4-19)-(4-21) regulate time-related constraints. Among them, constraint (4-20) defines how the intermediate variable τ_{im} updated while searching. (4-20) and (4-21) narrow down the working hours. Constraint (4-23) defines the update logic for the BETs' loading status, and constraint (4-24) and (4-25) specify the initial cargo weight and the cargo weight limit while running. Similarly, (4-26)-(4-30) are SOC-related constraints. It should note that p_{im} is a three-value discrete variable which indicates the charging rate according to three different charging types. Finally, we define x_{ijm} as a binary variable to indicate the selection of the routes.

$$\begin{aligned}\min \quad & \sum_{i\in(O\cap V\cap C),j\in(D\cap V\cap C),m\in M} E_{ijm}x_{ijm}R_C \\ & + \sum_{i\in(O\cap V\cap C),j\in(D\cap V\cap C),m\in M} t_{ijm}x_{ijm}R_h + \sum_{i\in C,j\in(V\cap C),m\in M} (Y_{im}-y_{im})/p_{im} \\ & \qquad \cdot x_{jim}R_h \\ & + \sum_{i\in V,m\in M} [R_e\cdot max(0,e_i-\tau_{im}) + R_l\cdot max(0,\tau_{im}-l_i)]\end{aligned} \tag{4-13}$$

subject to:

a) Graph Constraints

$$\sum_{j\in(D\cap V\cap C),m\in M} x_{ijm} = 1, \forall i \in (O \cap V) \tag{4-14}$$

$$\sum_{j\in(D\cap V\cap C)} x_{ijm} \leq 1, \forall i \in C, m \in M \tag{4-15}$$

$$\sum_{j\in(V\cap C)} x_{ijm} = 1, \forall i \in O, m \in M \tag{4-16}$$

$$\sum_{i\in(V\cap C)} x_{ijm} = 1, \forall j \in D, m \in M \tag{4-17}$$

$$\sum_{i\in(O\cap V\cap C)} x_{ijm} = \sum_{q\in(D\cap V\cap C)} x_{jqm}, \forall j \in (V \cap C), m \in M \tag{4-18}$$

$$\sum_{i\in(O\cap V\cap C)} x_{iP_sm} = \sum_{j\in(V\cap C)} x_{jD_sm}, \forall V_s \in V, s \in V, m \in M \tag{4-19}$$

b) Time constraints

$$\tau_{jm} = (\tau_{im} + t_{ijm} + (1 - k_i)s_i + k_i \cdot \frac{Y_{im} - y_{im}}{p_{im}})x_{ijm}, \forall i \in (O \cap V \cap C), m \in M \tag{4-20}$$

$$\tau_{Om} \geq T_O \tag{4-21}$$

$$\tau_{Dm} \leq T_D \tag{4-22}$$

c) Load constraints

$$u_{jm} = (w_{im} + q_i)x_{ijm}, \forall i \in (O \cap V \cap C), m \in M \tag{4-23}$$

$$u_{Om} = 0 \tag{4-24}$$

$$0 < u_{im} \leq W_m \tag{4-25}$$

d) SOC constraints

$$y_{jm} = ((1 - k_i) \cdot y_{im} + k_i \cdot Y_{im} - E_{ijm})\, x_{ijm} \tag{4-26}$$

$$0 < y_{im} \leq B_m \tag{4-27}$$

$$y_{im} < Y_{im} \leq B_m \tag{4-28}$$

$$p_i = p_1 \; or \; p_2 \; or \; p_3 \tag{4-29}$$

$$y_{Om} = B_m \tag{4-30}$$

e) Variable constraints

$$x_{ijm} = 0 \; or \; 1 \tag{4-31}$$

4.2.3 Methodology

VRP is NP-hard and computationally challenging to find optimal or near-optimal solutions [5]. This is because during real-world dispatch operations, there are many continuous decision variables such as partial charging and flexible departure times. To solve this in a computationally efficient manner, a bi-level hierarchy method has been proposed in the section. In the first level, continuous variables are frozen (i.e., discretized) to narrow down the searching space of the problem. Thereafter, a metaheuristic method, the Ant Colony Optimization (ACO) algorithm (first proposed by [1]), is applied to calculate a near-optimal solution. In the second level, the variables that were frozen in the previous level are fine-tuned around the near-optimal solution from ACO.

4.2.3.1 Coarse Scheduling

Coarse scheduling is done using the ACO algorithm. Inspired by the ant foraging behavior that the optimal path (usually shortest) can be gradually constructed by the

convergence of the ant pheromone trail, the algorithm is used to find the optimal path along a graph. When a group of ants forage, they explore randomly at the beginning. The ants lay pheromone along the path they walk once they find paths to food, and the pheromone on the path can arouse the interest of other ants and increase the probability of exploring that path. This positive feedback loop will eventually result in a path with the highest quantities of pheromone, and such a path is near-optimal.

The ACO algorithm has been proved to be a very efficient algorithm to solve VRP [13]. However, departure time and recharging strategy are the two continuous decision variables that extraordinarily increases the searching efforts. So, the departure times were discretized into multiple instances within a fixed interval, e.g. 8 AM, 10 AM, and so on. Additionally, the charging strategy is set to charge back to 100% battery capacity and does not change. Then, the ACO algorithm is iteratively applied a relatively small data set as described above.

The ACO algorithm has two major stages at each iteration - searching and updating. Initially, a number of ants are released to "explore the graph" randomly. During searching a feasible searching space is determined based on the current locations and the values of the intermediate variables. Then, the possibilities of visiting the feasible nodes to be explored can be calculated based on the existing pheromone level defined by (4-32).

$$p(i,\ j) = \frac{\tau(i,\ j)^{\alpha}\eta(i,\ j)^{\beta}}{\sum \tau(i,\ j)^{\alpha}\eta(i,\ j)^{\beta}} \tag{4-32}$$

where $(i,\ j)$ depicts the arc from node i to node j; $\tau(i,\ j)$ is the current pheromone value of this arc; and $\eta(i,\ j)$ is a heuristic term defined by the *a priori*. For example, it can be defined as $1/distance(i,\ j)$ to attract ants searching the closest node. α and β are two

variables to balance the importance between the pheromone and the heuristic term. The denominator is applied to normalize the overall value from 0 to 1. Based on these visitation possibilities, each ant decides its next visiting node using the roulette strategy.

Once the solution for the current iteration obtained, the algorithm goes to the updating stage. Based on the decisions, we update the values of the intermediate variables and the pheromone level. The pheromone updating strategy shows as follows:

$$\Delta\tau = \sum_{1}^{q} Q/cost \tag{4-33}$$

$$\tau(t+1) = (1-\rho)\tau(t) + \Delta\tau \tag{4-34}$$

where the cost in equation (4-33) is the objective of our optimization problem defined as (4-13); Q is the predefined pheromone increasing rate; q is the number of ants that find the feasible route in current iteration; ρ is the evaporation rate, which is applied to avoid the searching process being trapped into local optima. To increase the convergence speed of getting the near-optimal solution, we always keep one *elite* ant (i.e., the one with the minimum cost in the previous iteration) to the next iteration of the searching process.

4.2.3.2 Fine Scheduling

From the Cooperative Dispatching, Routing, and Coarse Scheduling level, we get rough itineraries for each BET. However, because we simplify the problem in the previous level, the flexibilities of the departure time and the recharging strategy are lost. The fine scheduling process is used to recover the flexibilities by iteratively searching for a better solution to approximate the optima. Figure 4-5 illustrates such an idea in the travel distance-time plot.

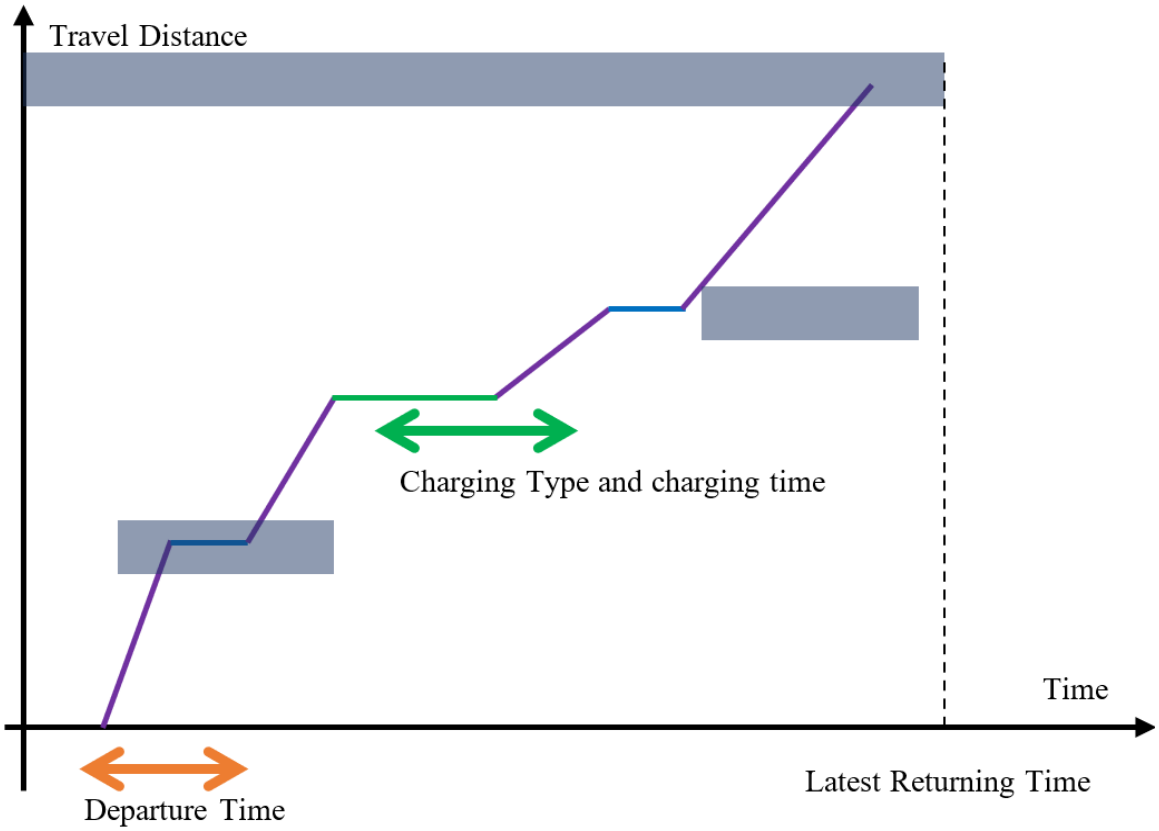


Figure 4-5 Illustration of fine scheduling process.

In the figure, an example trajectory of a BET calculated from the ACO algorithm is depicted, where only one customer is served. The gray bars in the figure depict the time window constraints including the scheduled time for the customer's pickup and delivery as well as the return time. The blue segments denote that the BET stops at the pickup or delivery location. The green segment denotes the process of recharging. As can be observed, by moving the departure time or changing the charging strategy, the trajectory is able to better align the time windows to achieve a lower cost. However, it should be noted that because of the time-variant traffic condition, the shape of the trajectory, specifically, the slope of the moved segments, will also change accordingly. This requires further tuning of the continuous variables. When there are multiple customers and recharging, the tuning

process can be rather complicated. Therefore, we proposed an iterative workflow to solve this tuning process for each BET given in ***Algorithm 4-1***.

Algorithm 4-1*: Fine tuning process*

```
1:   Start from the route calculated by the ACO algorithm
2:   for i from N to 0 (N is the recharging times)
3:     repeat
4:       record cost
5:       if i is not 0
6:         change charging strategy of charging station i to minimize the cost from
           current node to the end
7:         change charging strategy of charging station i to minimize the cost from
           current node to the end
8:         change charging strategy of charging station i to minimize the cost from
           current node to the end
9:         change charging strategy of charging station i to minimize the cost from
           current node to the end
10:      else
11:        change departure time to minimize the overall cost
12:        update traffic condition
13:      end if
14:    until (recorded cost - current cost) < threshold
15:  end
```

4.2.3.3 Heuristic Algorithm

For validating the proposed bi-level hierarchical method, a heuristic algorithm is deployed as the baseline, which uses the greedy strategy to search for the feasible solution of the proposed optimization problem. The pseudocode of the heuristic algorithm is given in ***Algorithm 4-2***.

Algorithm 4-2*: Heuristic algorithm*

```
Sort BETs in the order of SOC level from high to low
for each BET
    manage target nodes based on pickup/delivery history of all BETs
    if the elements in the target nodes less than 3
        add home base into the set of target nodes
    end if
    sort target nodes in the order of the time window constraints from early to late
    for each target node
        if (distance from the current node to the target node + distance from the
        target node to the closest charging station) >= rest range
            select the unvisited node as next node
              break
        else
            continue
         end if
    end
    if no feasible next node found
```

17: select the closest charging station as the next node, use standard charging to the 100% battery capacity

18: **end if**

19: **end**

4.2.4 Case Study

To validate the proposed bi-level hierarchical method, we perform the numerical simulation in this section. The previously introduced heuristic algorithm is carried out as a baseline. The numerical case study setup (in Table 4-7) is specified as follows.

Table 4-7 Case study setup

Type	Variable	Value
Scenario Assumptions	Map scale	70 × 70 miles
	Number of costumers	8
	Number of charging stations	5
	Loading/unloading time	0.3 hour
	Cargo weight	(0, 5] tons
	Working hour	[8 am, 6 pm]
	Buffer time	4 hours
	Traffic condition fidelity	0.5 hour
	Vehicle speed	40, 50, 60 mph
	Charging rate	1, 2, 3 hours to full capacity
	BET range	150 miles
	BET carrying capacity	20 tons
	Labor cost rate	25 dollars/hour
	Early/late penalty rate	90 dollars/hour
	Charging rate	0.3, 0.5, 0.7 dollars/mile
ACO Parameters	Number of ants	150
	Number of iterations	600
	Pheromone weight	0.8
	Heuristic term weight	0.1
	Evaporation rate	0.4
	Pheromone increasing rate	5

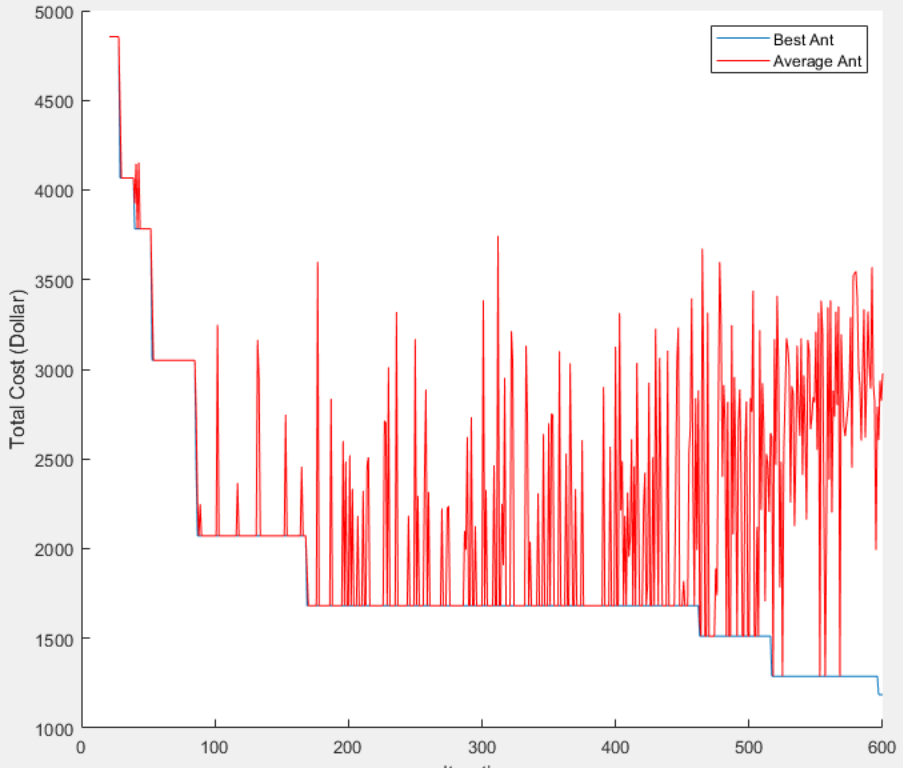


Figure 4-6 Convergence curve of ACO algorithm.

To mimic the regional dispatching scenario, we generate a map on the scale of 70 miles square (i.e., 70 × 70 miles). Within this region, we randomly define 5 charging stations and 8 customers with 16 corresponding pickup and delivery locations. For each location, the loading/unloading time is set to be identical to 0.3 hours, and the cargo weight is randomly set within 5 tons. Also, the time windows are stochastically created no shorter than one hour. We set the working hour of the fleet to be from 8 am to 6 pm, and a 4-hour buffer time after 6 pm is set such that the late returning is allowed with the penalty. The fidelity of the time-variant traffic condition is 0.5 hour. Within the 0.5-hour intervals, we define three levels of the vehicle speed, namely, 40 mph, 50 mph and 60 mph, to express different congestion levels. The distance from one location to another is calculated by their Euclidian distance. As to the charging stations, we use the full charging time, 1 hour, 2 hours and 3 hours to depict three types of charging, i.e. slow charging, regular charging,

and fast charging. We further assume the charging process is at a linear rate. Therefore, if a BET aims to charge from 50% to 100% using regular charging, for instance, it will take 1 hour. From the BET perspective, we use the remaining range to reflect the battery status, and the maximum range of a BET is 150 miles in this case study. Still, we assume a linear relationship between the battery/mileage consumption and the driving distance. There are 4 BETs in the fleet, and the carrying capacity of each BET is identical to 20 tons. Finally, we define the labor cost rate as 25 dollars per hour, the early/late penalty rate as 90 dollars per hour, and the charging rate for three types as 0.3, 0.5 and 0.7 dollars per mile respectively.

The parameters of the ACO algorithm are specified as follows. There are 150 ants searching in parallel for 600 iterations. The importance of the pheromone, α, is set to be 0.8, while the importance of the heuristic term, β, is set to be 0.1; The evaporation rate of the pheromone is 0.4, and the pheromone increasing rate is 5. The convergence curve of the ACO algorithm is shown in Figure 4-6. In the figure, the blue curve shows the performance of the best ant which is kept to the following iteration. The red curve shows the average performance of the ants that found the feasible route in the current iteration. The fluctuation of the red curve indicates the randomicity of the exploration within the searching spacing, which prevents the search from falling into the local optimum.

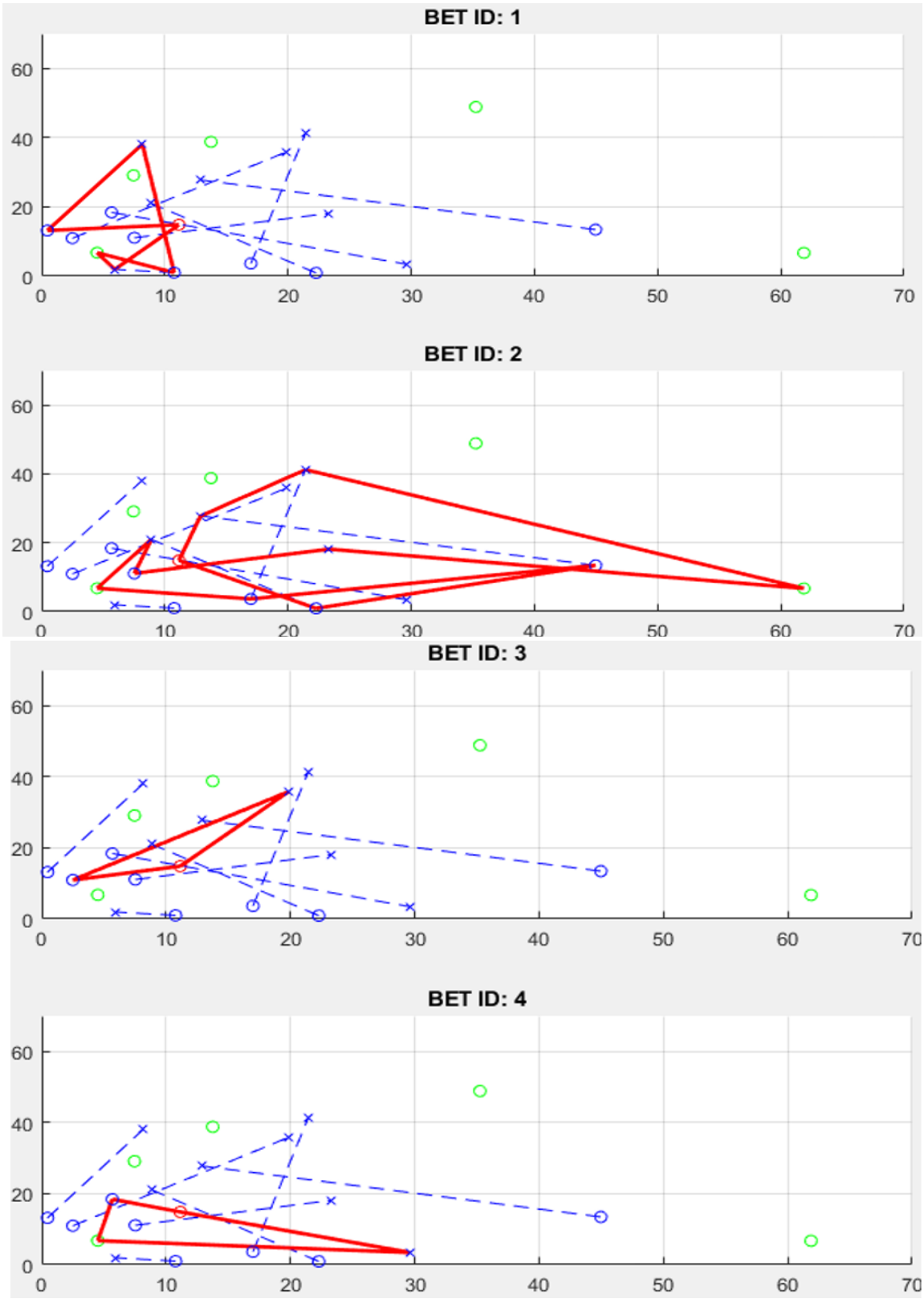


Figure 4-7 Shortest route giving by ACO algorithm.

The routes and the dispatching schemes are shown in Figure 4-7. The red circle depicts the home base, and the green circles denote the locations of the charging station. The blue circles and cross marks connected by the blue dash lines represent pairs of the pickup and delivery locations of corresponding customers. The travel distance-time plot illustrates the solution given by the ACO algorithm (see Figure 4-8). The trajectories are marked with three different colors. The red segments show that the BETs travel from one location to another; the yellow segments show the loading/unloading stages; the green segments show the charging stages. The blue bars represent the time window constraints. From the figure, we can observe that the trajectories align well with the time windows in the conditions of the fixed departure time and charging strategy. A lower-cost alternative can be expected after the Fine Scheduling process.

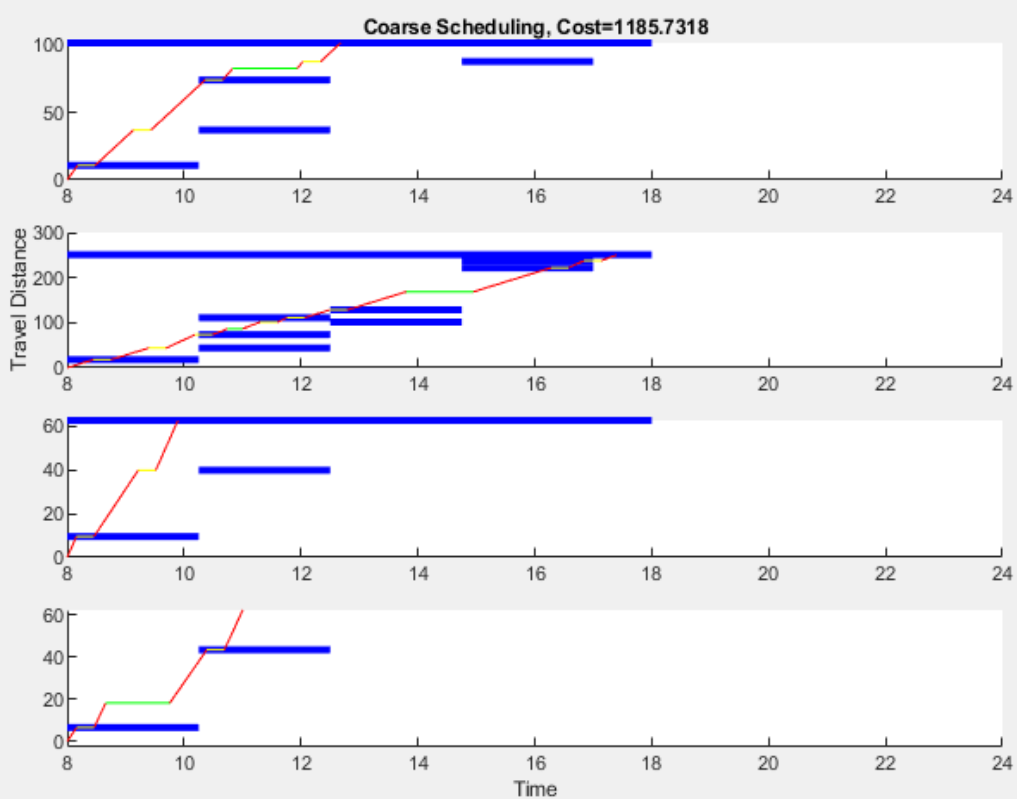


Figure 4-8 Travel distance – time plot giving by ACO algorithm.

The results of the Fine Scheduling algorithm are shown in Figure 4-9, and the costs a re given in Table 4-8. In this process, the results from the ACO algorithm are applied. Therefore, similar trajectories are demonstrated in the figure. By moving the departure time along the time horizon, and adjusting the charging strategy, more time window constraints can be fulfilled. As a result, much lower cost is given. Although the heuristic algorithm is capable of finding a feasible solution, the final cost is far from being acceptable. On the other hand, the result purely from ACO algorithm can already save 30% of the cost. When applying the Fine Scheduling algorithm, the overall cost is reduced by 60%.

Table 4-8 Cost comparison for different algorithms

	ACO Algorithm	ACO + Fine Scheduling Algorithm	Heuristic Algorithm
Cost	1185.7318	673.7622	1681.7737

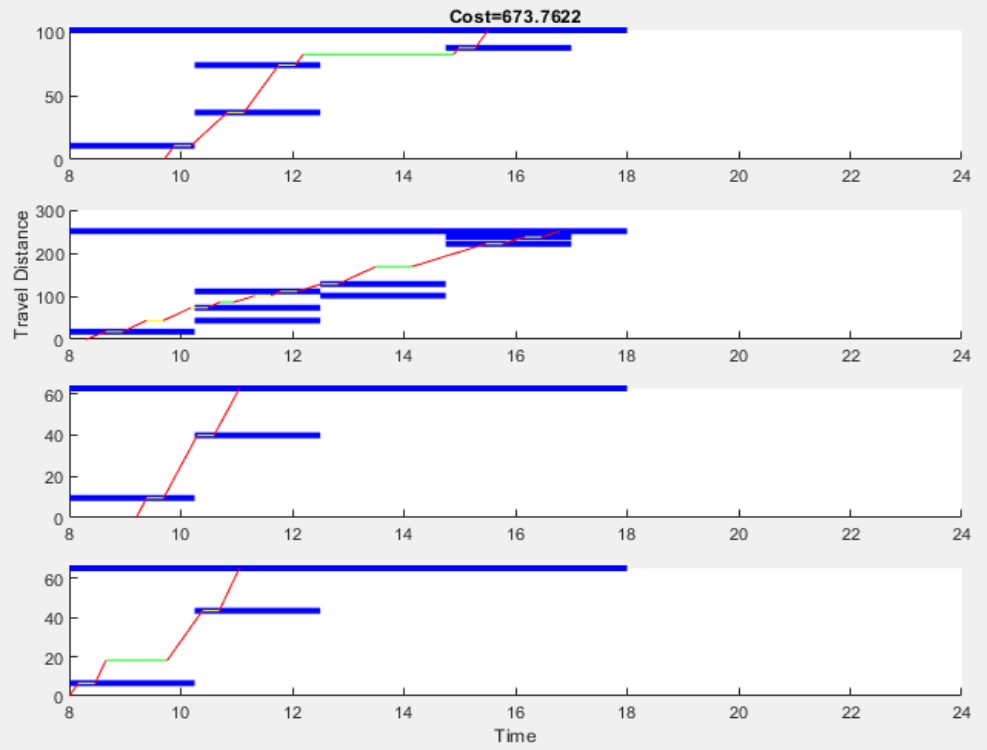


Figure 4-9 Travel distance – time plot giving by Fine Scheduling algorithm.

4.2.5 Summary

In this work, we proposed a methodology to route and dispatch energy-constrained BETs by a two-step optimization of the "electric vehicle routing problem" with pickup and delivery, time windows and partial recharge. We first apply the ACO algorithm coarsely scheduled the BETs. In order to speed up computation, the departure times for BETs were discretized, and the charging strategy was fixed. Then, with the optimization result from the ACO algorithm, we recover the continuity of the problem by the fine scheduling process. The case study results show a 30% saving of the operation cost comparing the

ACO algorithm only with the heuristic algorithm, and a 60% saving comparing the bi-level method with the heuristic algorithm. This is because the ACO algorithm can find out the near optimal solution out of a large set of feasible solutions and allowing flexible departure time and charging strategies can further satisfy the time window requirement of the customers. Some possible directions for future research include adding detailed energy consumption and travel time estimation models to obtain more accurate itineraries and incorporating real-world scenarios to evaluate the system effectiveness.

5 MICROSCOPIC COOPERATION

5.1 Corridor-Wise Eco-Friendly Cooperative Ramp Management System for Connected and Automated Vehicles

5.1.1 Introduction and Background

Ramp merging on highways is one of the most commonly seen scenarios in the modern traffic system. For conventional human-driven vehicles, merging at ramps is challenging and can inevitably cause many traffic-related problems [151]. From the safety perspective, the potential conflicts between on-ramp vehicles and mainline vehicles increase the risk of having accidents. From the mobility perspective, the uncontrolled inflow traffic to the highway network results in congestion. From the environmental sustainability perspective, due to the limited vision range (e.g., by obstructions) and uncoordinated merging behaviors, frequent unnecessary acceleration/deceleration maneuvers may occur in the merging area, which leads to excessive energy consumption and pollutant emissions.

The traditional ramp management method is ramp metering, which regulates the inflow rate of the traffic entering the mainline by using traffic signals located at the end of the. By controlling the traffic light to change between red and green, only a certain number of vehicles can enter the highway mainline during each predefined interval. The controlled inflow rate is calculated based on traffic conditions. Although ramp metering has been deployed in many real-world scenarios and has proven to be a cost-effective operational strategy to reduce mainline congestion, it has a few major drawbacks:

- Conventional ramp metering relies on traffic state estimation from loop detectors that may not be accurate enough to represent real-time traffic conditions and provide detailed guidance for merging maneuvers;
- The traffic signals may introduce unnecessary stop-and-go maneuvers to the on-ramp vehicles, which leads to extra travel time and excessive energy consumption, particularly for heavy-duty trucks;
- The ramp metering system leaves ramp vehicles a much smaller space in which to adjust their speeds to merge into the mainline stream (due to mandatory stops at the meter), which increases the safety risks.

Recently, with the emerging connected and automated vehicle (CAV) technology, researchers have turned their focus to CAV-based cooperative ramp control algorithms that have the potential to avoid the aforementioned drawbacks. CAVs that broadcast information such as position and speed can improve traffic state estimation. Additionally, by elaborately designing their trajectories, CAVs can drive in a cooperative manner (e.g., a vehicle string with closely spaced gaps), which enables smooth merging. As a result, no stop-and-go maneuvering is needed (improved fuel economy), and safety is guaranteed by the automatic control algorithm. Additional benefits include the increased roadway capacity as the headway can be reduced compared to human-driven vehicles. Despite all the advantages of the CAV-based cooperative ramp control approach, many issues need to be addressed to further improve system performance:

- Most of the existing studies only focus on the control of an isolated ramp merging area. The control effect on the down-/up-stream traffic is unknown. The uncoordinated

traffic management across multiple ramps along a corridor may mitigate the benefit provided by the local optimal controller;

- Numerous CAV-based cooperative ramp control algorithms have been developed to improve system mobility and to show environmental benefits, but very few of them are energy-oriented. In addition, most of them assume the first-come-first-serve (FCFS) sequencing strategy for simplicity, which cannot guarantee the system optimum;
- Most of the research validates the system performance within a limited scope (e.g., using numerical simulation or applying the simulation with only a handful of CAVs). However, such validation methods may not be able to explore the long-term impacts on the traffic across a wide variety of scenarios.

5.1.2 Problem Formulation and System Architecture

5.1.2.1 Problem Formula Assumptions

In this study [151], [152], we make the following assumptions for the system development. From the connectivity perspective:

- All vehicles are Connected and Automated Vehicles (CAVs);
- Vehicle information, such as position and speed, can be precisely captured and shared with each other as well as with the central traffic management unit via vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communications, respectively.
- The communication delay and package loss are considered to be zero.

From the control perspective:

- Vehicles can receive and strictly follow control instructions, i.e. accelerations or decelerations, from the central traffic management unit which may be a roadside unit

(RSU) deployed at each ramp. It can collect the data from CAVs within a certain range, estimate the traffic conditions, communicate with other traffic management units along the corridor, and calculate the detailed control instructions for CAVs;

- Cooperative maneuvers for ramp merging are considered only longitudinally. In the simulation study, lateral control is handled by the default model behavior.

5.1.2.2 Vehicle Dynamics

In this study, we assume that all the CAVs are governed by the second-order dynamics:

$$\dot{p}_i = v_i, \dot{v}_i = u_i \tag{5-1}$$

where $i(\in [1, 2, \dots, n])$ is the vehicle index (from downstream to upstream); p and v represent the position and speed of the vehicle, respectively; and u denotes the acceleration/deceleration of the vehicle, which acts as the input to the system. To improve the traffic efficiency, CAVs from on-ramp and/or mainline may be formed into a tight string (or group) at the merging area. Therefore, the states of the overall dynamic system of a group of CAVs can be defined as:

State: $x = (p_1, p_2, \cdots, p_n, v_1, v_2, \cdots, v_n)^T$; If there exists a reference vehicle in front of the whole group (e.g., the last vehicle in the preceding group), then

Observation: $y = (p_r - p_1, p_1 - p_2, \cdots, p_{n-1} - p_n, v_1, v_2, \cdots, v_n)^T$

otherwise,

Observation: $y = (p_1 - p_2, \cdots, p_{n-1} - p_n, v_1, v_2, \cdots, v_n)^T$.

The state vector x includes the positions and speeds of all the vehicles within the group to be controlled, and the observation vector contains the variables to be tracked giving the reference signals. The first entry, $p_r - p_1$, is added when there exists a preceding vehicle in front of the whole group so that the group leader can track the position of the preceding vehicle plus a certain time gap. Therefore, the whole group is able to follow the preceding vehicle smoothly without collision. The system can be written in the following linear form:

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx \end{aligned} \tag{5-2}$$

where $A = \begin{pmatrix} \boldsymbol{O}_1 & \boldsymbol{I} \\ \boldsymbol{O}_2 & \end{pmatrix}$, $B = \begin{pmatrix} \boldsymbol{O}_1 \\ \boldsymbol{I} \end{pmatrix}$, $\boldsymbol{O}_1$ is an $n \times n$ zero matrix, $\boldsymbol{O}_2$ is an $n \times 2n$ zero matrix, and I is an $n \times n$ identical matrix; and

$$\mathrm{C}= \begin{pmatrix} 1,0,\dots,0,0,0 \\ 1,-1,0,\dots,0,0,0 \\ 0,1,-1,0,\dots,0,0,0 \\ \dots \\ 0,\dots,1,-1,0,0,0 \\ \boldsymbol{O}_1 \quad \boldsymbol{I} \end{pmatrix} \in \mathcal{R}^{2n\times 2n}$$

if there exists a preceding reference vehicle in front of the whole group. Otherwise,

$$\mathrm{C}= \begin{pmatrix} 1,-1,0,\dots,0,0,0 \\ 0,1,-1,0,\dots,0,0,0 \\ \dots \\ 0,\dots,1,-1,0,0,0 \\ \boldsymbol{O}_1 \quad \boldsymbol{I} \end{pmatrix} \in \mathcal{R}^{(2n-1)\times 2n} .$$

5.1.2.3 Optimization Problem Formulation

We seek to keep the balance between the convergence speed of observation and the control effort. Therefore, we formulate an optimization problem in the quadratic form as follow:

$$\begin{aligned} \min \quad J = & \frac{1}{2}\sum_{k=0}^{N-1}\{(y_k - r_k)^T Q(y_k - r_k) + u_k^T R u_k\} \\ & + \frac{1}{2}(y_N - r_N)^T Q(y_N - r_N) \\ s.t. \quad & x_{k+1} = Ax_k + Bu_k, \\ & y_k = Cx_k \\ & Acc_{\min} \le u_k \le Acc_{\max} \\ & ((p_i)_k - (p_{i+1})_k) \ge Gap_{\min} \end{aligned} \tag{5-3}$$

where r_k is the gap and speed reference to be tracked; Q and R define the weighting matrices of the objective function to be tuned, respectively, for the system outputs and inputs. $[Acc_{\min}, Acc_{\max}]$ is a feasible input range that a vehicle can achieve, and the boundary values can vary with respect to the vehicle speed. For example, $Acc_{\max}$ at high speed is smaller than that at low speed (considering the power limit). $Gap_{\min}$ is the safety gap to avoid collisions. If vehicle i and vehicle $i+1$ are on the same lane (i.e., either both on the on-ramp or both on the mainline), this constraint should be held strictly. If vehicle i and vehicle $i+1$ are on different lanes (e.g., one is on the mainline while the other is on the on-ramp), this constraint needs to be held when they arrive at (or very close to) the merging area.

5.1.2.4 System Architecture

Figure 5-1. shows the bi-level system architecture of the proposed corridor-wise ramp management system, while no vehicle behavior model is used. At the corridor level, the developed system can calculate the corridor-wise optimal inflow rate at each on-ramp, based on the real-time traffic conditions and estimated traffic states, to guarantee the efficiency of networked traffic. At the ramp level, there are three major modules: 1) grouping CAVs that are in spatial proximity (on both the mainline and on-ramp) as well as satisfy the optimal inflow rate suggested by the upper level algorithm; 2) identifying the

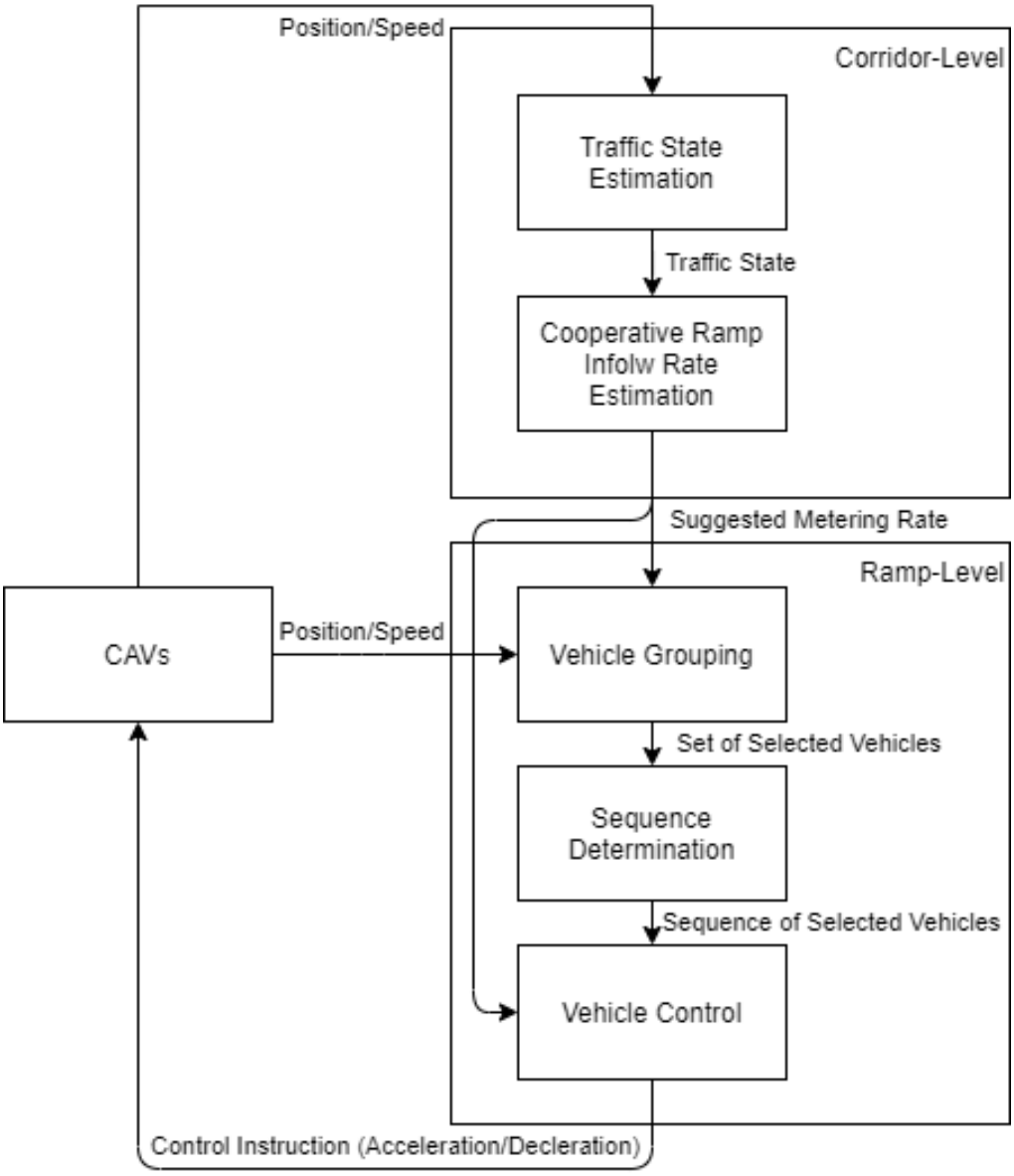


Figure 5-1 System Architecture of the Ramp Management System.

optimal merging sequence for the group of CAVs in terms of energy consumption; and 3) controlling CAVs' speeds in an energy-efficient manner with the model predictive control (MPC) approach to achieve the suggested metering rate at the same time.

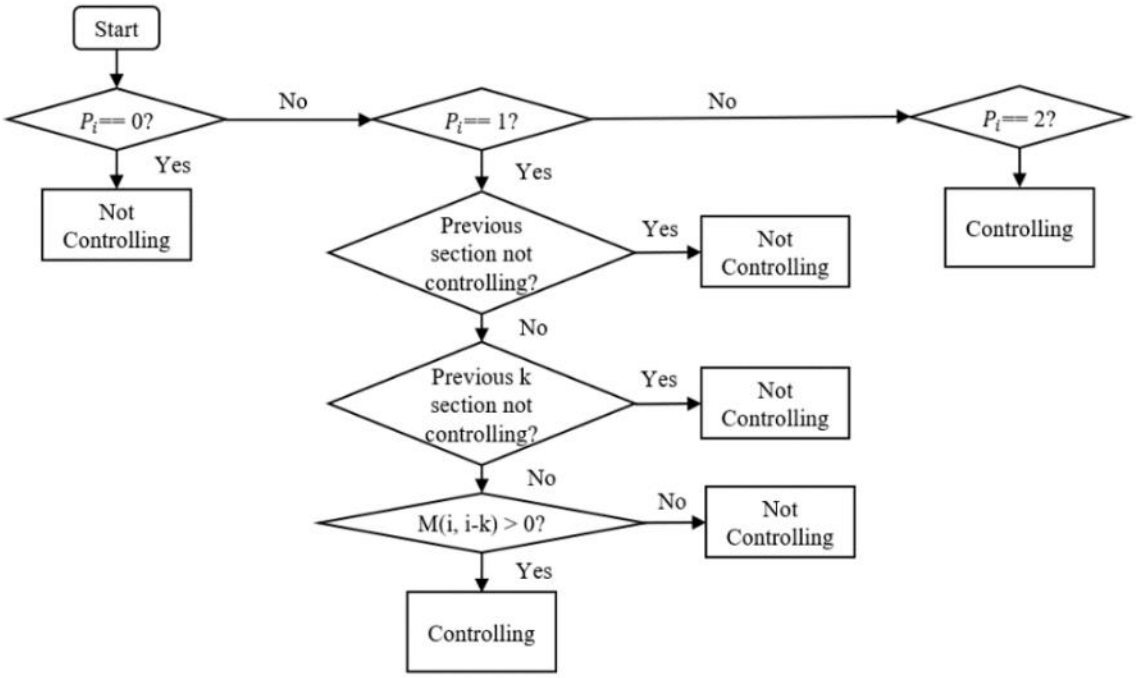


Figure 5-2 Flowchart of corridor-wise ramp metering rate determination.

5.1.3 Methodology

5.1.3.1 Corridor-Level: Metering Rate Estimation

The corridor-level control strategy is to maintain the optimal operation of the corridor by regulating the highway demand to be under the capacity. Therefore, the inflow rate of each ramp needs to be determined for boundary control, which also serves as the constraint for the ramp-level CAV motion control. In this study, we consider the following for metering rate determination:

- The free-flow condition of the mainline traffic should be maintained as much as possible;
- The queue length at on-ramps should be limited to avoid affecting traffic on adjacent arterials;

- The traffic condition should be able to recover from congestion (if any) as soon as possible.

Many existing corridor-wise ramp metering algorithms may take into account traffic conditions along the freeway and calculate coordinated metering rates for multiple ramps simultaneously. In this study, we adapted the Next Generation Stratified Ramp Metering Algorithm proposed by Geroliminis et al [153] and applied to the scenario with full penetration of CAVs. The objective is to balance the ramp waiting time and ramp inflow rate (or the demand and queue lengths at on-ramps) as well as the level of congestion on the mainline, to delay the operation of the breakdown and to accelerate system recovery. The zone is defined as a segment of the highway between two consecutive mainline detector locations (in traditional freeway system). For each section, there is a threat index denote the risk of being a bottleneck. Based on the indices, the controlled ramp can be determined. Figure 5-2 depicts the flow chart to identify the controlled ramp. In the figure, i in current section ID; k is number of consecutive downstream section; P_i is the congestion threat index of the current section; M(i, i-k) is the net inflow between the two locations i and i-k, defined as the sum of on-ramp volumes minus the sum of off-ramp volumes plus the capacity flow difference for all ramps between the two locations.

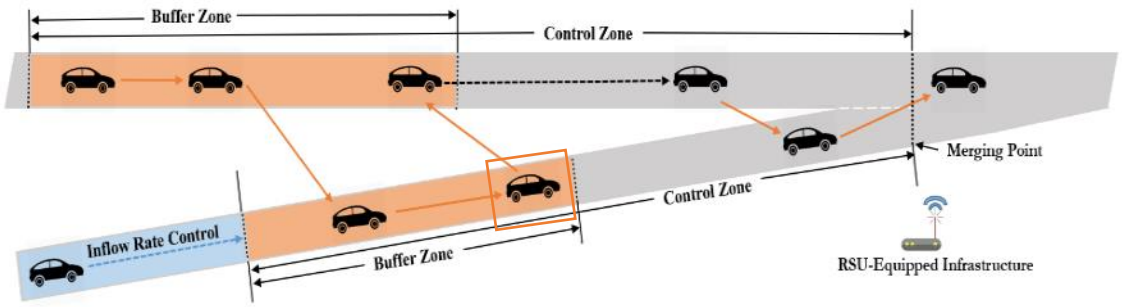


Figure 5-3 Definition of the zones of the proposed ramp-level control.

Then, with the zone identification information, the metering actions matrix are defined in the following Table 2. $c^h(i)$ is the uncongested capacity; q(i) is the mainline demand; d_i is the ramp demand; j is the downstream controlling ramp id; $r_t(j)$ is the current suggested metering rate of location j; $\Delta t_i = T_{crit} - w_t(i)$, where T_{crit} is the ramp delay constraint and $w_t(i)$ is the current maximum waiting time at ramp i; K_1 and K_2 are contribution parameters depicting the importance of breakdown on the ramp and on mainline; τ_w is the safe time-to-breakdown defined for the ramp; $T_t^w(i)$ and $T_t^k(i)$ are the estimated time remaining to congestion on the mainline and ramp. The values of the parameters and the variables are calculated based on historical or real-time data.

5.1.3.2 Ramp-Level: Movement Control and Rate Regulation

At the ramp-level control, three major modules are developed to enable the energy-efficient cooperative maneuvers of CAVs, i.e., vehicle grouping, sequence determination, and longitudinal speed control.

The vehicle grouping module is essential to ensure that the system can manage the continuous flow of traffic. We first define control zone and buffer zone shown in Figure 5-3. Both mainline and ramp have these two types of zones. The control zone is a segment of road from the upstream of the merging area to the merging point. In the control zones, a

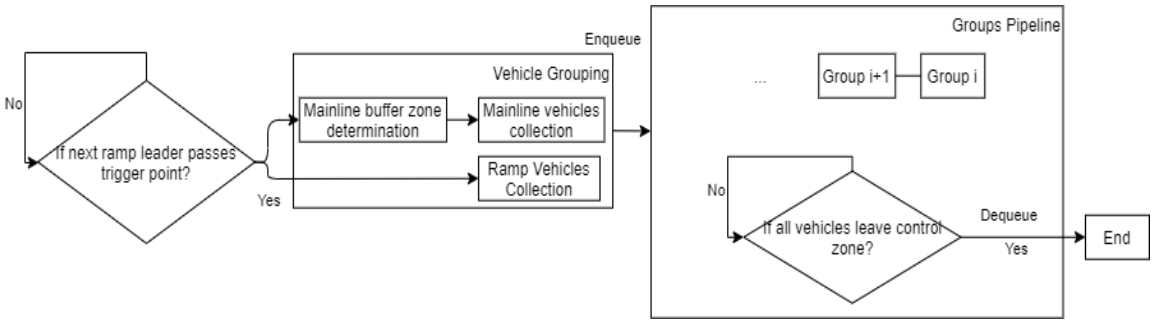


Figure 5-4 Flowchart of the vehicle grouping.

roadside unit serves as a centralized traffic controller to receive and process the incoming information from CAVs and send control signals back to CAVs. Then the CAVs should strictly follow the instructions such that the platoon with designed time gaps can be formed. The buffer zone, on the other hand, locates at the upstream portion of the control zone. In the Buffer zones, the incoming vehicles are monitored and to be grouped.

Before describing the details of the event-based vehicle grouping strategy (see Figure 5-4), we should define the concept of ramp leader. Ramp leader is an on-ramp vehicle that triggers the grouping process. Ramp leader is the first on-ramp vehicle that does not pass through the buffer zone and has not been controlled. We first assume that there exists a group of vehicles under control. Then, the current ramp leader would be the first vehicle on the ramp who is following the group. Once the ramp leader passes the downstream boundary of the on-ramp buffer zone or the trigger point, the grouping process is activated, and the vehicles in the buffer zones of both mainline and ramp is collected as a group. At the same time, the current ramp leader is changed accordingly. If there is no group of vehicles under control, then, the first on-ramp vehicle is the current ramp leader until it passes the trigger point. It should be noted that the control for a group happens right after the group is determined until all the vehicles in the group drive through the control zone. It possible that the new group is constructed while the previous group is still driving

within the control zone. As a result, there could be multiple groups in parallel. Therefore, we use a pipeline architecture of store, process, and control the groups of the vehicles.

For each ramp, the length of the control zone and buffer zone are predefined to better adapt the physical condition. For the corresponding mainline area, the downstream side of the buffer zone is fixed while the length of these zone changes dynamically according to the real-time traffic condition and the suggested metering rate from the corridor-level module. In this way, the vehicle grouping module is able to collect an appropriate number of mainline vehicles into the group. The length of the mainline buffer zone is determined with the following equation:

$$L_{main} = \frac{q_{main}}{q_{suggested}} \cdot n/d_{main} \tag{5-4}$$

where q_{main} is the mainline traffic flow known from corridor traffic condition; $q_{suggested}$ is the suggested on-ramp inflow rate assumed to be known; n is the number of on-ramp vehicles currently in the buffer zone; d_{main} is the mainline density.

After CAVs are grouped respectively, those within the same group will be controlled together coordinately. To enable efficient merging maneuvers, it is critical to determine the merging sequence of CAVs. In this research, we propose a three-step Optimal Sequence Determination process as follows.

Feasible sequence generation: In this step, all the possible entrance sequences of CAVs in a group are first generated. As we assume that the vehicles on the same lane cannot overtake their preceding vehicles, the number of all the feasible sequences equals

to $P(m+n,n)$, where $P(\cdot)$ is the permutation operation; m is the number of mainline vehicles.

Linear quadratic tracking: The LQ tracking algorithm is applied to solve the optimization problem as defined previously. Different merging sequences correspond to different initial states of the system. Using the finite-horizon linear quadratic tracking algorithm, we are able to calculate the control inputs and the specific trajectories of the vehicles for each possible sequence. The Q and R are the weight matrices of the objective function. By tuning these two matrices, the convergence speed of observations and the control effort can be balanced. The control input can be obtained by solving the algebraic Riccati equation [154]:

$$\begin{cases} S_N = C^T Q_N C \\ V_N = C^T Q_N r_N \end{cases} \tag{5-5}$$

$$\begin{cases} S_i = C^T Q C + A^T S_{i+1} - S_{i+1} B (R + B^T S_{i+1} B)^{-1} B^T S_{i+1} A \\ V_i = \{A^T - A^T S_{i+1} B (R + B^T S_{i+1} B)^{-1} B^T\} V_{i+1} + C^T Q r_i \end{cases} \tag{5-6}$$

$$\begin{cases} K_i = (B^T S_{i+1} B + R)^{-1} B^T S_{i+1} A \\ K_i^v = (B^T S_{i+1} B + R)^{-1} B^T \end{cases} \tag{5-7}$$

where N is the predefined finite horizon; i is the discrete-time index for each iteration; K_i is the feedback gain; and K_i^v is the feed-forward gain. S_i, V_i, K_i, and K_i^v can be found iteratively backward in time. Then the control input is given by $\mu_i = -K_i x_k + K_i^v V_i$. Therefore, for each possible sequence, the speed profile can be calculated.

Energy consumption estimation: Based on the calculated speed profile under each possible sequence, the corresponding energy consumption can be estimated for the vehicle with different classifications (such as passenger cars, transit buses, or trucks) and

powertrains (e.g., internal combustion engines or electric motors). In the simulation, we assume all the vehicles are passenger cars and the road grade is trivial, and we will evaluate the system performance for both gasoline-powered vehicles and electric vehicles. In addition, for gasoline-powered vehicles, we refer to the model proposed by [155]:

$$f_V = b_0 + b_1 v + b_2 v^2 + b_3 v^3 + a(c_0 + c_1 v + c_2 v^2) \quad (5\text{-}8)$$

where b_i and c_i are the model parameters calibrated by different driving conditions; v and a are the speed and acceleration of the vehicles.

For electric vehicles, the model developed in our previous work [156] is used:

$$P = f(v, a) = f_0 + l_1 v\cos(\alpha) + l_2 v\sin(\alpha) + l_3 v^3 + l_4 va + \quad (5\text{-}9)$$
$$l_5 v^2\cos(\alpha) + l_6 v^2\sin(\alpha) + l_7 v^4 + l_8 v^2 a$$

where l_i is the model parameter calibrated by different driving conditions; α is the road grade (rad).

Finally, the sequence with the least aggregated energy consumption is selected as the optimal scenario, and the control algorithm in the next section will be applied to this sequence.

As aforementioned, we focus on the longitudinal speed control to engage energy-efficient cooperative ramp merging of CAVs in this study. There are two parts in this module. The first part is an MPC controller which applies a receding-horizon LQ tracking algorithm to control the vehicles in each group. To match the predicted energy consumption in the optimal sequence determination step, the same control parameters are used as in the associated finite-horizon LQ tracker. Also, there are multiple MPC

controllers running in parallel to control multiple groups in the pipeline. The second part of the speed control module is a ramp inflow rate controller. Based on the suggested inflow rate from corridor-level control, the speed of the ramp leader within each group is regulated to fulfill the boundary constraint. Towards this end, we first calculate the estimated time of arrival (ETA) of the ramp leader, assuming it would follow the preceding vehicle without any external control. And the Intelligent Driver Model (IDM) [157] is used for the prediction of its ETA.

$$\dot{v} = a(1 - \left(\frac{v}{v_0}\right)^{\delta} - \left(\frac{s^*(v - \Delta v)}{s}\right)^2 \tag{5-10}$$

$$s^*(v - \Delta v) = s_0 + vT + \frac{v\Delta v}{2\sqrt{ab}} \tag{5-11}$$

where v_0 is desired speed; s_0 is the minimum spacing; T is the desired time headway; a is the maximum vehicle acceleration; and b is the comfortable braking deceleration.

Then, we compare the predicted ETA with the recommended time of arrival (RTA) based on the suggested inflow rate from the corridor-level control, using the following equation:

$$t_{expect} = \frac{n_{ramp}}{q_{suggested}} \tag{5-12}$$

where n_{ramp} is the number of ramp vehicles within the previous group; $q_{suggested}$ is the metering rate from the corridor-level module.

If ETA is smaller than RTA, the ramp leader should be controlled to slow down and pass the trigger point at the recommended time. Or if ETA is larger than RTA, the ramp leader is not controlled. To achieve better energy consumption, we apply the dynamic

eco-driving strategy (given the target arrival time [158] for the speed control of the ramp leader. Dynamic programming is used to obtain the least energy consumption solution.

5.1.4 Simulation Study

We use the simulation environment introduced in the previous section, PTV-VISSIM Two-ramp Freeway Scenario, to evaluate the effectiveness of the proposed management system. The lane change maneuver is controlled by the lateral behavior model provided by VISSM with the default parameters. The desired speeds for both mainline and merging traffic are 74.6 mph (or 120 km/h), and the initial speed of the on-ramp vehicles entering the network is set to be 11.2 mph (or 18km/h). The weight matrices for the quadratic objective function, i.e. Q and R, are tuned manually to adapt to the road network. We set the Q and R to be diagonal matrices. The position weight between both two mainline vehicles is 5, between both two ramp vehicles is 30, and between one ramp vehicle and one mainline vehicle is 6; the speed weight for the mainline vehicle is 50 and for the on-ramp vehicles is 30; the acceleration weight for the mainline vehicles is 2800 and for the on-ramp vehicles is 2500.

We also set two control groups (with human-driven vehicles) for comparison: the first one applies the ramp metering strategy with the same metering rate at each ramp as the test group that is obtained from the corridor-wise SRM algorithm; and the second one had no external control for merging maneuvers (i.e., no metering). There are no communication or driving assistance for the vehicles in both control groups.

Simulation scenarios are set up to evaluate different traffic conditions by defining the average input volume from each entrance of the network, including mainline, ramp 1,

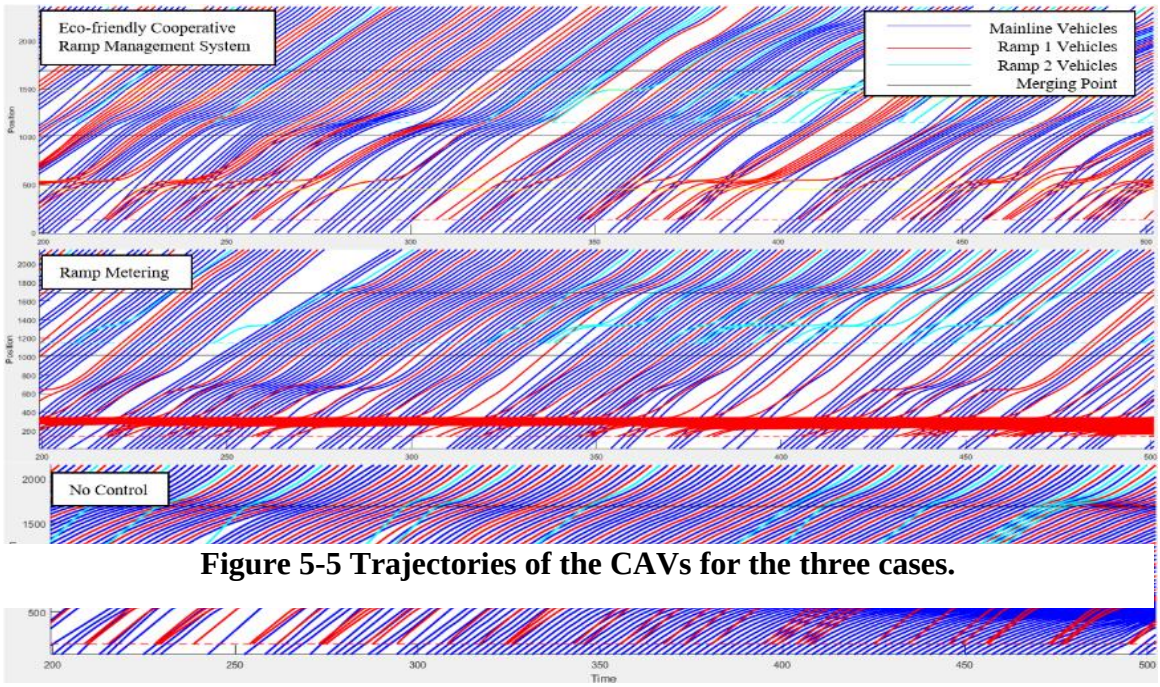


Figure 5-5 Trajectories of the CAVs for the three cases.

and ramp 2. Within the 20-minute simulation period, there exist two phases of traffic demands. In Phase 1 (from 0 to 600 seconds), we set the mainline input to be 1200 passenger-car-unit/hour/lane (pcu/hr/ln), ramp 1 input to be 500 pcu/hr/ln, and ramp 2 input to be 200 pcu/hr/ln. In Phase 2 (from 600 to 1200 seconds), we set the mainline input to be 1200 pcu/hour/lane, ramp 1 input to be 300 pcu/hr/ln, and ramp 2 input to be 200 pcu/hr/ln. During each phase, the incoming vehicles are randomly spawn into the network, and after phase 2, the simulation would keep running until the network clears up.

5.1.4.1 Results Comparison for Gasoline Vehicles

Figure 5-5. shows the example trajectories of CAVs (from 200 to 500 seconds) generated from VISSIM. In the top subplot, the proposed eco-friendly cooperative ramp management system can not only smooth the speed profiles of ramp vehicles, but also mitigate the potential congestion of downstream mainline traffic by regulating the inflow rates of both on-ramps. In the middle subplot, although the mainline traffic is not affected by the merging vehicles, traffic on ramp 1 significantly slows down and forms a long queue

due to the ramp metering. In the no control case as shown in the bottom subplot, the merging vehicles cause the shockwaves on mainline which propagates upstream and eventually results in congestion on both mainline and ramp 1.

To investigate the performance of the proposed system, mobility is measured by the network efficiency, $Q = \frac{VMT}{VHT}$, where VMT denotes the vehicle-miles traveled of all CAVs in the network; and VHT denotes the vehicle-hours traveled in the network. The energy consumption for the gasoline vehicles (with Eq. 8) is estimated in the unit of miles per gallon (mpg). And the results are shown in the following table.

Table 5-1 Simulation Results of Mobility and Energy Performance for Gasoline Vehicles

		Mobility (mph)	Energy (mpg) The bigger the better
Proposed	Overall	59.10 (48.6%) (79.4%)	44.40 (35.1%) (0.8%)
	Mainline	62.17 (-4.9%) (107.3%)	41.45 (10.0%) (-4.0%)
	Ramp 1	53.14 (210.0%) (-7.7%)	51.75 (119.2%) (8.4%)
	Ramp 2	50.60 (-2.7%) (14.9%)	65.12 (112.1%) (56.0%)
Ramp Metering	Overall	39.76	32.87
	Mainline	65.40	37.67
	Ramp 1	17.14	23.61
	Ramp 2	52.03	30.70
No Control	Overall	32.95	44.05
	Mainline	29.98	43.19
	Ramp 1	57.55	47.73
	Ramp 2	44.05	41.74

As can be observed from Table 5-1, the overall mobility of the proposed system is increased by 48.6% and 79.4%, compared to the ramp metering case and no control case, respectively. Also, vehicles can achieve higher mpg with the proposed ramp management system, compared to both cases (35.1% and 0.8%, respectively) due to the mitigation of traffic congestion. In addition, it turns out that although ramp metering can significantly

mitigate the mainline congestion, it penalizes too much (in terms of both mobility and energy) on ramp traffic, in particular for ramp 1.

The safety performance is analyzed using the time-to-collision (TTC) distributions (obtained from the Surrogate Safety Assessment Model [159]) as shown in Figure 5-6. The x-axis denotes the TTC in second, and the y-axis denotes the number of events in the corresponding bins. The figure shows that there is no event of TTC being less than 1.5 seconds for the proposed ramp management system. This indicates that the proposed system would result in a trivial number of potential conflicts and significantly improve the safety performance, compared with the other two cases.

5.1.4.2 Results Comparison for Electric Vehicles

The simulation of electric vehicles is also conducted in this study. For the EV analysis, we use kWatt/100mile as the measurement of the energy consumption. The results are shown in Table 5-2.

Table 5-2 Simulation Results for Electric Vehicles

		Mobility (mph)	Energy (kWatt/100mile) The smaller the better
Optimal Control	Overall	62.31 (56.7%) (89.1%)	40.91 (-24.0%) (-2.5%)
	Mainline	65.28 (-1.8%) (117.7%)	44.15 (-7.6%) (3.5%)
	Ramp 1	57.14 (233.4%) (-0.7%)	33.75 (-54.2%) (-13.0%)
	Ramp 2	54.03 (3.8%) (22.7%)	29.04 (-44.9%) (-37.3%)
Ramp Metering	Overall	39.76	53.84
	Mainline	65.40	47.76
	Ramp 1	17.14	73.68
	Ramp 2	52.03	52.72
No Control	Overall	32.95	41.98
	Mainline	29.98	42.64
	Ramp 1	57.55	38.80
	Ramp 2	44.05	46.33

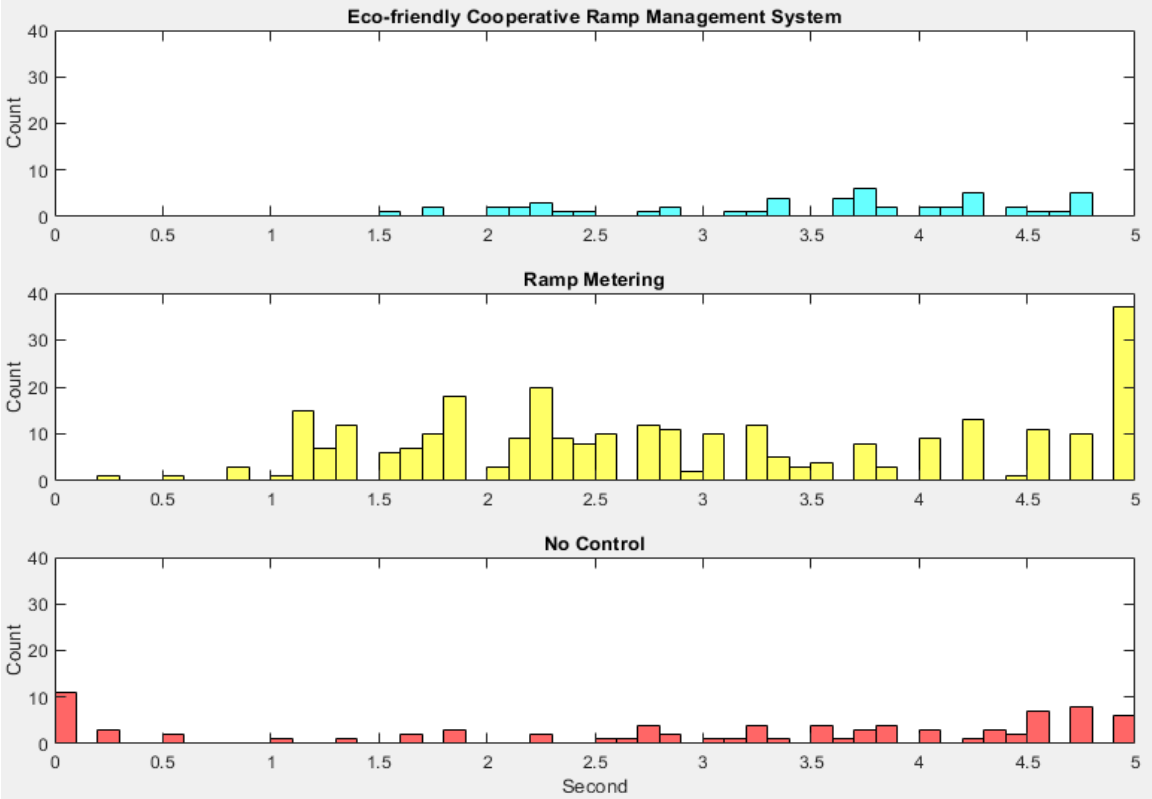


Figure 5-6 Distributions of TTC.

The mobility performance of the EVs scenario is like the Gasoline vehicles, and the trend of energy performance is also the same. The major seasons cause the mobility differences between the electric vehicles and the gasoline-powered vehicles fall into two aspects. First, different energy consumption models applied may change the determination of the entrance sequence. Second, when controlling the ramp leader to slow down, the energy models play role in the optimization process. Although proving the similar mobility improvement, due to the regenerative braking feature of the electric vehicles, a smaller level of the energy saving can be achieved.

5.1.5 Summary

This study proposed a corridor-wise eco-friendly cooperative ramp management system for connected and automated vehicles (CAVs). At the macroscopic level (corridor

level), the optimal inflow rate for each ramp along the corridor is calculated by the Next Generation Stratified Ramp Metering Algorithm. At the microscopic level (ramp level), we developed a dynamic control strategy for CAVs to achieve rate regulation, group determination, and motion control in an energy-efficient manner. With the microscopic traffic simulation in VISSIM, the effectiveness of the system is demonstrated from the perspectives of safety, mobility, and environmental sustainability, compared to cases of human-driven vehicles with and without ramp metering control. To understand the robustness of the proposed system with respect to different powertrain technologies, we also evaluated the performance for scenarios with gasoline vehicles and electric vehicles, respectively.

6 SOLVING THE MIXED TRAFFIC PROBLEM

6.1 Infrastructure-Side Sensing

6.1.1 Introduction and Background

Roadway intersections are the most common yet complicated traffic scenario that connects two or more road segments and involves potential conflicts between multiple types of road users whose view can be partially occluded because of road geometry. As a result, it is a highly challenging task for road users to move at intersections in terms of safety, mobility, and environmental sustainability. From a safety perspective, about 40% of crashes in the U.S. are intersection-related, which causes more than 20% of traffic fatalities yearly [160]. From the mobility perspective, traditional traffic control methods such as traffic lights and stop signs assign priority levels by forcing road users to stop and wait. Although existing research uses adaptive signalized control to reduce the waiting time and improve the throughput of the network to some degree, further optimization can be achieved by introducing higher-level cooperation. From the environmental sustainability perspective, the stop-and-go maneuvers of road users, especially for heavy-duty vehicles, waste a large among of energy.

To address the aforementioned challenges, more and more researchers have turned to CAV technologies. For example, Cooperative Driving Automation (CDA), proposed by Federal Highway Administration, US DOT [161], is expected to reduce 20% of fuel consumption at intersections and double the road network capacity if vehicles can communicate between vehicles, infrastructure devices, and road users (e.g., pedestrians and cyclists), and make decision cooperatively. To support this, many communication

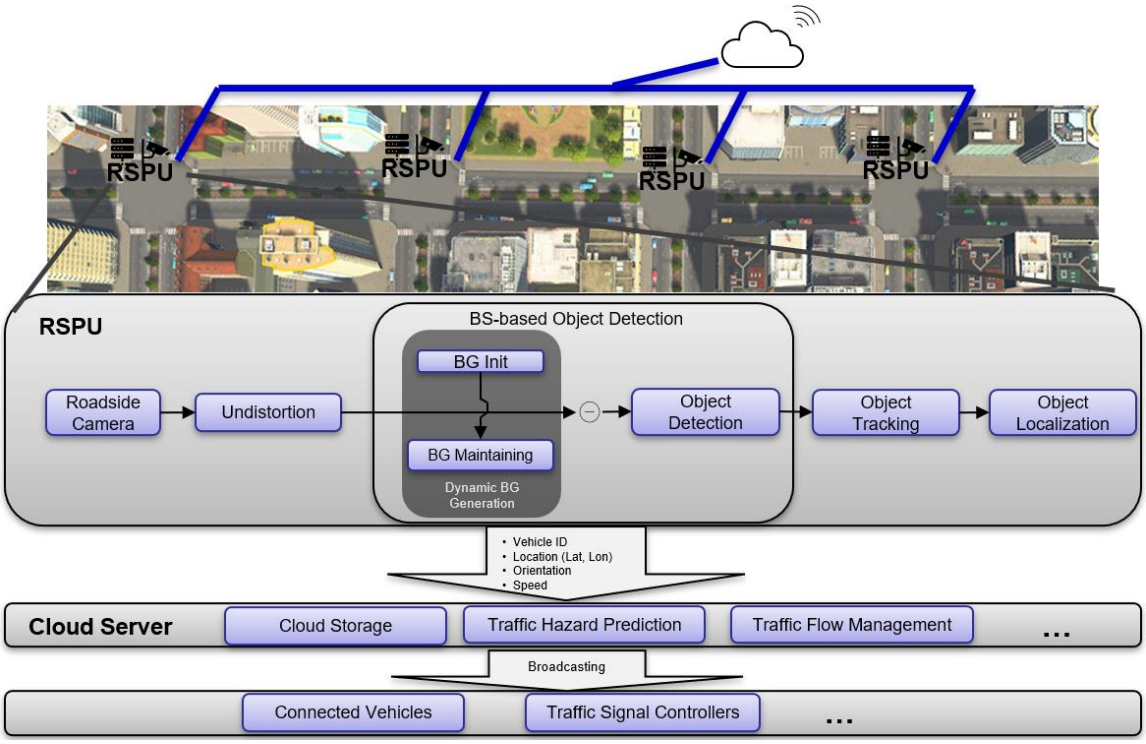


Figure 6-1 System architecture of the infrastructure-side sensing.

standards are established, such as Cooperative Awareness Message (CAM) [162], Basic Safety Message (BSM) [163], and Collective Perception Message (CPM) [164], [165], which allows the road users and infrastructures to receive and broadcast the detected objects.

LiDAR is an active sensor that generates the range and reflection point cloud, whereas a camera is a passive sensor that projects the 3D space into 2D planes with three color channels. As a result, it is easier for the camera to recognize the objects but harder to obtain precise range information. Because the roadside sensors often shoot to the ground and all the objects (e.g., cars, pedestrians) are assumed to move on the ground surface, the ranging impacts may be minimal. In comparison to LiDAR, however, the camera normally has significantly greater resolution. Therefore, in this study, we chose the camera as the sensor of our RSPU. Compared to the regular perspective camera, the fisheye camera or

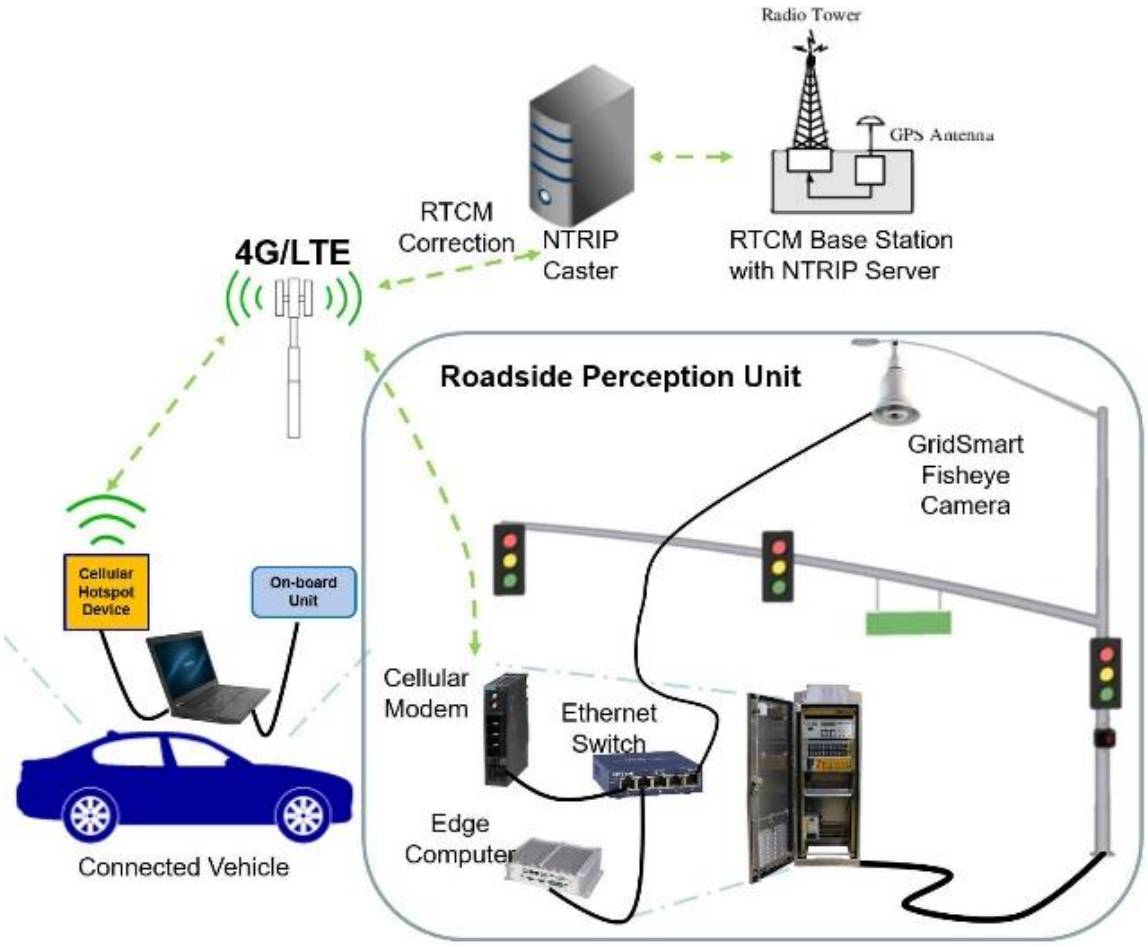


Figure 6-2 Roadside Perception Unit (RSPU) and communication topology.

omnidirectional cameras are becoming increasingly used in multiple areas such as traffic monitoring [71] and drone sensing [72] due to its wider detection view. Taking advantage of this feature, fewer cameras are required to cover the entire range of the target intersection. However, the images from the fisheye camera suffer from distortion and perspective effects, which necessitate additional processing. Furthermore, unlike the development of onboard detection algorithms, which already has a large amount of labeled data, the roadside camera dataset with positioning information, let alone fisheye camera dataset, are scarce. As a result, instead of deep learning-based approach, we have to rely on the image processing and conventional computer vision techniques to achieve the data-free real-time detection, localization and tracking tasks.

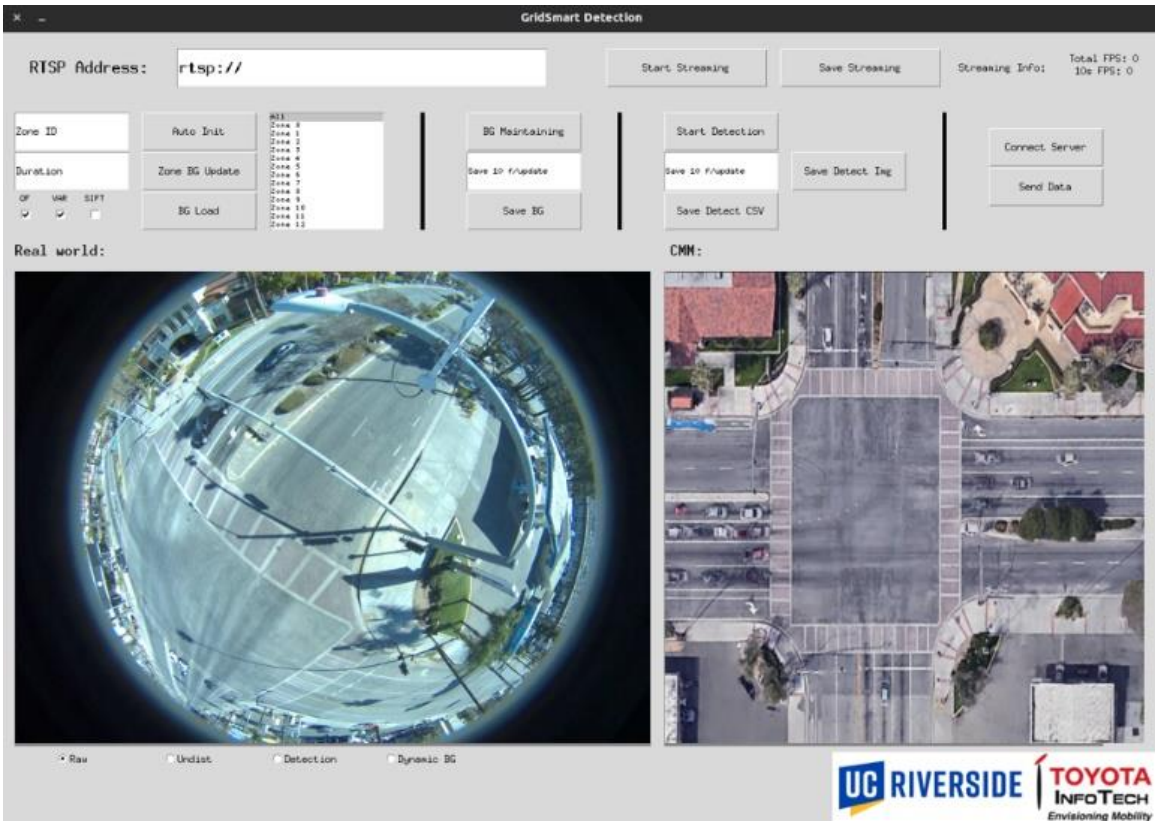


Figure 6-3 RSPU Graphic User Interface.

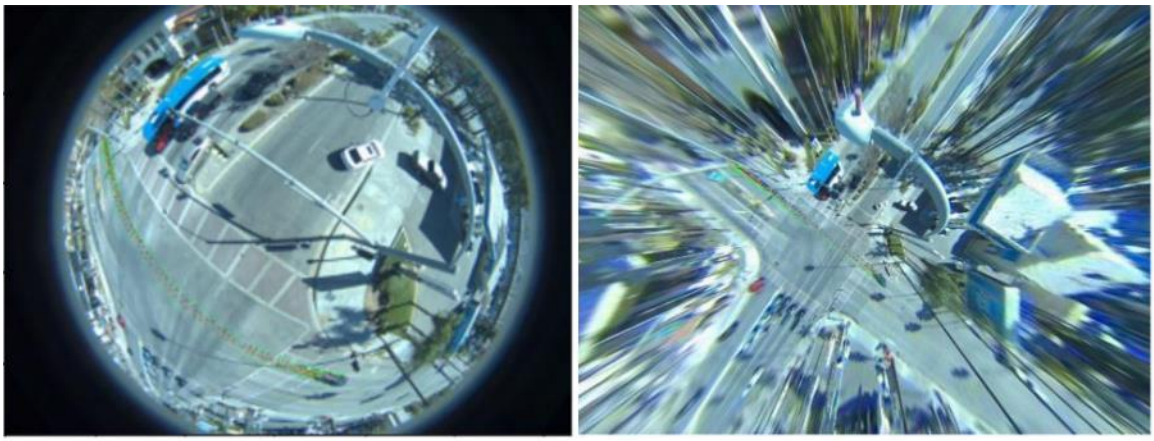
6.1.2 System Architecture

Figure 6-1 shows the overall architecture of the proposed real-time data-free perception system [73]. The RSPUs are deployed on top of each traffic intersection and send the detected vehicle trajectories back to the cloud server. The cloud server receives information, including consistent IDs, positions, orientations, and speed of the vehicles, and runs microservice applications such as traffic hazard prediction and traffic flow management. Finally, the agents, like connected vehicles and roadside infrastructures, equipped with communication modules, will be assisted in making decisions cooperatively for safety and mutually beneficial purposes.

The perception algorithm running on the RSPUs contains four major modules, including fisheye image undistortion, BS-based object detection, object tracking, and

object localization. First, the undistortion module recovers the perspective bird's-eye view (BEV) image based on the calibration parameters which are calculated using the auto-calibration method. Next, the undistorted video stream is fed into an innovative multi-layer BS-based object detector. The cascaded structure of the dynamic background generation module can synthesize real-time background with high frequency under all lighting conditions. It is also suitable for all moving objects at different speeds. Then, the DeepSort algorithm is implemented to associate IDs for all the detected objects. Finally, we calculate the chassis center of the detected vehicles based on the geometric relations and project the points to the world coordinate using the Haversine formulation. The details of the algorithms are introduced later in the Methodology section.

As shown in Figure 6-2, the Roadside Perception Unit includes a GRIDSMART Bell fisheye camera, an edge computer, and the communication devices such as a cellular modem and an ethernet switch located in the traffic control cabinet. The GRIDSMART Bell camera is mounted parallel to the road surface on top of the traffic pole with a 180-degree Field of View (FOV). It can only provide raw image data using Real-Time Streaming Protocol (RTSP) from a roadside processor unit (also located in the traffic control cabinet). Directly broadcasting the raw data through RTSP takes a significant amount of network bandwidth and usually has an unacceptable delay. Therefore, in our application, we use an edge computer to subscribe to the RTSP streaming locally in the RSPU and only broadcast the detection results via cellular communication. The Graphic User Interface (GUI) running on the edge computer is shown in Figure 6-3, where all the key modules of the RSPU are running with a multiprocessing scheme. The connected vehicle is equipped with a cellular module and an onboard PC to receive, process, and store



(a) (b)

Figure 6-4 Illustration of auto-calibration process. (a) Extracted trajectories in raw image. (b) Extracted trajectories after undistortion with estimated calibration parameters.

data. For system validation purposes, the connected vehicle also installed an RTK GPS unit (ublox F9R) as the ground truth.

The RSPU is implemented and tested in a real-world environment at the intersection of University Avenue and Iowa Avenue, as shown in the bottom right of Figure 6-3. The target intersection has 3 to 4 lanes for each direction and requires at least a 50 m by 50 m detection coverage. The fisheye camera is mounted at the southeast corner of the intersection, which can cover the whole intersection and detect incoming traffic from different approaches.

6.1.3 Methodology

Due to the significant distortion from edge to center, and the limited roadside camera dataset with positioning information, the existing methods based on the deep neural networks may not be applicable to fisheye images. This section introduces the design of the system and the detailed algorithms.

6.1.3.1 Fisheye Camera Calibration

To undistort the image and to build up the relationship between the image coordinates and the world coordinates for localization, camera calibration is necessary. Different from the ordinary camera that can use the pinhole model for calibration, three different models can be used to calibrate the fisheye camera, e.g., equisolidity projection, orthogonal projection, or stereographic projection. Although frameworks such as OpenCV provide functions to calibrate the camera readily, there is still some difficulty in obtaining correct calibration results in practice. Therefore, we go over the lesson and experiences from the field test in this subsection.

The typical way to calibrate a camera is to get a set of point pairs in both image space and 3D space and then use the Least Square to estimate the parameters based on the camera model, such as Zhang's method [166]. The easiest way to get the point pairs is to capture a chessboard image with known measurements from multiple perspectives. However, there are two key obstacles to the intersection monitoring application in practice. First, the fisheye camera is normally put on the top of a traffic signal pole, which can be up to 5 meters in height. It's difficult to get pictures of a chessboard of good quality. Second, because the road surface and detection target are so far away from the camera, even trivial calibration errors might result in meter-level position errors, especially for objects that are far from the fisheye camera's center. Another way is to first select sample points in the field, measure their locations in the world coordinates, and then fit the fisheye model with the corresponding points in the image plane. The 3D location of the sample points can be acquired using an RTK GPS device or well-calibrated LiDAR. However, it is necessary to

manually select the corresponding points in the image plane. As a result, the calibration accuracy is not acceptable for our application.

Inspired by the work by Jain et al. [167], we propose the auto-calibration method to get the camera parameters, including intrinsic, extrinsic, and distortion parameters. By assuming the camera is an equidistant fisheye, the following equation holds:

$$r_u = f \cdot \theta \tag{6-1}$$

where θ is the angle between a point in the real world and the optical axis; f is the focal length of the lens; r_u is the radial position of a point on the image plane. Also, we assume there are only radial effects for the camera distortion. The relationship between the points in undistorted and distorted coordinates can be depicted in a polynomial form:

$$\begin{aligned} x_u &= x_d \cdot (1 + k_1 \cdot r^2 + k_2 \cdot r^4 + k_3 * r^6) \\ y_u &= y_d \cdot (1 + k_1 \cdot r^2 + k_2 \cdot r^4 + k_3 * r^6) \end{aligned} \tag{6-2}$$

$$r_d^2 \to r^2 = x_d^2 + y_d^2 \tag{6-3}$$

where (x_u, y_u) and (x_d, y_d) are the corresponding points in undistorted and distorted coordinates, and r_d is the radius in the distorted plane for point (x_d, y_d). Then, the map between undistorted and distorted images using the equidistant fisheye assumption is given as follows:

$$(x_u, y_u) \to \left(x_d \cdot \frac{\tan\left(\frac{r_d}{f}\right)}{\frac{r_d}{f}}, y_d \cdot \frac{\tan\left(\frac{r_d}{f}\right)}{\frac{r_d}{f}} \right) \tag{6-4}$$

$$(x_d, y_d) \rightarrow (x_u \cdot \frac{\text{atan}\left(\frac{r_u}{f}\right)}{\frac{r_d}{f}}, y_u \cdot \frac{\text{atan}\left(\frac{r_u}{f}\right)}{\frac{r_d}{f}})$$

At the intersection, many vehicles move in orthogonal directions along a straight line. However, due to the fisheye effect, these straight lines turn to be curved in the distorted image. Therefore, by adjusting the value of the focal length, it is expected that a curve can be recovered to a straight line with the minimum least square error. Once recovered the focal length, the distortion coefficients k_1, k_2, and k_3 are calculated with the mapping equation (6-2).

The trajectories of the vehicles are extracted by tracking the pixel movement using the optical flow tracker with SIFT feature points. Then, a filter is applied to select the points that travel long enough distances without any lane change or turning. The filtered trajectories are marked in red and green in Figure 6-4. The curved trajectories in (a) should be straight lines going across the intersection. After estimating the parameters using the proposed auto-calibration method, the reprojected undistortion image is shown as (b). As can be seen from the figure, the trajectories are successfully corrected into straight lines.

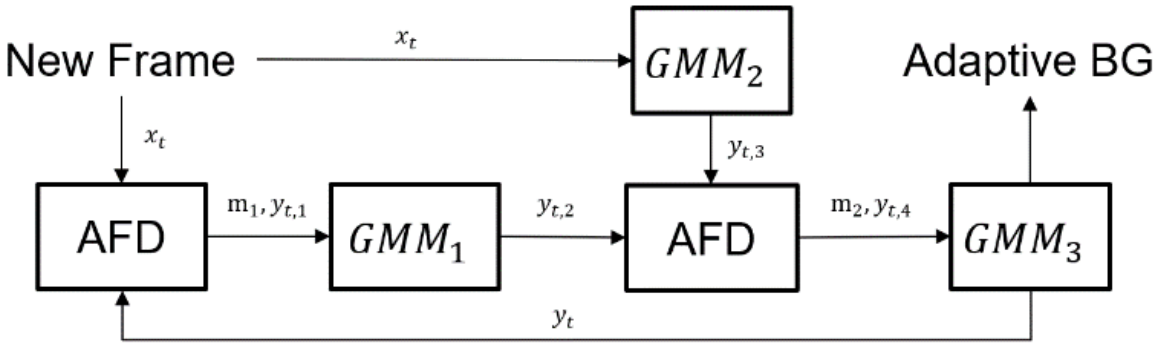


Figure 6-5 The cascaded structure of the background maintenance.

6.1.3.2 BS-Based Object Detection

One of the primary goals of this study is to design and deploy a perception method that is flexible and scalable for real-world applications. BS-based object detection is the most suitable method due to its data-free and robustness to different scenarios. However, all the existing BS algorithms are not capable of separating objects (e.g., vehicles) that keep stationary for a series of frames from the background, which is a commonly observed traffic scenario at a signalized intersection. In addition, to meet the requirement of 24/7 traffic surveillance, a dynamic BS strategy is needed to adapt to the changes in weather and lighting conditions. To solve the aforementioned problems, we propose an innovative cascaded adaptive background subtraction algorithm based on Adaptive Frame Difference (AFD) [60] and a modified Gaussian Mixture Model (GMM). This algorithm can dynamically update the regions without objects and reserve the regions with objects so that much fewer trials are left in the background. The previous BS algorithms aim at only separating the moving objects, while our method tries to remove objects no matter whether they are moving or staying still. Also, the feature-based methods, such as LBSP [168], LOBSTER, and SuBSENSE [169], fail to distinguish the foreground because of the edge distortion in the fisheye images, while the proposed one performs robustly for the fisheye images.

The proposed BS-based detection method has two major steps. The first step is background initialization, and the second step is background maintenance. In general, the task of both the background initialization module and the background maintenance module is the same, which is to generate dynamic background images. The background initialization module is more focused on background generation with a relatively short period of time and without initial input; the background maintenance module is more focused on high-quality background generation given a relatively reliable initial background image.

For background initialization, we define 13 different zones based on the moving direction of the traffic flow and process the incoming image frames separately. Because the GMM algorithm only works for the moving foreground, at each zone, optical flow-based movement detection is used to trigger the GMM module. Then, the GMM only updates the triggered frames. In this way, the slow-moving or stopped vehicles will not be recorded as the initial background. After all the zones are triggered, an initial background can be composited of the results from all the zones.

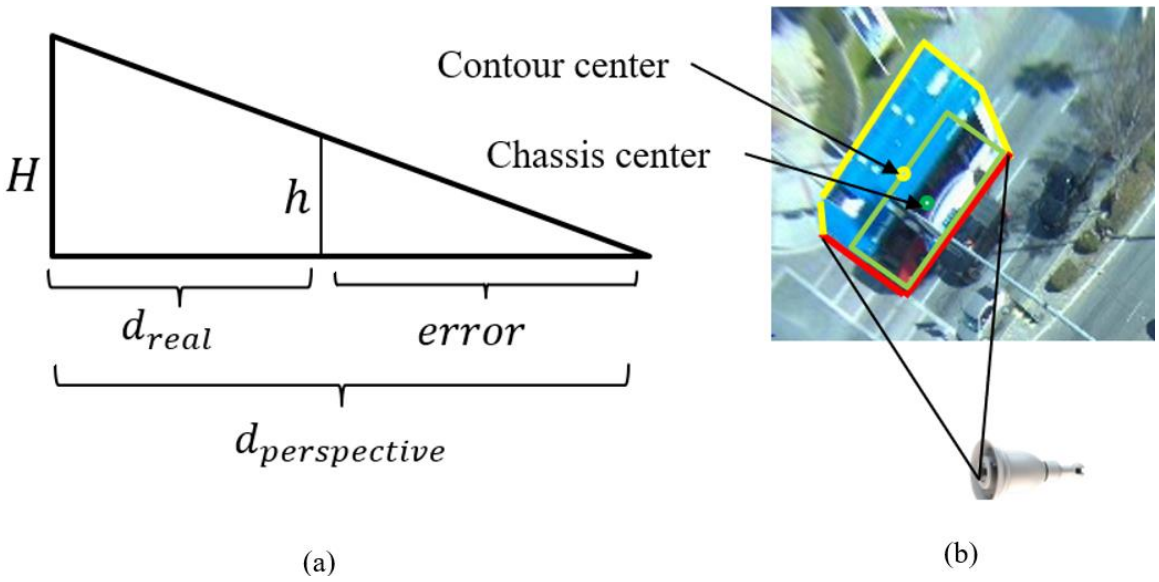


Figure 6-6 Chassis Center Estimation. (a) Projection error. (b) Illustration of contour center and chassis center.

The existing AFD methods only update the region without moving objects, which fails to remove static objects from the background. Therefore, we modify it to only update the region without an object. We compare the current frame I to the current background B and apply a threshold truncation to obtain a mask of the foreground M. The reverse of the mask M_{bg} is expected to be able to exclude the objects. The GMM modules try to model the distribution of pixels inside the image with N Gaussians, and each of them accounts for different image features and textures.

The workflow of the cascaded structure shows in Figure 6-5. We combine the ADF and GMM methods into a cascaded structure to overcome the shortcomings of each individual method. The ADF accounts for choosing updated regions, and GMM is responsible for updating the regions. More specifically, the ADF module is used to obtain the background mask with fewer foreground trails and the primitive low-quality background $y_{t,1}$. Then, the first GMM model GMM_1 is applied in the second module, which takes the mask and defective background as the input and generates a relatively clean

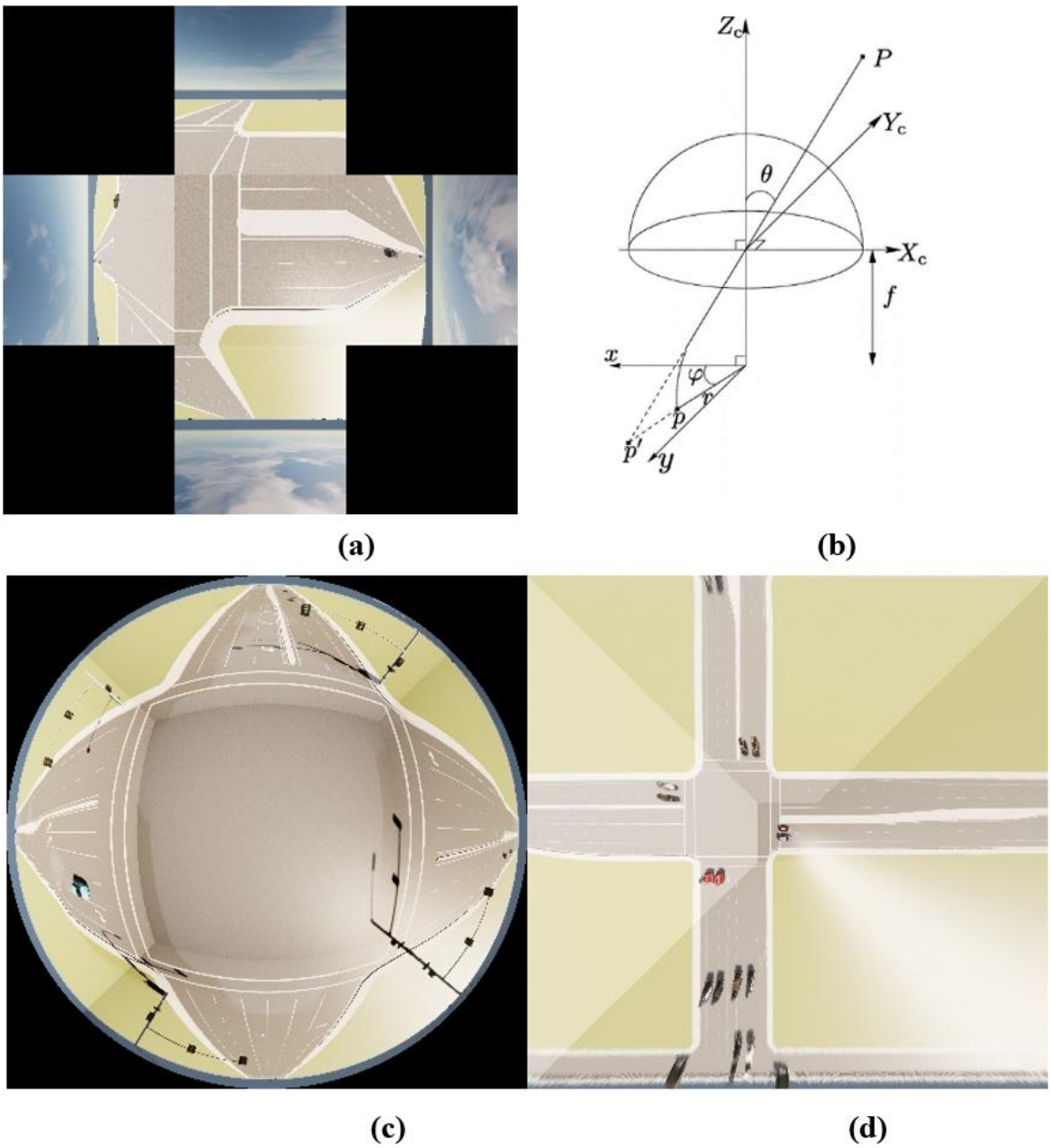


Figure 6-7 Illustration of virtual fisheye image synthesis. (a) The stitched cubic camera image with 90-degree of FOV from each perspective. (b) The projection relation from hemisphere to the image plane synthesized fisheye image. (c) Raw fisheye image. (d) Undistorted image using real-world calibration parameters.

background $y_{t,2}$. The third module is another AFD module, accepting an output $y_{t,3}$from the second GMM model GMM_2 as the "background" to obtain the mask. GMM_2 keeps being directly updated so that its result has the same global color distribution as the ground truth, and some mild blurs of shadow and vehicle are acceptable for this module. The trail

region from $y_{t,1}$ is identified and updated with the background $y_{t,2}$ from GMM_1 by AFD. $y_{t,3}$ can bring more robust information to the background in the color space while keeping the detail in $y_{t,2}$. The output $y_{t,4}$ is fed into the third GMM model GMM_3 and the final background y_t can be obtained.

After adaptively generating the dynamic background, contours representing the detected objects can be calculated by subtracting the current frame from the background. Sometimes, the adjacent vehicles have conglutination which leads to bad detection. To further purify the detection result, post-processing is needed. First, the morphology transform is used to remove the burr, glitch, and some slight conglutination. Concave point detection is then used to eliminate severe conglutination. If a contour has more than one concave point and its size and shape satisfy the predefined threshold, the contour is identified as a conglutinated one. Finally, we connect the two most concave points so that the contour is cut into two sub-contours.

6.1.3.3 Object Tracking

The object tracker is developed based on the DeepSort algorithm, which is one of the most popular real-time object tracking methods. As the algorithm requires the bounding box input of the detected objects, we calculate the minimum bounding rectangle of the contours. As a result, the tracking module assigns consistent IDs for all the given bounding boxes.

6.1.3.4 Localization

Due to the camera distortion and projection error, the speed and orientation of the vehicles cannot be calculated accurately only using the detected position information.

Therefore, we use the average contour optical flow as the representation of the movement of the vehicles. Instead of only tracking the optical flow of the feature points (e.g., SIFT or Shi-Tomasi corner), which leads to unstable results, a dense optical flow calculation is needed. However, the dense optical flow calculation is computationally heavy and does not fit real-time applications. We downsample raw images to accelerate this process without sacrificing accuracy.

The object detection is given using the contour representation as illustrated in the previous subsection. Then, the contour center can be easily estimated by calculating the center of the minimum bounding rectangle of the contour. Ideally, if the vehicle is right under the camera, the contour center point (the yellow point in Figure 6-6 (b)) aligns well with the chassis center point (the green point in Figure 6-6 (b)). However, due to the camera projection effect, we cannot directly use the contour center as the position of the vehicle. As shown in Figure 6-6 (a), the positioning error of a point with height h is given by $\frac{d_{perspective} \times h}{H}$, where H is the height of the camera. Assuming a vehicle is a cube, a BEV camera can see around half of the vehicle outline pixels on the ground, which does not have any positioning error ($h = 0$). Therefore, the overall positioning error of a vehicle can be approximated by equation (6-5).

$$Positioning\ Error = \frac{d_{perspective} \times h}{2H} \tag{6-5}$$

The average projection height of a vehicle h is approximated using equation (6-6).

$$h = \frac{h_{max} - h_{min}}{2} |\cos(\theta - \varphi)| + \frac{h_{max} + h_{min}}{2} \tag{6-6}$$

$$(\theta, d) = \text{card2pol(x, y)}$$

where θ is the position angle in the camera coordinate, and φ is the yaw angle of the vehicle calculated from the previous step. h_{max} and h_{min} are the maximum and minimum heights of a vehicle, respectively, which are the predefined parameters.

This step is to project the corrected center points to the world coordinate using the calibrated camera parameters and the Haversine formulation. Assuming that the road surface is a plane in the 3D space, we project the points from the image plane into the road surface plane in the camera coordinate. Then, the Haversine formulation (11) gives the projection between the great-circle distance of two points and their longitudes and latitudes.

$$d = 2r \cdot \arccos\left(\sqrt{sin^2\left(\frac{\varphi_2\text{-}\varphi_1}{2}\right) + cos\varphi_1 cos\varphi_2 sin^2\left(\frac{\lambda_2\text{-}\lambda_1}{2}\right)}\right) \quad (6\text{-}7)$$

6.1.4 Experiments

6.1.4.1 BS-Based Detection Algorithm Validation

Before implementing the proposed system in the simulation and real-world environments, we first validate the perception pipeline on CDnet 2014, an expanded change detection benchmark dataset, that is widely used to test BS-based algorithms. For the detection performance measurement, the confusion matrix is adopted as the metric. After applying the BS algorithms, the detected foreground (objects) is compared to the ground truth labels at the pixel level. The pixels of the overlapped region and non-overlapped region are obtained so that the confusion matrix of each frame is available. To

compare the proposed algorithm with five state-of-the-art BS-based algorithms, we employed the following metrics: precision, recall, False Positive Rate (FPR), False Negative Rate (FNR), Percentage of Wrong Classifications (PWC), and F-1 score.

Table 6-1 Dynamic background generation performance on CDNet2014

Method	Precision	Recall	FPR	FNR	PWC	F1
Our Method	74.85%	**82.35%**	1.78%	**1.13%**	2.73%	78.42%
GMM	83.03%	22.14%	**0.29%**	5.00%	4.97%	34.96%
ViBe	89.84%	53.16%	0.39%	3.01%	3.19%	66.79%
PBAS	85.79%	54.04%	0.57%	2.95%	3.31%	66.31%
LOBSTER	**91.64%**	69.51%	0.41%	1.96%	2.22%	79.05%
SuBSENSE	91.43%	79.18%	0.48%	1.34%	**1.70%**	**84.86%**

The results are presented in Table 6-1. Although our algorithm does not excel in pixel-level precision, the correctly detected pixels suffice to form complete bounding boxes, ensuring the accuracy of object detection. Furthermore, out algorithm's high recall rate guarantees no missed detections of objects. In comparison to the LOBSTER and SuBSENSE algorithms that have equivalent F1 scores, our algorithm requires significantly less computation. This advantage cannot be demonstrated in the CDnet 2014 dataset, as the image size is only 320 pixels by240 pixels. In actual traffic intersections, where smaller targets need to be detected, the image size will be larger, and LOBSTER and SuBSENSE algorithms may not support real-time detection. Additionally, it is worth noting that all the

objects in the CDnet 2014 dataset are all in motion, so the advantage of our algorithm in detecting stationary objects cannot be fully demonstrated.

Table 6-2 BS-Based detection performance on simulation environment

Method	Precision	Recall	FPR	FNR	PWC	F1	FPS
Our Method	67.28%	**77.99%**	0.61%	**0.35%**	**0.95%**	**72.24%**	12.60
GMM	57.59%	14.09%	**0.17%**	1.38%	1.52%	22.65%	25.66
ViBe	70.18%	44.25%	0.30%	0.90%	1.18%	54.27%	**55.81**
PBAS	57.65%	38.70%	0.46%	0.99%	1.42%	46.31%	10.78
LOBSTER	**71.91%**	43.30%	0.27%	0.91%	1.16%	54.05%	3.68
SuBSENSE	66.02%	76.96%	0.64%	0.37%	0.99%	71.07%	2.24

Table 6-3 Vehicle tracking performance on simulation environment

Method	Low volume		Medium volume		High volume	
	MOTP	MOTA	MOTP	MOTA	MOTP	MOTA
Our Method	**82.11%**	**9.94**	**78.29%**	**11.43**	**76.90%**	**11.23**
GMM	29.04%	14.49	33.02%	15.64	33.10%	15.63
ViBe	80.25%	10.49	72.74%	12.03	70.68%	12.94
PBAS	49.06%	12.96	53.70%	12.76	50.72%	12.41
LOBSTER	75.36%	10.68	73.36%	11.63	67.15%	12.23
SuBSENSE	70.53%	10.2	62.70%	13.16	59.88%	12.1

6.1.4.2 Simulation Experiment

We create a replicative environment same as the real-world target intersection using CARLA Simulator, which is an open-source simulator for autonomous driving research [170]. Using RoadRunner, an interactive editor for creating 3D scenes for modeling and testing autonomous driving systems [171], we manually draw the street map aligned with Google Earth according to the target intersection. Also, to simulate real traffic situations, the traffic signal phase and timing is set to match the one in the real world.

As CARLA does not provide any fisheye camera module, a Cubic-to-Fisheye algorithm is applied to synthesize virtual fisheye images out of five different regular perspective cameras. The focus points of the five cameras are located at the cubic center, and their optical axes are perpendicular to each other. To ensure that there is no overlap and no deficiency in the synthesized image, every camera has a 90-degree of FOV so that all the images have aligned edges (see Figure 6-7 (a)). After that, we project pixels from the cubic surface onto a hemisphere. The hemisphere is determined by connecting all pixels to the cubic center, and the intersection is the projection point. With the hemisphere image, a fisheye image can then be generated using the polynomial fitting, as also illustrated in Figure 6-7 (b). After obtaining the synthesized fisheye camera, as shown in Figure 6-7 (c), we calibrate this virtual camera using the same parameters as the real-world camera. As seen in Figure 6-7 (d), the fisheye image can cover 180 degrees of FOV, and the outside blue circular ring is the skyline in the simulation environment.

The object detection ground truth, or the bounding box, outlining the target object of interest (vehicles), can be obtained from the CARLA Application Programming Interface (API). By using inverse transformation, we could easily obtain the bounding box

in both the fisheye image and the undistorted image. Since BS-based methods typically suffer from high-density object situations, to demonstrate the effectiveness of our method, we simulated three different traffic volumes by controlling the number of vehicles in the intersection network: low volume, medium volume, and high volume. The low volume situation had 20 randomly spawned vehicles in the network, the medium volume had 42 vehicles, and the high volume had 64 vehicles. We recorded 9000 frames for each vehicle volume to create the datasets. All vehicles moved with predefined vehicle autopilot mode. The synthesized dataset is generated at a frame rate of 10 with a dimension of 960 pixels by 960 pixels covering a 140 m by 140 m area.

Our performance measurement is divided into two parts. For object detection, similar to the measurement on the CDnet 2014 dataset, we evaluate the pixel-level foreground (object) and background classification (see Table 6-2.). We compare the proposed algorithm with the other state-of-the-art BS-based detection algorithms such as GMM, ViBe, PBAS, LOBSTER, as well as SuBSENSE. For object tracking, Multiple Object Tracking Precision (MOTP) and Multiple Object Tracking Accuracy (MOTA) is adopted as the metrics (see Table 6-3).

As presented in Table 6-2, our method outperforms other state-of-the-art BS-based algorithms in most metrics while maintaining a high frame rate. This can be attributed to its superior performance in detecting stationary objects. Compared to LOBSTER and SuBSENSE, our method exhibits better FPR and FNR. This could be due to the fact that feature-based methods are susceptible to distortion from fisheye cameras, leading to some foreground in the edge regions being misclassified as background. Although the average precision is not very high at the pixel level, it is sufficient for accurate detection and

localization applications. Moreover, our method can process incoming video in real-time, and GMM, ViBe, and PBAS can also meet real-time requirements. However, LOBSTER and SuBSENSE take four to five times longer to process the same video as our method.

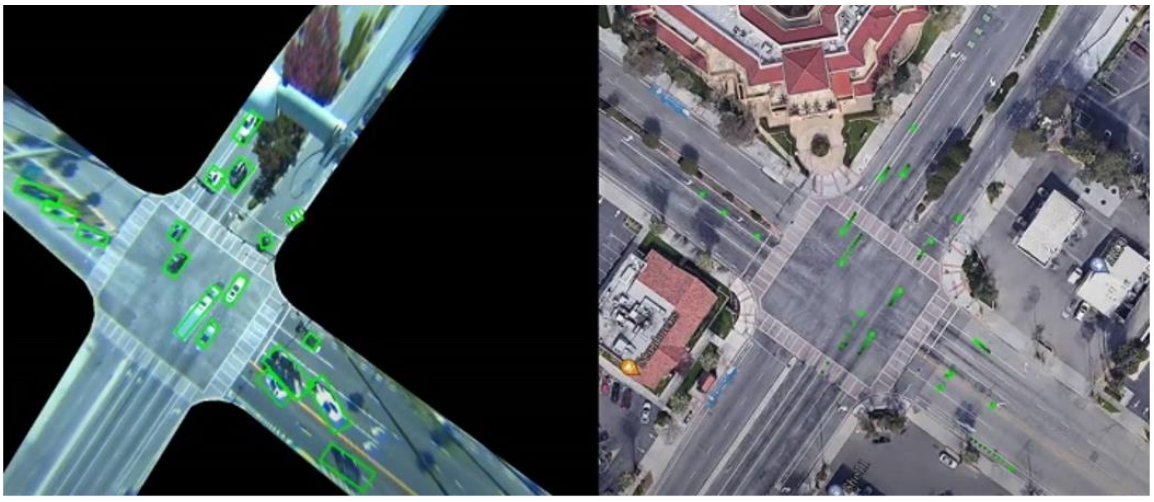
The tracking performance is shown in Table 6-3, where MOTA is measured in pixels. It is observed that as traffic volume increases, the performance of all algorithms, except for the GMM algorithm, decreases. However, our method achieves the best performance in all scenarios for all metrics. Also, because all other methods fail when vehicles stop at the intersection for red light, MOTP and MOTA only measure the frame when vehicles are moving. The MOTA of our method in the low-volume scenario is 9.94 pixels, equivalent to 1.45 meters, and in the high-volume scenario is 11.23 pixels, equivalent to 1.63 meters in the real world.

6.1.4.3 Real-World Experiment

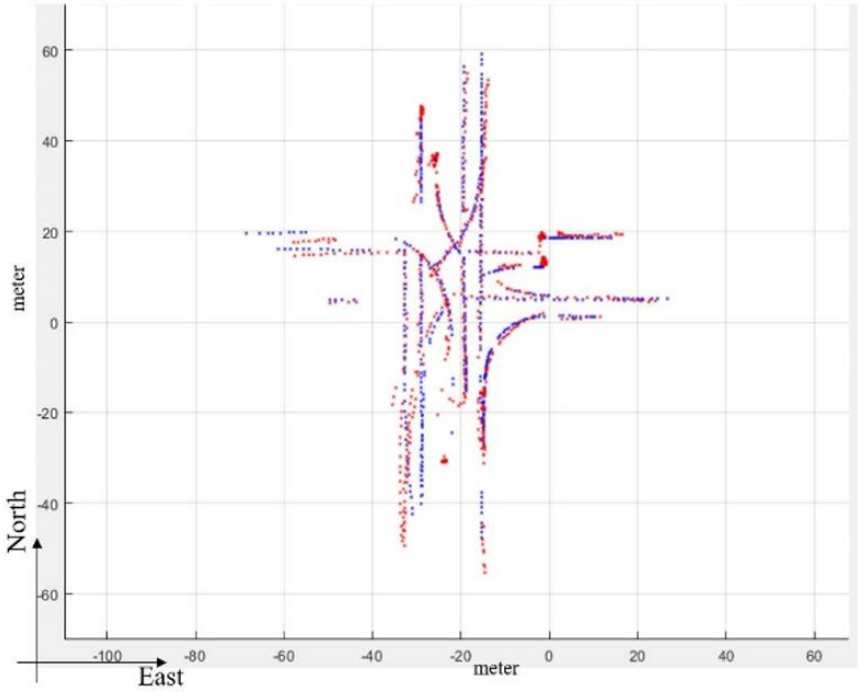
The RSPU is implemented and tested in a real-world environment at the intersection of University Avenue and Iowa Avenue, which has 3 to 4 lanes for each direction and requires at least a 50 m by 50 m detection coverage. The GRIDSMART bell fisheye camera is mounted at the northwest corner of the intersection, which can cover the whole intersection and detect incoming traffic from a different direction. The qualitative result of the detection shows in Figure 6-8, where Figure 6-8 (a) shows all the detected vehicles by the bounding boxes in a sample frame; Figure 6-8 (b) shows the detected trajectories after the tracking module.

As it is difficult to acquire the ground truth position of all the vehicles in public traffic, the quantitative performance is only measured for the test vehicle. The test vehicle is a connected vehicle equipped with Real-Time Kinematic (RTK) GPS device. The RTK

positioning is the application of surveying to correct for common errors in the Global Navigation Satellite System (GNSS), which uses reference stations to enhance the measurement via the phase of the signal's carrier wave. Based on our previous examination, the system can achieve centimeter-level accuracy at the target intersection. Therefore, in this research, we use it as the real-world ground truth. Figure 6-8 (c) shows the collected trajectory from both the RSPU detection and the RTK ground truth. As can be seen from the figure, the detection points (in red) align well with the ground truth (in blue). The quantitative results are shown in Table 6-4.



(a) (b)



(c)

Figure 6-8 Qualitative Results for Real-world experiment. (a) Detected bounding boxes. (b) Detected trajectories. (c) Trajectory of the ego vehicle, ground truth in blue and detection in red.

It should be noted that the average processing time of the RSPU is 90ms, and the average communication delay from the RSPU to the connected vehicle is 160ms. Compared to the MOTA in the simulation, all the following accuracy measurements are considered with such a large delay. The most accurate detection is achieved within the

range of 10 meters. Due to the radial distortion, it is expected the detection error increasing with the increase of the detection range. However, this does not always hold, as can be seen in Table 6-4. This is because of the communication delay and the feature of the intersection. For example, the average speed from the range of 30 to 40 meters is much higher than 20 to 30 meters and 40 to 50 meters, as there is no stop line within the range. As a result, the accuracy in the range of 30 to 40 meters is lower than the others.

Table 6-4 Quantitative results of real-world experiment

Range (m)	Overall	0~10	10~20	20~30	30~40	40~50	50~
Accuracy (m)	2.28	1.22	2.11	2.85	3.02	2.16	2.31

6.1.5 Summary

This section introduces an implementation of a real-time roadside perception unit using fisheye cameras, which detects and broadcasts vehicles' movement to the connected devices. The perception pipeline consists of four major modules, including fisheye image undistortion, background subtraction-based object detection, object tracking, and object localization. For the roadside fisheye camera undistortion, we use an auto-calibration technique, and for detection, we have used an innovative cascaded structure BS algorithm. Our approach outperforms five existing BS-based detection algorithms in terms of F1 score and computation time when tested on CDnet 2014 benchmark.

The system is implemented in the RSPUs in both simulation environments and real-world environments. The CARLA-based simulation environment is a replica of the real-

world environment, where we use the Cubic-to-Fisheye algorithm to synthesize the virtual fisheye camera with the same parameters as the real-world one. The results show that the proposed algorithm outperforms the state-of-art BS-based detection algorithms without compromising the computation efficiency. To our best knowledge, the proposed algorithm is the only one that is robust to different speeds of the objects and is readily implemented in real-time. In the real-world environment, we deployed the RSPU in a target intersection with 3 to 4 lanes in each direction. We use a connected vehicle equipped with an RTK GPS device to receive the detection broadcast and record the ground truth. The real-world experiment shows that our roadside perception system can achieve lane-level position accuracy for the whole target intersection.

Multiple future steps are mapped out and discussed as follows. First, it is planned to broadcast the detection in the form of V2X standards, such as CAM, to improve the system's compatibility. Second, it is expected to have a better performance to use our method as an auto-labeling approach to providing ground truth data (maybe a bit noisy) for training the ANN-based model. Also, as the proposed system is flexible and scalable, we plan to deploy the system along a corridor to achieve a larger-scale roadside perception. Finally, it is intended to develop a corporative perception system fusing information from other roadside sensors (e.g., LiDAR) or onboard sensors to improve situation awareness.

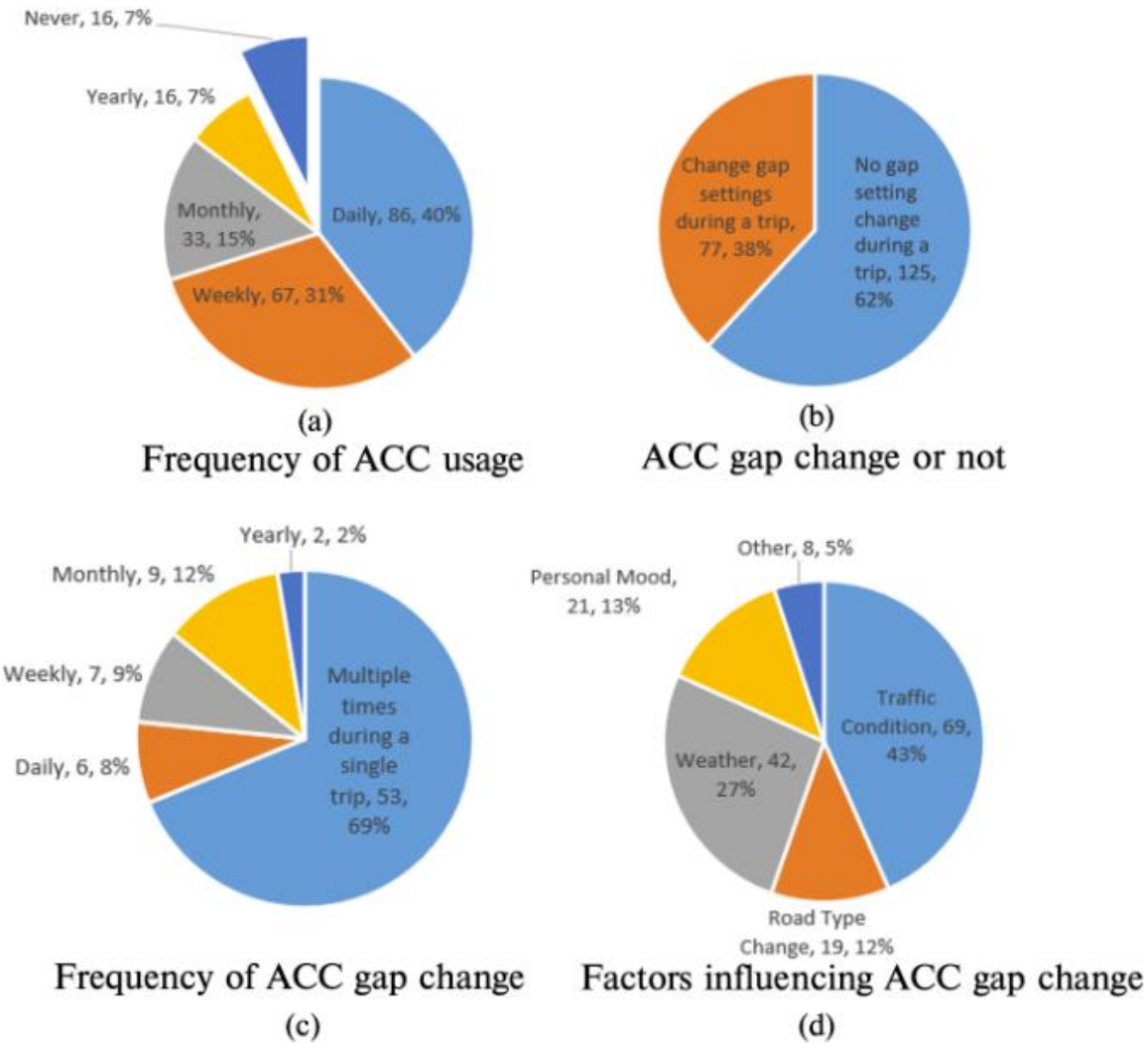


Figure 6-9 Result of customer survey regarding ACC usage.

6.2 Driver Behavior Modeling

6.2.1 Introduction and Background

Adaptive Cruise Control (ACC), as one of the most widely researched advanced driver-assistance systems (ADAS), has been deployed on more and more vehicles over the last decade [172]–[174]. The function of the ACC system is to control the ego vehicle's longitudinal speed so that it follows the leader with a user-selected speed or adaptively maintains a constant time headway. The ACC system has been demonstrated to improve

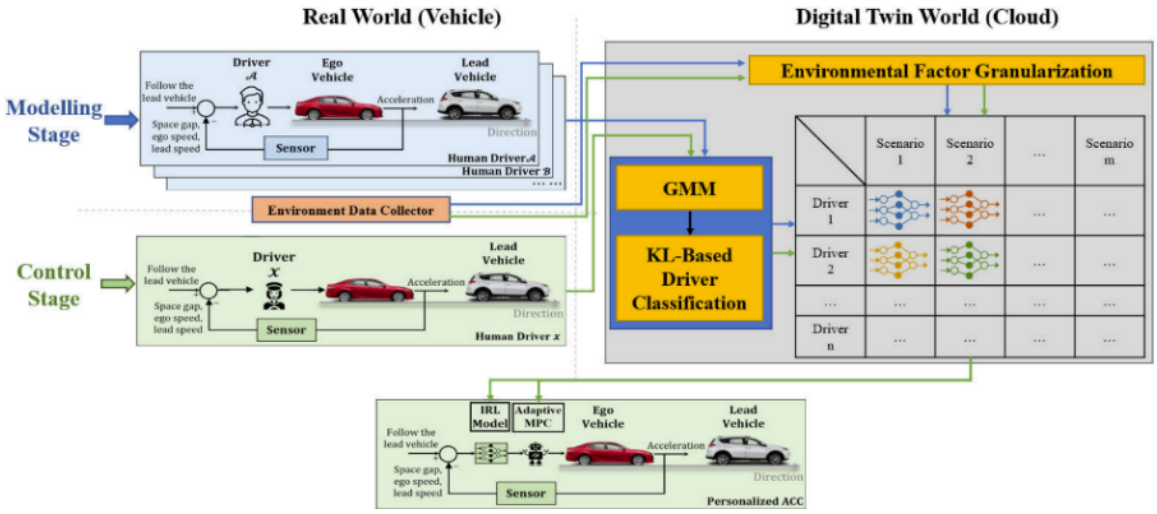


Figure 6-10 System architecture: Digital twin framework.

traffic-wise mobility, safety, and environmental sustainability [175]. Nevertheless, the existing ACC is normally an expert system where the control scheme is based on well-tuned look-up tables, and only a few settings are available to the users. Those limited settings (e.g., long gap, medium gap, and short gap) may not always align well with the drivers' preferences due to their aggressiveness or conservativeness. The diverse car-following strategy may lead to an uncomfortable experience and reduce the trust in driving automation. As a result, the driver may adjust the setting frequently or even stop using it.

To validate our observation, a survey targeting at ACC users has been conducted in January 2022, which covers 13 vehicle brands on the U.S. market that carry the ACC function (i.e., Ford, Honda, Hyundai, Jeep, KIA, Lexus, Lincoln, Mazda, Subaru, Tesla, Toyota, Volkswagen, and Volvo). The survey's sample size is 275 people, with 3.0% being between the ages of 18 and 24, 30.8% being between the ages of 25 and 34, 30.1% being between the ages of 35 and 44, 24.% being between the ages of 45 and 54, 10.1% being between the ages of 55 and 64, and 1.5% being older than 65. As presented in Figure 6-9 (a), majority of the respondents use ACC on their vehicles. Among these ACC users, 38%

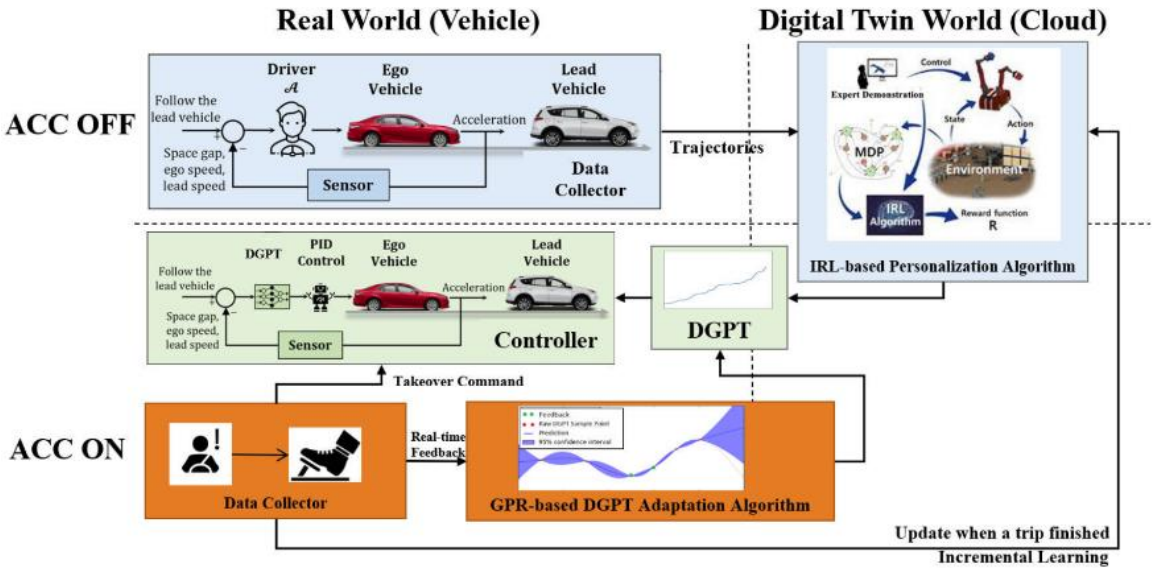


Figure 6-11 System architecture: Offline learning (blue blocks), Online learning (orange blocks), Personalized controller (green blocks).

of them like to change the ACC gap setting during a trip, as shown in Figure 6-9 (b). Figure 6-9 (c) illustrates a more important result: Among these drivers who like to change ACC gap setting, 69% of them change it multiple times during a single trip, which indicates that the current ACC setting needs more improvement for driver personalization. In addition, Figure 6-9 (d) shows the major factors influencing the driver to change the ACC setting include traffic condition, weather, personal mood, and road type.

To improve the current expert system-based ACC, a more advanced ACC system that takes personalized behavior into consideration is imperative. Referring to [176], personalized driving style and preference can significantly impact drivers' acceptance of driving automation. However, in the automobile industry, the personalization functions are still restricted to a complimentary level, such as seat ergonomics and personalized infotainment systems.

In this study [177], we present a novel P-ACC framework that combines both offline and online learning. The offline learning is achieved through the Inverse

Reinforcement Learning (IRL) algorithm, which is trained from the naturalistic car-following trajectories of individual drivers to infer their driving style and create a driving gap preference table (DGPT) that is used as a reference for the P-ACC system. Unlike other P-ACC methods that simply clone the driver's behavior from the historical data, our offline model takes into account the driver's task-specific preferences. The online learning component adapts the DGPT in real-time based on the driver's feedback through ACC overrides. This is achieved using Gaussian Process Regression (GPR) [178], which is a statistical method that can be used for model-free prediction and estimation tasks. The driver's feedback data is also used to update the offline personalization module as an incremental learning scheme. Our proposed P-ACC system provides a personalized driving experience for each driver, resulting in enhanced driving comfort and safety.

6.2.2 Problem Formulation and System Architecture

6.2.2.1 Digital Twin Framework

This subsection presents the digital twin framework of the P-ACC system. The flowchart of the proposed P-ACC system is presented in Figure 6-10. The system has two stages: *modeling stage* and *control stage*. Both stages have their corresponding tasks in the real world (deployed on the real vehicle) and the Digital Twin world (deployed on the cloud). In this paper, the proposed P-ACC system is a subset of the Mobility Digital Twin framework that was previously developed by authors of this paper [179].

Modeling Stage: The modeling stage happens when the driver operates the vehicle manually. Based on a rule-based logic, all the car-following events can be filtered out and transmitted to the Digital Twin cloud together with the environment information (e.g., weather conditions, road types, etc). The car-following events, represented as sequential

trajectory data, will be approximated using GMM on the Digital Twin cloud. The GMM distribution of a trajectory is regarded as the driver's driving style. Then, the new GMM is compared with the existing GMM distributions using the KL divergence metric. If it is close enough to one of the existing distributions, the trajectory data would be merged into the corresponding class. Otherwise, a new driver type will be created. As environmental factors also impact drivers' behaviors, different models are trained for different scenarios. For each scenario and each driver type, an IRL model will be trained to represent the driver's driving style.

Control stage: The control stage happens when the driver activates the P-ACC driving automation. Before the system is activated, a short-term observation of the driver is updated to the Digital Twin cloud together with the current environment information. The short-term trajectory is also approximated using GMM and driver type will be determined by finding the maximum KL divergence of the existing distributions. Based on this driver type and scenario ID, the corresponding pre-trained model will be downloaded to the target vehicle. Next, multiple trajectories of the ego vehicle are generated to replicate the stored short-term trajectory, given the speed profile of the leader, using MPC with parameters in a predefined pool. The parameters that reproduce the closest trajectory are selected for the P-ACC controller. It should be noted that if the driver activates the P-ACC system recently, there is no need to re-classify the driver type or to conduct the MPC parameters adaptation again. Based on the current scenario, the corresponding model can be downloaded directly.

6.2.2.2 P-ACC With Online Adaptation

Figure 6-11 shows the general system architecture of the proposed P-ACC framework. Different from Figure 6-10, where the personalization model only relied on the historical demonstration car-following trajectory (blue blocks in Figure 6-11), we introduce a novel approach to incorporate the driver's real-time feedback on the ACC system as a dynamic input to adjust the model in this work (orange blocks in Figure 6-11). Based on our literature review, no previous studies have considered the driver's real-time feedback on the ACC system. The physical layer of the framework is divided into the real world (vehicle) and the digital twin world (cloud), while the implementation process is divided into two phases: ACC OFF and ACC ON.

At the ACC OFF phase, when the driver manually follows the lead vehicle, the system considers the trajectory as an expert demonstration and transmits it to the cloud along with environmental factors that could potentially impact driving behavior. On the cloud, the IRL algorithm assumes that the collected expert demonstration is near-optimal in terms of the Markov Decision Process (MDP) and infers the reward function that drives the driver's behavior. This reward function is then transferred to the DGPT as the control reference.

At the ACC ON phase, when the driver turns on the automatic following mode, the personalized driving model (i.e., DGPT) is downloaded locally. The DGPT is a discrete table that records the relationship between the preferred following distance and its corresponding speeds. We employ GPR to represent the DGPT, allowing for continuous outputs that can be updated as new data points (driver's feedback) come. The DGPT is designed to describe the driver's preferred following distance at different speeds. The

controller (green blocks in Figure 6-11) maintains the distance to the lead vehicle. However, due to differences between the scenario of demonstration trajectories and the current driving scenario, as well as changes in the driver's driving habits or mood, the driver may not always be satisfied with the current automated control. Therefore, the driver can provide feedback to the system by pushing the accelerator to shorten the car-following gap or brake pedals to lengthen the gap, leading to a takeover of the vehicle. These takeover segments are used to adjust the DGPT in real-time, responding to the driver's behavior. What's more, when the current ACC ON trip is completed, these takeover segments are sent back to the cloud to fine-tune the IRL model, improving the personalization of the system.

6.2.2.3 Assumptions and Specifications

In this study, we focus on modeling and controlling personalized car-following maneuvers based on states of the ego vehicle and its preceding vehicle. Although the real-world ACC needs to manage both car-following stage and free-flow stage, the ACCs discussed in this section only include the car-following stage, i.e., we assume the leader of the ego vehicle is always present. Also, only the longitudinal movement is observed and controlled. The goal of this work is to design speed control strategy for the ego vehicle to satisfy the driver's preference. The remainder of this section specifies the mathematical formulation of the proposed system.

$$x = \begin{pmatrix} p \\ v \\ g \end{pmatrix} \tag{6-8}$$

$$\dot{x} = \begin{pmatrix} \dot{p} \\ \dot{v} \\ \dot{g} \end{pmatrix} = \begin{pmatrix} v \\ a \\ vf - v \end{pmatrix} = A \begin{pmatrix} p \\ v \\ g \end{pmatrix} + B \cdot a + C \cdot v_f \tag{6-9}$$

where p and v are the position and speed of the ego vehicle; g is the distance gap between the ego vehicle and preceding vehicle; a is the acceleration, which is the only control input of the dynamic system; vf is the speed of the preceding vehicle; A, B, and C are linear coefficient matrices. As the speed of the preceding vehicle is determined by the driving style of the lead driver and downstream traffic, vf is regarded as an uncontrollable disturbance to the system.

Because we define the static car-following preference as a function of the speed and desired gap, we focus on the dynamic in the speed-gap space, and the corresponding variables are updated in a discrete form:

$$v[t+1] - v[t] + a[t] \cdot \Delta t + \sigma_w \tag{6-10}$$

$$g[t+1] = g[t] + \left(v_f[t] - v[t]\right) \cdot \Delta t + \frac{1}{2} + a[t] \cdot \Delta t^2 + \sigma_g \tag{6-11}$$

As shown in equation (6-10) and (6-11) we add Gaussian noises, σ_w and σ_g, denoting imperfectness of the driver's observation.

It is assumed that the input of the system is acceleration, and the driver controls vehicle's speed based on his/her observation of the surroundings. We presume the driver's decision-making process is a Markov Decision Process (MDP) defined by a five-tuple $\{S, U, T, r, \gamma\}$where S is the state space spanned by v and g; U is the one-dimensional action space of all possible acceleration of the ego vehicle; T is the transition probability

determined based on equation (6-10) and (6-11); r is the reward function that represents the driver's personalized car-following style; and γ is the discount factor weighting the importance of the historical rewards; At each time step, the process in certain state v and g, and the driver may choose any action a. The process responds at the next time step by moving into a new state s' based on T. Notably, although the speed of the preceding vehicle, v_f, is considered in MDP, it can be observed while driving. It should be noted that we also assume the human driver is rational and his/her actions are optimizing a cumulative reward function formulated as follows:

$$v(\xi) = \sum_{t=0}^{N} \gamma^t \cdot r_t(s) = \sum_{t=0}^{N} \gamma^t \cdot \alpha^T \cdot \Phi(s) \tag{6-12}$$

where N denotes the time horizon, and ξ denotes the trajectory of the ego vehicle. As seen in equation (6-12), the instantaneous reward r_t is assumed to be expressed in a span of the reward basis Φ, whose dimension equals the total number of features, and α^T stands for a vector of weight defining the linear combination. Additionally, it is assumed that the collected trajectory can reflect the drivers' driving style and that drivers are comfortable with their own driving style.

6.2.3 Methodology

Most existing studies classify driver type via explanatory definitions. For example, Chiyomi et al. classified drivers into five categories including extremist, flow conformist, planner, hunter/tailgater, and ultraconservative [180] based on drivers' levels of aggressiveness/conservativeness. However, this assumption oversimplifies the logic behind drivers' behaviors. For example, a newly licensed driver may present aggressive

behaviors due to lack of skills. Therefore, in this section, rather than defining driving behaviors with semantic meaning, we cluster them purely based on their car-following trajectory distributions. Specifically, all the trajectories of the car-following events are first approximated using GMM. Then, the GMM distribution of a new trajectory compares with the distributions of existing types of drivers on the cloud using the KL divergence metric as shown in the following equation (6-13}). KL divergence, also referred to relative entropy, is a metric for divergent one probability distribution is from another.

$$\begin{aligned} D_{KL}(P \parallel Q) &= \sum_{x \in X} P(x) \cdot \log \left(\frac{P(x)}{Q(x)}\right) \\ &= \sum_{x \in X} P(x) \cdot \log \left(P(x)\right) \\ &- \sum_{x \in X} P(x) \cdot \log \left(Q(x)\right) \end{aligned} \tag{6-13}$$

$$D_{KL}(P, Q) = \frac{1}{2}\left(D_{KL}(P \parallel Q) + D_{KL}(Q \parallel P)\right) \tag{6-14}$$

where x is the random sample in the trajectory space X; P(.) and Q(.) are the probability of a sample under the trajectory GMM distribution P and existing driver type GMM distribution Q, respectively. $\sum_{x \in X} P(x) \cdot \log \left(P(x)\right)$ is the sample mean of weighted log probabilities. Because the KL divergence is not symmetric, we take the average of both the KL divergence from P to Q and the one from Q to P as the metric of distribution distance as shown in equation (6-14). Based on the KL distance, a rule-based classifier is defined in ***Algorithm 6-1***. It should be noted that the number of gaussian distributions for GMM approximation and the KL distance threshold are two predefined parameters chosen based on empirical experience. The number of Gaussians keeps the balance between overfitting

and underfitting for trajectory distribution approximation; the KL distance threshold determines the resolution of the driver type classification.

Algorithm 6-1*: Driver Type Classification*

Data: New driver's trajectory: $\boldsymbol{\xi} = \{\xi_0, \xi_1, \dots, \xi_n\}$, GMM distributions of existing driver types: $\boldsymbol{Q} = \{Q_0, Q_1, \dots, Q_m\}$, number of Gaussian: k, KL distance threshold: θ

Result: Updated GMM distributions of driver types: Q', Driver type of the new driver: DT_index

```
Approximate the distribution of ξ, Q_new, using k-GMM
min_KL_distance = Inf
min_KL_distance_index = -1
if ξ is empty then
    add Q_new into Q
    min_KL_distance_index = 0
end
else
    for Q_i in Q do
        Calculate current_KL_distance
        if current_KL_distance < min_KL_distance then
            min_KL_distance = current_KL_distance
            min_KL_distance_index = i
        end
    end
```

16: **if** current_KL_distance > θ then

17: add Q_{new} into Q

18: min_KL_distance_index = i+1

19: **end**

20: **else**

21: Update $Q_{\text{min\_KL\_distance\_index}}$

22: **end**

23: **end**

24: **Q'=Q**

25: **DT_index =** min_KL_distance_index

Based on observations of both simulation and naturalistic data, even the same driver can present completely different driving styles in different scenarios. For example, an experienced driver with poor vision can be aggressive with good lighting conditions (e.g., daytime), while being very conservative with bad lighting conditions (e.g., evening). Moreover, the transition from one mode to another can be subtle and discrete. In addition, according to the customer survey, the factors influencing the driver to change the ACC setting include traffic condition, weather, personal mood, and road type. Therefore, in this section, we train different car-following models for different scenarios of different environmental factors. Specifically, vehicle type, road type, weather conditions (i.e., precipitation, visibility, clouds), and time-of-day (i.e., daytime, nighttime) are the factors to consider. Using the **OpenWeatherMap API** [181], the weather information can be collected given the position and time. All the environmental factors are then granularized

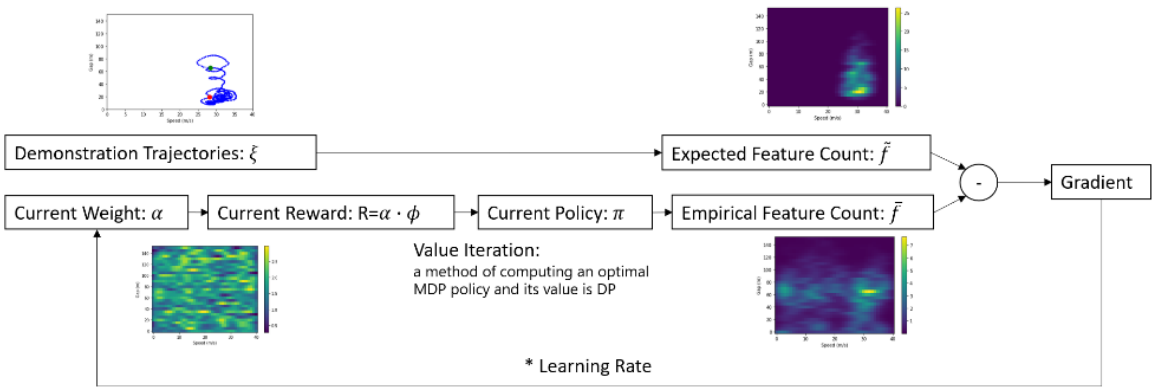


Figure 6-12 The training process of the maximum entropy IRL (Max-Ent IRL) used for static car-following preference modeling.

into subcategories. If new driving data is collected under the corresponding subcategory, the car-following model will be updated.

The discrete 2-dimensional speed-gap state space is defined based on their range for normal car-following tasks, where v extents from 0 to 36m/s with the interval of 0.5m/s, and g extents from 0 to 120m with the interval of 0.5m. As discussed previously, it is assumed that the driving behavior, namely, the determination of the control input (acceleration), is based on optimizing a cumulative reward function defined by equation (6-12). The reward function is a linear combination of N-dimensional reward basis Φ. As recommended by [93], we define a Gaussian kernel in the reward basis to further improve the nonlinear representation ability for approximating the reward function. Each entry, Φ_i, in the reward basis vector Φ corresponds to a kernel function of a state s and the other state s_i, in the discrete state space, as shown in equation (6-15).

$$\Phi(s) = \begin{bmatrix} \Phi_1(s) \\ \Phi_2(s) \\ \cdots \\ \Phi_n(s) \end{bmatrix} = \begin{bmatrix} \mathrm{K}(s, s_1) \\ \mathrm{K}(s, s_2) \\ \cdots \\ \mathrm{K}(s, s_n) \end{bmatrix} \tag{6-15}$$

where K is the Gaussian-like kernel, and n is the total number of discrete states. The kernel function measures the relationship between s and s_i shown as follows:

$$\mathrm{K}(s, s_i) = \exp\left(-\frac{|s - s_i|^2}{\sigma^2}\right) \tag{6-16}$$

$$|s - s_i|^2 = (v - v_i)^2 + (g - g_i)^2 \tag{6-17}$$

σ controls the size of the kernel, which is chosen manually according to the resolution of the state space to achieve a balance between overfitting and underfitting.

Conforming to the maximum entropy IRL (Max-Ent IRL) algorithm presented by Ziebart et al. [92], the probability of a trajectory is proportional to the sum of the exponential rewards accumulated along with the trajectory as described in equation (6-18).

$$\begin{aligned} p(\xi \mid \alpha) &= \frac{1}{Z(\alpha)} \exp\left(\sum_t R_\alpha(s_t)\right) \\ &= \frac{1}{Z(\alpha)} \exp\left(\sum_t \alpha^T \mathbf{\Phi}(s_t)\right) \end{aligned} \tag{6-18}$$

where $Z(\alpha)$, named partition function, which equals to $\sum_\xi \exp\left(\sum_t R_\alpha(t)\right)$. The maximum log-likelihood approach is then used upon the demonstrations with respect to the weight of the reward function:

$$L(\alpha \mid \xi) = \max_\alpha \sum_\xi \log\, p(\xi \mid \alpha) \tag{6-19}$$

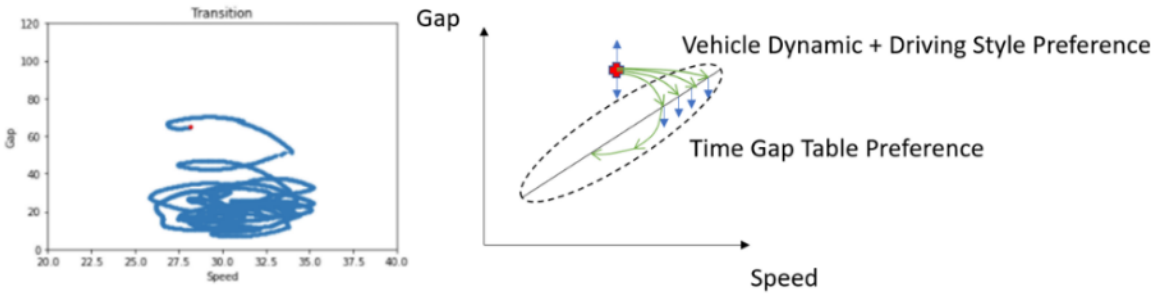


Figure 6-13 Illustration of drivers' dynamic preference.

Thereby, the gradient of the weight, α, can be derived in the following form:

$$\nabla_\alpha L = \sum_\xi p(\xi) \sum_{s\in\xi} \Phi(s) - \sum_\xi \mathrm{D}_s \sum_{s\in\xi} \boldsymbol{\Phi}(s) \tag{6-20}$$

The first term, $\sum_\xi p(\xi) \sum_{s\in\xi} \Phi(s) = \tilde{f}$, is called *expected feature count*. The second term, $\sum_\xi \mathrm{D}_s \sum_{s\in\xi} \boldsymbol{\Phi}(s) = \bar{f}$, is called *empirical feature count*, and D_s is the state visitation frequency.

The training process is demonstrated in Figure 6-13. Initially, the reward weight, α, is randomly generated, and the expected feature count, $\tilde{f}$, can be derived following the first term of the Equation (6-20), where $p(\xi)$ can be estimated by counting the state visitation frequency of the demonstration trajectories. Then, at each iteration, the value iteration algorithm [182] is applied to approximate the optimal policy $\pi^*(s,a)$ and the value function $v(s)$ based on current reward. Theoretically, the state visitation frequency, D_s can be derived by traversing the state space exhaustively given a predefined time horizon, the distribution of initial state, transition probabilities (car-following dynamic), and an optimal policy under the current reward function [92]. However, this is extremely computation heavy, especially when the state space is large and/or the time horizon is long. To solve this, we approximate $\mathrm{D}_s$via the Monte Carlo method. Based on the optimal policy

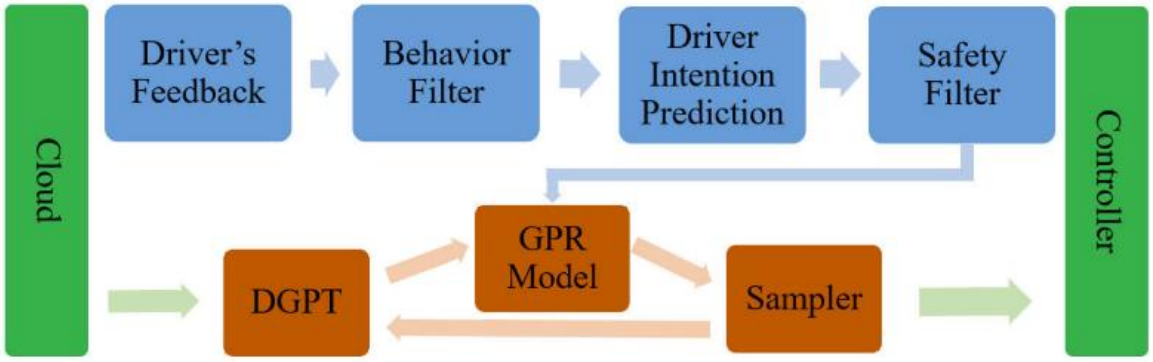


Figure 6-14 Flowchart of online DGPT adaptation algorithm.

at the current iteration, a number of trajectories are generated randomly. The initial state and time horizon of those trajectories should also match the distribution of the demonstration randomly.

Although a control policy $\pi^*(s,a)$ can also be derived during the training process as a byproduct using the value iteration algorithm, it can not be used directly as a P-ACC controller. A major reason is that the demonstration trajectory cannot fully cover the state space, and the performance of the learned policy cannot be guaranteed. Instead of the end-to-end manner, MPC is implemented later to track the value of IRL speed- g_{desired} table. The static driving behavior (i.e., the preferred gap at different speeds) is calculated using equation (6-21) from the recovered reward function r. Also, a low pass filter is applied afterwards to ensure smoothness.

$$g_{\text{desired}}(v) = \arg\max_{g} r(v) \tag{6-21}$$

The Model Predictive Control (MPC), also known as Receding Horizon Control (RHC), is a well-used model-based optimal control scheme that involves repeatedly solving a constrained objective function of control efforts, disturbances, and constraints over a moving horizon to calculate control input. It is widely used in automobile control

tasks and has been proven to be accurate and robust. In this section, we adapt MPC in a linear quadratic form to keep a balance between the convergence speed of the tracking error and control effect. The optimization problem is formulated as follows:

$$
\begin{aligned}
&min\mathrm{J} = \frac{1}{2}\sum_{t=0}^{M-1}\{(g[t]-h[t])^T \cdot Q \cdot (g[t]-h[t]) \\
&+a^T[t] \cdot R \cdot a[t])\} \\
&+\frac{1}{2}(g[M]-h[M])^T \cdot Q \cdot (g[M]-h[M]) \\
&\text{s.t. (3) and (4)} \\
&Acc_{min} \leqslant a[t] \leqslant Acc_{max} \\
&g[t] \geqslant Gap_{min}
\end{aligned}
\tag{6-22}
$$

where M is the optimal control horizon; $h[t]$ is the desired gap to be tracked; Q and R are the diagonal matrices, respectively, defining weights of the objective function to be selected adaptively based on dynamic preference of the target driver; $[Acc_{min}, Acc_{max}]$ is the range of acceleration that the ego vehicle can achieve, and boundary values can vary with respect to the vehicle speed. Gap_{min} is the safety gap to avoid collisions.

The controller's goal is to direct the vehicle to follow its leader with the desired gap, which is a function of speed given by a speed- g_{desired} table learned from IRL. The table represents static preference. In other words, if the controller can maintain the desired gap as the equilibrium, it would best satisfy the driver's preference. On the other hand, if the vehicle's car-following state is off from the equilibrium, how the vehicle is controlled to converge is called *dynamic driving style*. As introduced previously, we assume the car-following dynamic as a discrete linear system, and the controller is chosen to be an MPC with a linear quadratic objective function. The advantage using the (linear quadratic) LQ form of the objective function is that the control input can be easily obtained by solving

the algebraic Riccati equation [154] recursively. Because the control input/system dynamic is parameterized by Q and R in equation (6-22), and they keep balance between the speed of convergence and the effort of control, it is possible to change Q and R to adapt the driver's dynamic preference.

The parameter adaptation process is based on short-term observations of the driver. Specifically, there is a set of parameter pairs in a candidate pool. When conducting parameter adaptation, a short-term human-driving trajectory is collected as a benchmark, and each parameter pair is used to reproduce the same trajectory given the information of the front vehicles. Then, the candidate with the smallest Mean Square Error (MSE), compared to the benchmark, is selected for P-ACC. Because solving the algebraic Riccati equation is relatively efficient, in practice, it is still tractable to have a set of a large number (50) of candidates.

Compared to the naturalistic car-following data that is used for training the IRL model, the online feedback data usually has a very short time period and is distributed sparsely in the time domain. Therefore, preprocesses are required to use this data to maintain the DGPT, as illustrated in blue blocks of Figure 6-11.

First, a behavior filter is needed to ensure that only necessary updates are made to the DGPT since drivers' takeover behaviors can be very noisy and may only last for a short duration or have small inputs. Second, we need to infer the preferred steady state that the driver wishes to achieve through a short period of takeover trajectory. Intuitively, this steady state should correspond to the state when the driver stops the takeover. However, based on extensive experiments and observations, we found that this assumption is not accurate. Drivers may anticipate or prolong takeover behavior based on the speed

difference with the preceding vehicle, or they may achieve a steady state through multiple takeovers. Therefore, a robust prediction mechanism is needed to determine the steady state that needs to be updated. Finally, the DGPT may fall into an unreasonable range due to emergency situations or driver's mistakes, leading to potential safety hazards. Therefore, a safety space is defined for the DGPT, and a safety filter is applied to ensure that the DGPT remains bounded within this safe space.

As demonstrated, the prepossessing has the potential to be a highly intricate system. In this study, we propose a simplified heuristic algorithm that adheres to the aforementioned workflow in order to validate its effectiveness. The algorithm is shown in *Algorithm 1*. v and g are the current speed and gap; v_f is the current speed of the preceding vehicle; p is the takeover status; p_t is the takeover time; P_T is minimum takeover time; K_T is the coast-down coefficient; V_D is the maximum speed difference; Safe_TG_max and Safe_TG_min are the safety time gap bounds; v_{out} and g_{out} are predicted steady state feedback.

***Algorithm 6-2**: Driver's Feedback Preprocessing*

Data:

Input: DGPT, $v,\ v_f, g, p, p_t$

Parameters: P_T, K_T, V_D, window_size, Safe_TG_max, and Safe_TG_min

Result: v_{out}, g_{out}

1: **for** each iteration **do**

2: **if** not p **then**

3: **if** $(v - v_f) \geq V_D$ or $p_t \leq P_T$ **then**

4: $update\_flag$=False

5: **end**

6: **else**

7: $update\_flag$=True

8: **end**

9: **if** $update\_flag$ **then**

10: **if** $v_f \geq v$ **then**

11: $g_{desire} = g$

12: **end**

13: **else**

14: $g_{desire} = g\left(K_T + \left(v - v_f\right)\right)$

15: **end**

16: $v_{out} = v$

17: $g_{out} = \max\left(g_{desire}, \text{Safe\_TG\_min}\right)$

18: $g_{out} = \min\left(g_{out}, \text{Safe\_TG\_max}\right)$

19: **end**

20: **end**

21: **end**

Gaussian processes extend multivariate Gaussian distributions to infinite dimensionality. They are a form of supervised learning and the training result represents a nonlinear mapping, $f\_GP(z): R^{dim}(z) \rightarrow R$. Here, the dimension of the input vector is 1. The mapping between the input vector z and the function value $f\_GP(z)$ is accomplished

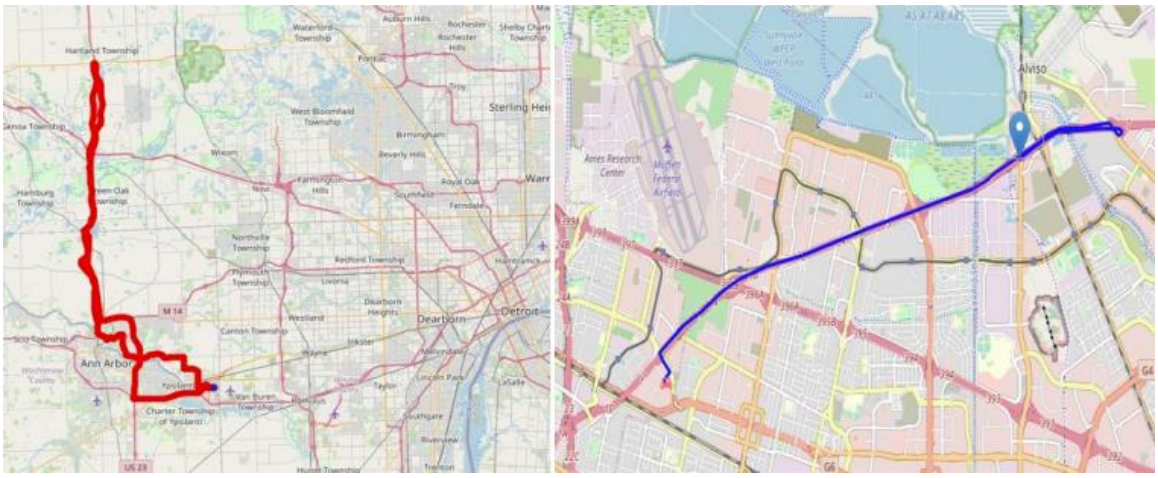


Figure 6-15 Example trajectories of the naturalistic driving data collected by our test vehicle in Ann Arbor, MI (left) and Mountain View, CA (right).

by the assumption that $f\_GP(z)$ is a random variable and is jointly Gaussian distributed with z, which is also assumed to be a random variable [183].

The configuration of the GPR model includes selecting the model regressors, the mean function, and the covariance function (i.e., kernel function). In this study, we apply a commonly used zero-mean and the squared-exponential covariance function that relates two sample input vectors z_i and z_j:

$$c(\mathbf{z}_i, \mathbf{z}_j) = \sigma_j^2 \exp\left(-\frac{1}{2}(\mathbf{z}_i - \mathbf{z}_j)^T P^{-1}(\mathbf{z}_i - \mathbf{z}_j)\right) + \sigma_n^2 \delta_{ij} \qquad (6\text{-}23)$$

where $\sigma_{ij} = 1$ if i = j and $\sigma_{ij} = 0$ otherwise, and $P = \text{diag}\left[l_1^2, \ldots, l_{\dim(\mathbf{z})}^2\right]$ contains the characteristic length scale for each dimension of the input vector. The hyperparameters of the covariance function $\theta = \left[\sigma_f, \sigma_n, l_1, \ldots, l_{\dim(\mathbf{z})}\right]^T$ include the measurement noise σ_n,the

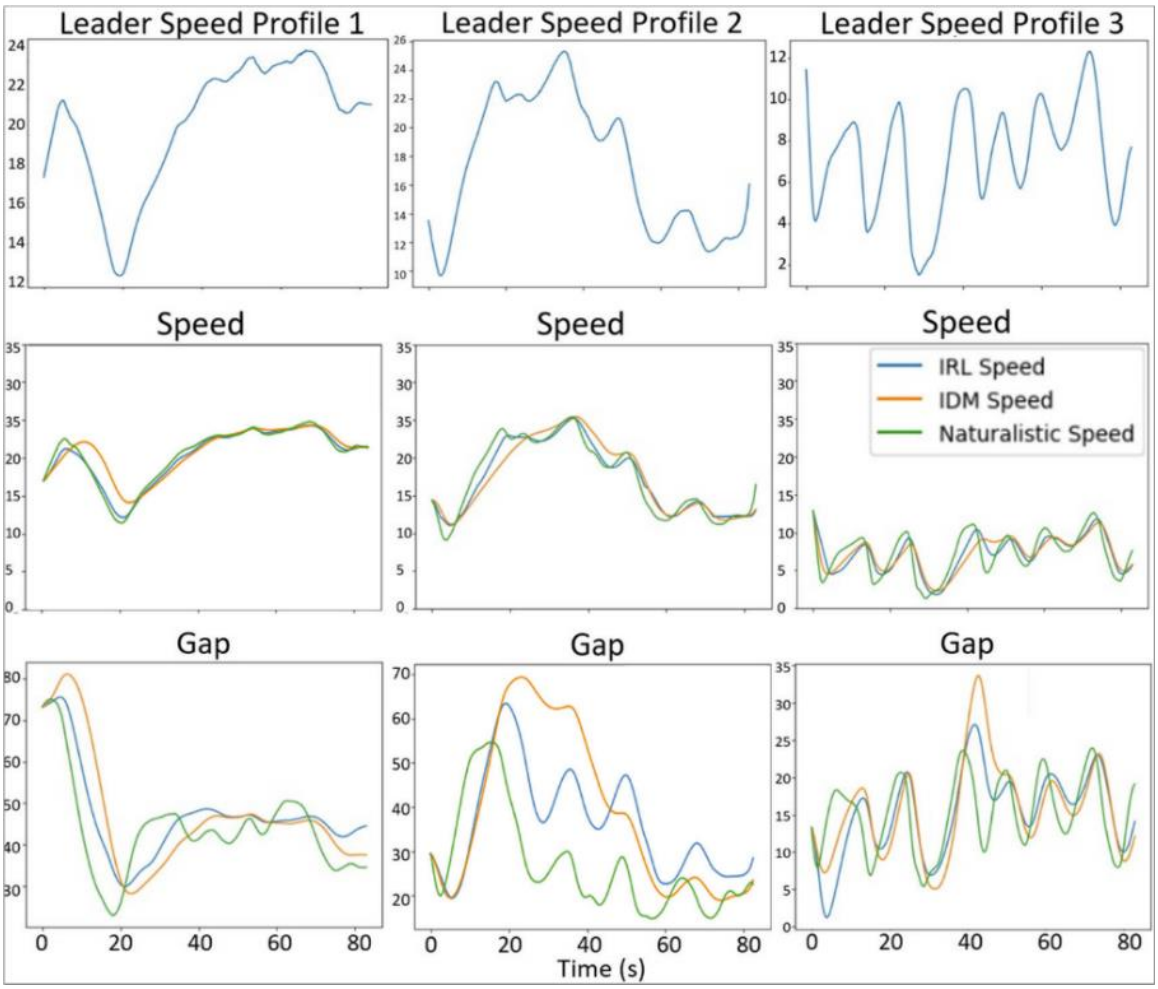


Figure 6-16 Simulated trajectories of numerical experiments.

process standard deviation σ_f, and the characteristic length scales, which are learned by maximizing the likelihood of the observation.

6.2.4 Numerical Experiment

We use a Lexus prototype vehicle to collect naturalistic driving data for training the P-ACC model. The data is collected from Michigan and California, and two example trajectories are demonstrated in Figure 6-15. The status of the preceding vehicle is from the ACC radar equipped on the prototype vehicle, and only the driving events containing car-following maneuvers are picked out from the raw data for training and validation. Also,

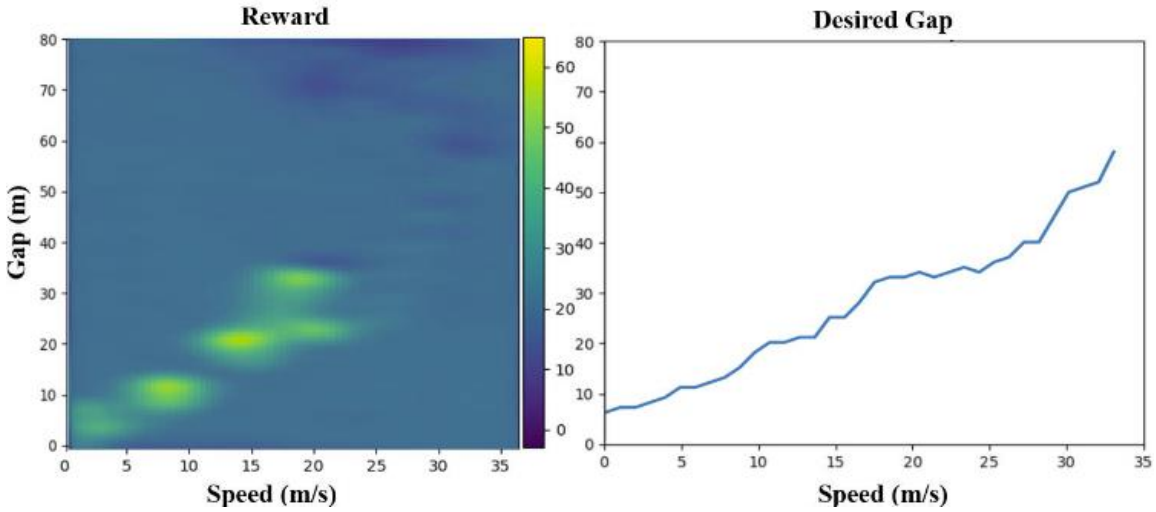


Figure 6-17 Static driving preference from IRL modeling: (a) recovered reward function from naturalistic driving data using IRL, (b) smoothed DGPT.

the car-following trajectories are down sampled and synchronized at 10Hz. Because Max-Ent-IRL is applied to recover the driver's preference, it is expected that the trajectories should cover the whole range of speed in the state space (extents from 0 to 35m/s). The car-following events at different speed ranges are selected.

The recovered reward in the speed-gap space, using IRL, is presented on Figure 6-17 (a), where dark area (low reward) in the figure means the driver is very unhappy with the corresponding states which the driver always wants to avoid. The reason why the reward is lower than other unfavorable states is that those are the beginning of the new car-following events due to cut-in by others. Therefore, it is expected that if more trajectories are collected, the effect of the beginning states would be smaller. The speed- g_{desired} table is presented on Figure 6-17 (b). As seen from the figure, the desired gap increases as the ego vehicle speeds up.

After IRL training and MPC parameter adaptation, three leader speed profiles, with different speed ranges and characteristics, are chosen for validation, which are shown as

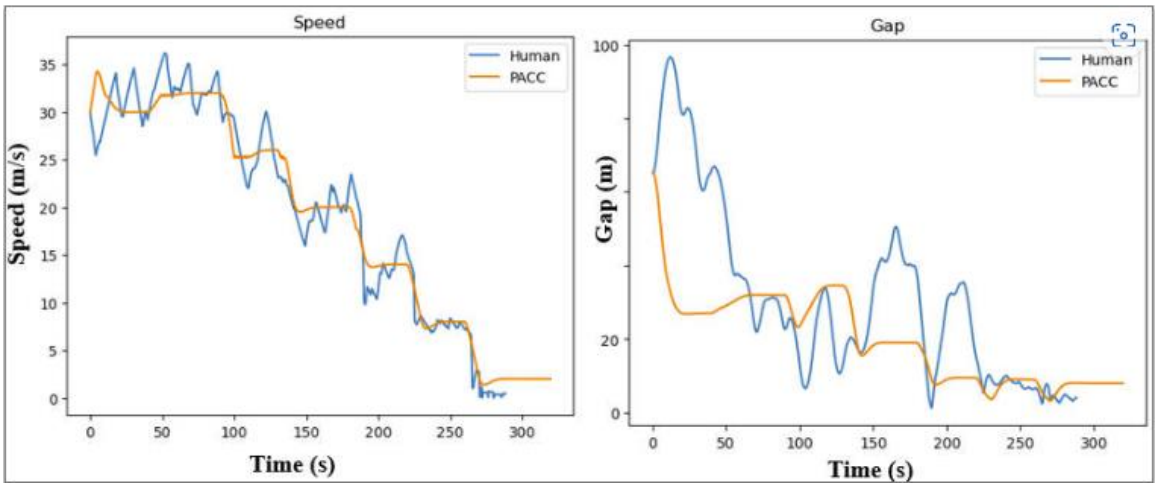


Figure 6-18 Human driver vs. P-ACC.

the first three subfigures in Figure 6-16. The car-following trajectories are calculated based on the trained P-ACC model given different leader speed profiles. Then, the root mean square percentage error (RMSPE) metric, as defined in equation (6-24), is applied for both speed and gap to measure the reproduce-ability of the proposed P-ACC system.

$$\text{RMSPE}\ (x) = \sqrt{\frac{\sum_t\ (\hat{x}[\ell] - x[l])^2}{\sum_t\ x[l]^2}} \tag{6-24}$$

where $\hat{x}$ represents the simulated trajectory based on P-ACC model or the baseline model; x is the actual trajectory that the ego vehicle drive. The RMSPE metric is a measure of how close one trajectory is to another along each time step.

Figure 6-19 Human-in-the-loop simulator.

For comparison, we implemented the widely used ODE-based method, the Intelligent Driver Model (IDM) [184], as the baseline controller, which is defined by equation (6-25) and (6-26).

$$\dot{v} = a\left(1 - \left(\frac{v}{v_0}\right)^{\delta} - \left(\frac{s^*\left(v, v - v_f\right)}{g}\right)^2\right) \tag{6-25}$$

$$s^*\left(v, v - v_f\right) = s_0 + v \cdot T + \frac{v \cdot \left(v - v_f\right)}{2\sqrt{ab}} \tag{6-26}$$

The lower six plots in Figure 6-16 show the simulated speed and gap trajectories of the proposed P-ACC and IDM compared to the real data. As observed from the figures, the proposed P-ACC can reproduce the speed and gap trajectories successfully. However, due to the linear control feedback of the proposed P-ACC, there are oscillations and overshoots. Table 6-5 lists the quantitative results measured using RMSPE. From the table, we can see that the proposed P-ACC has better trajectory reproduce-ability than IDM. However, due to the imperfectness and randomness of human driving, a low RMSPE does not always mean good performance. An example is presented in Figure 6-18. The blue trajectory is

from the human driver, and the orange trajectory is from the P-ACC. In this experiment, driver follows the leading vehicle with a stepwise slowdown pattern. As can be observed, the P-ACC strictly follows the desired gap learned from IRL. However, the human driving trajectory has large oscillation. As a result, the RMSPE is not small.

Table 6-5 Quantitative results of numerical simulation

	Speed RMSPE			Gap RMSPE		
	Profile 1	Profile 2	Profile 3	Profile 1	Profile 2	Profile 3
IRL	6.4%	8.7%	25.4%	20.9%	63.4%	35.0%
IDM	7.8%	10.5%	25.7%	26.6%	71.1%	39.1%
Improvement	17.9%	17.1%	1.7%	21.4%	10.8%	10.5%

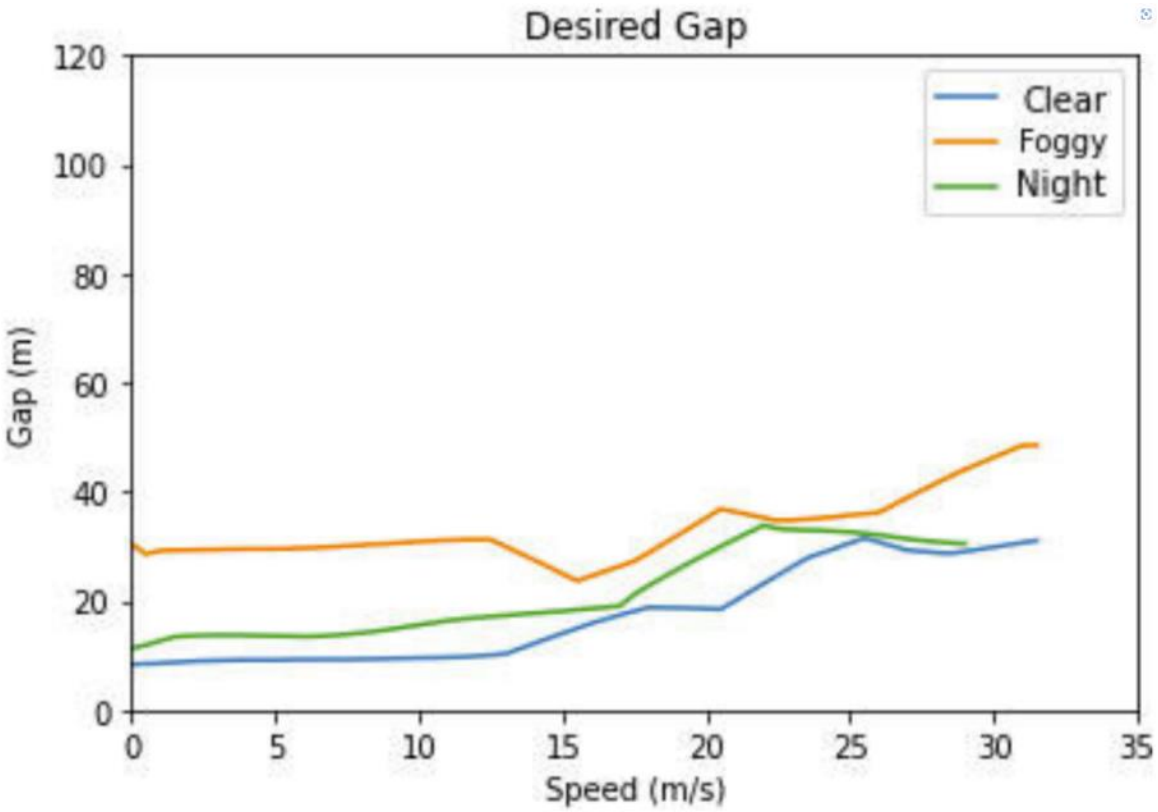


Figure 6-20 Three simulated weather conditions in the game engine: Clear Sky (Day), Clear Sky (Night), and Foggy.

6.2.5 Simulation Study

Game engines have been widely used by software developers to develop video games, which normally include a physics engine, a rendering engine, and a scene graph to manage multiple game elements (e.g., sound, threading, scripting, and models). Very recently, game engines have been adopted by other areas than video gaming, where the automotive industry becomes one of the major beneficiaries among all. This emerging technology has been used to simulate various advanced vehicular technologies [185], including autonomous driving [186], [187], cooperative driving [109], human-machine interface [188], and driver behavior modeling [188], [189].

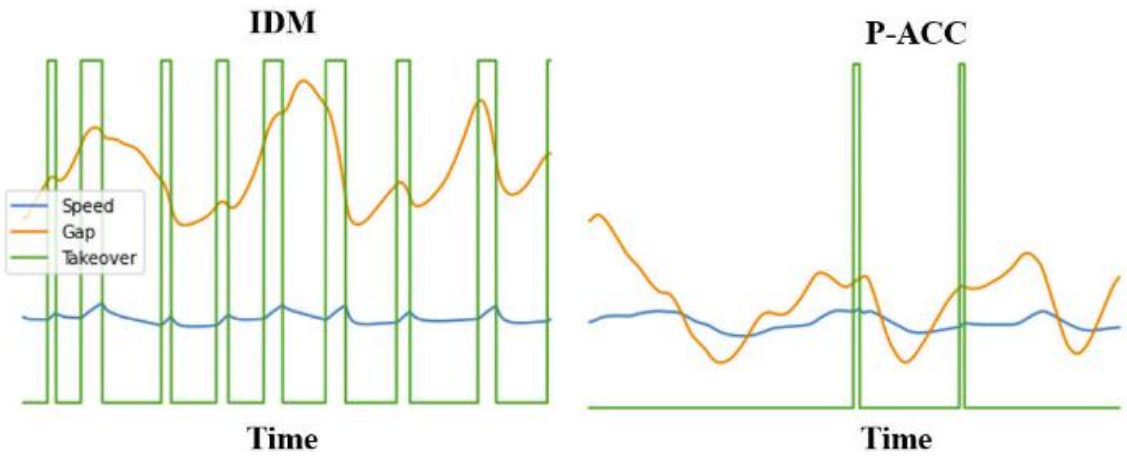


Figure 6-21 Driver's interruption during the trip.

In this section, we conduct human-in-the-loop simulations on the game engine-based driving simulator. This platform is built with the Unity game engine and a Logitech G27 Racing Wheel (see Figure 6-21). The simulation environment has a three-lane freeway scenario with flexible weather conditions. The driver may experience the change in weather both visually and aurally as shown in Figure 6-21. During the test, the driver can choose to either drive the ego vehicle manually, or drive with P-ACC by monitoring the automation and pressing the acceleration pedal or brake pedal when he/she feels uncomfortable.

For each driving test, a speed profile of the leading vehicle is required to define a car-following scenario. The speed profile has the following requirement:

- To avoid being weary during the test, each driving cycle is set to be 300 seconds.
- It is expected to collect the demonstration trajectory covering as much the range of speed as possible for training the IRL model.
- The speed profile of the leading vehicle should be close to real-world driving.
- The scenario has to be random so that the test driver cannot predict what would happen.

To meet the above four requirements, we develop a stochastic scenario generation approach to synthesize the speed profile for the leading vehicle. First, a set of short

Figure 6-22 Static preference of Driver A in three simulated weather conditions.

trajectory segments are sampled from naturalistic driving data. Then, the high-level random speed sequence is generated. Each element of the sequence defines an average speed for a period of time. Next, based on the random speed sequence, the trajectory segments with corresponding average speed are randomly selected and concatenated together. Finally, a filter is applied to ensure the acceleration/deceleration of the synthesized speed profile is bounded as required. Because each driving cycle is only 300 seconds, and the drastic change in speed may lead to abnormal driving behaviors, the test drivers need to take multiple driving cycles, and each of them covers a different speed range.

There are five drivers involved in the P-ACC test. Four of them experience both the modeling stage and control stage in different scenarios. The rest one driver experiences only the control stage. In the control stage, four drivers monitor the system on the screen and react by pressing the gas pedal or brake pedal if he/she feels uncomfortable. The last driver is observed on a short-term trajectory, and then, based on the driver type classification and scenario defined by environmental factors, the most suitable model is downloaded and applied.

Figure 6-22 shows the speed- g_{desired} table of Driver A learned from IRL in three different scenarios defined by environmental factors. According to the figure, Driver A likes to maintain a smaller gap in the clear sky (day) condition, a medium gap in the clear

sky (night) condition, and a larger gap in the foggy weather condition. This implies that, when the driving condition (e.g., visibility) is relatively good, Driver A drives more aggressively. What's more, in foggy weather, the desired gap at around 15m/s differs from the gap at it's neighboring speed. It reveals that the driver does not always behave optimally and coherently as expected, and this imperfection can be observed even more significantly in bad weather conditions.

We use Percentage of Interruption (PoI) and Number of Interruption-per-Minute (NIM) to quantitatively measure the driver's comfort and trust in the P-ACC system. PoI denotes the time percentage when the driver steps onto the acceleration pedal or brake pedal, and NIM denotes the number of times the driver steps onto the pedals. The results are presented in Table 6-6, and example trajectories are shown in Figure 6-21. As seen from the table, the PoI has been greatly reduced, which indicates the drivers are satisfied with automatic car-following based on the proposed P-ACC. Also, the less interruption of Driver E is untrained. This proves the effectiveness of the KL-divergence-based driver type classification module.

Table 6-6 Results of Human-in-the-Loop simulation: Percentage-of-Interruption (PoI) during a 300-Sec trip

Driver	P-ACC	IDM	Improvement
A	3.4%	18.3%	81.4%
B	2.4%	15.6%	84.6%
C	1.9%	9.7%	80.4%
D	3.4%	12.0%	71.6%
E (Untrained)	3.7%	14.6%	74.8%

To validate the effectiveness of the online adaptation function, we test four different combinations of controllers: *Predefined*, *Predefined + Online Adaptation*, *IRL*, and *IRL + Online Adaptation*. The *Predefined* ACC controller involves the driver choosing a constant time headway from high (4s), medium (3s), and low (1s) levels as the control reference, which is similar to the ACCs currently equipped on commercial vehicles. The *Predefined + Online Adaptation* setup involves the driver choosing a constant time headway as the control reference but incorporating an online adaptation algorithm to update the control reference table based on real-time feedback, achieving a certain degree of personalization. *IRL* involves using the DGPT trained through offline IRL to control the vehicle, which has been shown in our previous studies to significantly improve drivers' comfort and trust in the system compared to ACC without personalization. Finally, *IRL + Online Adaptation* is the complete proposed framework, which uses the DGPT obtained through offline IRL as the initial control reference to control the vehicle and incorporates an online adaptation algorithm to continuously refine personalization based on real-time feedback.

Table 6-7 Effectiveness of online adaptation

		Predefined		Predefined + Online Adaptation		IRL		IRL + Online Adaptation	
		PoI	NIM	PoI	NIM	PoI	NIM	PoI	NIM
Driver 1	Seen	14.1%	620	2.9%	3.0	17.6%	4.7	4.7%	3.3
	Unseen	13.3%	7.0	7.3%	4.7	17.1%	5.0	10.6%	3.0
Driver 2	Seen	12.0%	5.3	3.7%	4.7	4.1%	6.3	9.3%	4.3
	Unseen	6.7%	5.7	7.8%	7.0	5.6%	5.3	11.2%	4.7
Driver 3	Seen	35.4%	29.3	10.9%	15.7	31.4%	11.0	3.2%	3.0
	Unseen	17.0%	12.7	3.1%	6.7	25.9 %	13.7	11.3%	9.0
Driver 4	Seen	26.0%	5.7	10.4%	4.0	3.6%	1.7	2.6%	13
	Unseen	21.9%	5.3	10.0%	3.0	8.7%	2.7	2.4%	13
Driver 5	Seen	29.8%	16.0	13.8%	10.3	16.3%	7.3	10.8%	6.3
	Unseen	26.8 %	143	11.6%	4.3	14.9%	6.0	93%	4.3
Average		20.3%	10.7	8.2%	6.3	14.5%	6.4	75%	4.1

6.2.6 Summary

In this study, we have introduced the design of a personalized car-following system, namely P-ACC system. We have first proposed a driver type classification approach based on the KL divergence measurement under the approximated GMM trajectory distributions. Then different scenarios are determined by granularizing the environmental factors such as road type and weather conditions. Next, for each scenario, and for each driver type, the model-based MaxEnt-IRL has been applied to learn the static driving preference, which is represented by the speed- g_{desired} tables. Based on the static driving preference, the adaptive MPC controller has been designed so that the ego vehicle can maintain the preferred gap with its leader. The parameters of the controller are able to adapt to the target driver's driving style based on short-term observations. The numerical experiments show that the proposed system can closely reproduce the real-world naturalistic driving trajectory. The performance of the P-ACC system has also been tested using a game engine-based human-in-the-loop simulator. To quantify the level of trust and comfort, PoI is used as the metric. The results show that for both the driver who gets trained directly using his or her own demonstration and the driver who is only providing the short-term trajectory, the PoI metric is reduced by at least 71% compared with the well-used IDM-based ACC.

7 CONCLUSIONS AND FUTURE WORK

7.1 Conclusions

This dissertation has extensively explored the domain of intelligent transportation systems, particularly focusing on the complex interactions and cooperation between demand and supply as well as Connected and Autonomous Vehicles (CAVs) and non-CAVs. It has proposed a comprehensive cooperative framework which aims to harmonize the decision-making of agents in multi-modal transportation environments, on both macroscopic and microscopic levels. This significant step forward has allowed a more holistic exploration and understanding of cooperation in transportation systems.

At the macroscopic level, the dissertation has delved into the realm of shared automated mobility applications and has also studied dispatching and scheduling algorithms for battery electric truck fleets. The aim is to enhance operational efficiency, which is a key factor in creating sustainable and effective transportation systems.

At the microscopic level, the dissertation introduced a novel corridor-wise ramp management system. This has the potential to improve traffic flows and efficiency in mixed traffic conditions. To further address the complexities of mixed traffic, this dissertation ventured into two important research directions: infrastructure-based sensing and driver behavior modeling. The former enhances detection and monitoring capabilities, thereby leading to improved decision-making and system management. The latter aims to better understand and predict human actions in mixed traffic scenarios, leading to improved traffic flow management.

7.2 Selected Publications Resulting from This Research

[1] Dongbo Peng, Zhouqiao Zhao, Guoyuan Wu, and Kanok Boriboonsomsin, Bi-Level Fleet Dispatching Strategy for Battery-Electric Trucks: A Real-World Case Study, Sustainability, vol. 15, issue 2, page 925, 2023.

[2] Jiahe Cao, Zhouqiao Zhao, Guoyuan Wu, Matthew J Barth, Yongkang Liu, Emrah Akin Sisbot, and Kentaro Oguchi, Real-Time Adaptive Background Subtraction for Traffic Scenarios at Signalized Intersections Based on Roadside Fish-Eye Cameras, IEEE 25th International Conference on Intelligent Transportation Systems (ITSC), Macau, China, Oct. 2022.

[3] Zhouqiao Zhao, Ziran Wang, Kyungtae Han, Rohit Gupta, Prashant Tiwari, Guoyuan Wu, and Matthew J Barth, Personalized Car Following for Autonomous Driving with Inverse Reinforcement Learning, 2022 IEEE International Conference on Robotics and Automation (ICRA), Philadelphia, PA, May 2022.

[4] Zhouqiao Zhao, Guoyuan Wu, and Matthew J Barth, Corridor-Wise Eco-Friendly Cooperative Ramp Management System for Connected and Automated Vehicles, Sustainability, vol. 13, issue 15, page 8557, 2021.

[5] Zhouqiao Zhao, Guoyuan Wu, Kanok Boriboonsomsin, and Aravind Kailas, Vehicle Dispatching and Scheduling Algorithms for Battery Electric Heavy-Duty Truck Fleets Considering En-Route Opportunity Charging, 2021 IEEE Conference on Technologies for Sustainability (SusTech), Irvine, CA, Apr. 2021.

[6] Lei Zhu, Zhouqiao Zhao, and Guoyuan Wu, Shared Automated Mobility with Demand-Side Cooperation: A Proof-Of-Concept Microsimulation Study, Sustainability, vol. 13, issue 5, page 2483, 2021.

[7] Zhouqiao Zhao, Guoyuan Wu, Ziran Wang, and Matthew J Barth, Optimal Control-Based Eco-Ramp Merging System for Connected and Automated Vehicles, 2020 IEEE Intelligent Vehicles Symposium.

[8] Zhouqiao Zhao, Ziran Wang, Guoyuan Wu, Fei Ye, and Matthew J Barth, The State-Of-The-Art of Coordinated Ramp Control with Mixed Traffic Conditions, IEEE 22nd International Conference on Intelligent Transportation Systems (ITSC), Auckland, New Zealand, Oct. 2019.

[9] Zhouqiao Zhao, Ziran Wang, Kyungtae Han, Rohit Gupta, Matthew J Barth, and Guoyuan Wu, Personalized Adaptive Cruise Control with Inverse Reinforcement Learning and Adaptive Model Predictive Control, IEEE Internet of Things Journal. (Under review)

[10] Zhouqiao Zhao, Jiahe Cao, Guoyuan Wu, Matthew J Barth, Yongkang Liu, Emrah Akin Sisbot, and Kentaro Oguchi, Development and Implementation of Fisheye Camera-Based Infrastructure-Supported Real-Time Cooperative Perception System at Intersection, IEEE Transactions on Intelligent Transportation Systems, IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2023. (Under review)

[11] Zhouqiao Zhao, Xishun Liao, Amr Abdelraouf, Kyungtae Han, Rohit Gupta, Matthew J. Barth, and Guoyuan Wu, Real-Time Learning of Driving Gap Preference for Personalized Adaptive Cruise Control, IEEE International Conference on Systems, Man, and Cybernetics (SMC) 2023. (Under review)

[12] Zhouqiao Zhao, Xishun Liao, Amr Abdelraouf, Kyungtae Han, Rohit Gupta, Matthew J. Barth, and Guoyuan Wu, Inverse Reinforcement Learning and

Gaussian Process Regression-Based Real-Time Framework for Personalized Adaptive Cruise Control, IEEE International Conference on Intelligent Transportation Systems ITSC 2023. (Under review)

7.3 Future Work

The future work segment of this dissertation sets forth several promising avenues for further research.

Refinement of the driver behavior model is a significant area for improvement. This includes effectively incorporating road geometry information into the model to deepen understanding of the environment. Further, the interaction of multiple vehicles should be assimilated into the model, reflecting more realistic traffic situations. Considerations of downstream traffic ought to be included as well, as these provide a comprehensive traffic scenario for the model. Of importance is the modeling of both short-term and long-term driver behavior to capture immediate reactions as well as the lasting impacts of traffic over time on mood.

A subsequent area of future research would be the integration of this refined driver behavior model into the Cooperative Driving Automation (CDA) for optimal system-wide operation. This merger holds potential for better predicting traffic flows and more effective traffic management.

Moreover, future research should also consider the utilization of Connected and Autonomous Vehicles (CAVs) as actuators within the traffic system, or so-called *Vehicle as an Actuator* (VaaA). This approach aims to regulate and optimize traffic flows through the use of CAVs, leading to enhanced road efficiency and safety. Viewing and utilizing

CAVs as actuators open up new avenues for manipulating traffic dynamics, which can significantly improve the performance of the overall transportation system.

BIBLIOGRAPHY

[1] U.S. Department of Transportation Federal Highway Administration, “Public Road Length - 2017 Miles By Functional System.” 2017.

[2] World Health Organization Global Health Observatory Data Repository, “Road Traffic Deaths.”

[3] U.S. Department of Transportation Federal Highway Administration, “Traffic Congestion and Reliability: Trends and Advanced Strategies for Congestion Mitigation.”

[4] US Energy Information Administration, “Monthly energy review.” pp. 4, 5, 2019.

[5] Uber, “Express Pool: A More Affordable Shared Ride.” 2019. [Online]. Available: https://www.uber.com/us/en/ride/express-pool/

[6] Z. Zhao, Z. Wang, G. Wu, F. Ye, and M. J. Barth, “The state-of-the-art of coordinated ramp control with mixed traffic conditions,” in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, IEEE, 2019, pp. 1741–1748.

[7] G. Wu *et al.*, “Dyno-in-the-Loop: An Innovative Hardware-in-the-Loop Development and Testing Platform for Emerging Mobility Technologies.” 2020.

[8] S. A. E. International, “Taxonomy and definitions for terms related to cooperative driving automation for on-road motor vehicles,” 2020.

[9] A. Kohn and others, *No contest: The case against competition*. Houghton Mifflin Harcourt, 1992.

[10] T. Caric and H. Gold, “Vehicle routing problem.” 2008.

[11] T. K. Ralphs, L. Kopman, W. R. Pulleyblank, and L. E. Trotter, “On the capacitated vehicle routing problem,” *Math Program*, vol. 94, no. 2–3, pp. 343–359, Jan. 2003.

[12] M. Desrochers, J. Desrosiers, and M. Solomon, “A new optimization algorithm for the vehicle routing problem with time windows,” *Oper Res*, vol. 40, no. 2, pp. 342–354, Apr. 1992.

[13] S. Yanik, B. Bozkaya, and R. deKervenoael, “A new VRPPD model and a hybrid heuristic solution approach for e-tailing,” *Eur J Oper Res*, vol. 236, no. 3, pp. 879–890, Aug. 2014.

[14] M. Bruglieri, A. Colorni, and A. Lue, “The vehicle relocation problem for the one-way electric vehicle sharing: an application to the Milan case,” *Procedia-Social and Behavioral Sciences,* vol. 111, pp. 18–27, 2014.

[15] T. Erdelic, T. Caric, M. Erdelic, and L. Tišljaric, “Electric vehicle routing problem with single or multiple recharges,” *Transp. Res. Procedia,* vol. 40, pp. 217–224, 2019.

[16] A. Afroditi, M. Boile, S. Theofanis, E. Sdoukopoulos, and D. Margaritis, “Electric vehicle routing problem with industry constraints: trends and insights for future research,” *Transportation Research Procedia,* vol. 3, pp. 452–459, 2014.

[17] M. Furuhata, M. Dessouky, F. Ordóñez, M.-E. Brunet, X. Wang, and S. Koenig, “Ridesharing: The state-of-the-art and future directions,” *Transportation Research Part B: Methodological,* vol. 57, pp. 28–46, 2013.

[18] Z. Wang, G. Wu, and M. J. Barth, “A review on cooperative adaptive cruise control (CACC) systems: Architectures, controls, and applications,” in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, Nov. 2018, pp. 2884–2891.

[19] K. Boriboonsomsin, M. J. Barth, W. Zhu, and A. Vu, “Eco-routing navigation system based on multisource historical and real-time traffic information,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 4, pp. 1694–1704, 2012.

[20] R. T. Collins, A. J. Lipton, H. Fujiyoshi, and T. Kanade, “Algorithms for cooperative multisensor surveillance,” *Proceedings of the IEEE,* vol. 89, no. 10, pp. 1456–1477, 2001.

[21] S. A. E. International, “Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles,” 2016.

[22] T. Seo, A. M. Bayen, T. Kusakabe, and Y. Asakura, “Traffic state estimation on highway: A comprehensive survey,” *Annu Rev Control*, vol. 43, pp. 128–151, Jan. 2017.

[23] G. Newell, “A simplified theory of kinematic waves in highway traffic, part i: General theory,” *Transportation Research Part B: Methodological*, vol. 27, no. 4, pp. 281–287, 1993.

[24] B. Greenshields, J. Bibbins, W. Channing, and H. Miller, “A study of traffic capacity,” *Highway Research Board proceedings*, vol. 1935, 1935.

[25] C. F. Daganzo, “The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory,” *Transportation Research Part B: Methodological*, vol. 28, no. 4, pp. 269–287, 1994.

[26] M. Lighthill and G. Whitham, “On kinematic waves II. A theory of traffic flow on long crowded roads,” *Proc R Soc Lond A Math Phys Sci*, vol. 229, no. 1178, pp. 317–345, May 1955.

[27] P. I. Richards, “Shock Waves on the Highway,” *Oper Res*, vol. 4, no. 1, pp. 42–51, Feb. 1956.

[28] H. Payne, “Models of freeway traffic and control,” *Mathematical Models of Public Systems*, no. 1, pp. 51–61, 1971.

[29] G. B. Whitham, *Linear and Nonlinear Waves*, vol. 42. John Wiley & Sons, 1974.

[30] B. Coifman, “Estimating travel times and vehicle trajectories on freeways using dual loop detectors,” *Transp Res Part A Policy Pract*, vol. 36, no. 4, pp. 351–364, 2002.

[31] Y. Wang and M. Papageorgiou, “Real-time freeway traffic state estimation based on extended kalman filter: a general approach,” *Transportation Research Part B: Methodological*, vol. 39, no. 2, pp. 141–167, 2005.

[32] S. Tak, S. Woo, and H. Yeo, “Data-driven imputation method for traffic data in sectional units of road links,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 6, pp. 1762–1771, Jun. 2016.

[33] R. Florin and S. Olariu, “On a variant of the mobile observer method,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 2, pp. 441–449, Feb. 2017.

[34] N. Bekiaris-Liberis, C. Roncoli, and M. Papageorgiou, “Highway traffic state estimation with mixed connected and conventional vehicles,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 12, pp. 3484–3497, Dec. 2016.

[35] S. R. E. Datondji, Y. Dupuis, P. Subirats, and P. Vasseur, “A survey of vision-based traffic monitoring of road intersections,” *IEEE transactions on intelligent transportation systems*, vol. 17, no. 10, pp. 2681–2698, 2016.

[36] J.-P. Jodoin, G.-A. Bilodeau, and N. Saunier, “Urban tracker: Multiple object tracking in urban mixed traffic,” in *IEEE Winter Conference on Applications of Computer Vision*, 2014, pp. 885–892.

[37] S. Messelodi, C. M. Modena, and M. Zanin, “A computer vision system for the detection and classification of vehicles at urban road intersections,” *Pattern analysis and applications*, vol. 8, no. 1, pp. 17–31, 2005.

[38] Y. Guo, C. Rao, S. Samarasekera, J. Kim, R. Kumar, and H. Sawhney, “Matching vehicles under large pose transformations using approximate 3d models and piecewise mrf model,” in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–8.

[39] W. Hu, X. Xiao, D. Xie, T. Tan, and S. Maybank, “Traffic accident prediction using 3-D model-based vehicle tracking,” *IEEE Trans Veh Technol*, vol. 53, no. 3, pp. 677–694, 2004.

[40] J. Lou, T. Tan, W. Hu, H. Yang, and S. J. Maybank, “3-D model-based vehicle tracking,” *IEEE Transactions on image processing*, vol. 14, no. 10, pp. 1561–1569, 2005.

[41] G. Zhang, R. P. Avery, and Y. Wang, “Video-based vehicle detection and classification system for real-time traffic data collection using uncalibrated video cameras,” *Transp Res Rec*, vol. 1993, no. 1, pp. 138–147, 2007.

[42] L.-W. Tsai, J.-W. Hsieh, and K.-C. Fan, “Vehicle detection using normalized color and edge map,” *IEEE transactions on Image Processing*, vol. 16, no. 3, pp. 850–864, 2007.

[43] J. Nuevo, I. Parra, J. Sjöberg, and L. M. Bergasa, “Estimating surrounding vehicles’ pose using computer vision,” in *13th International IEEE Conference on Intelligent Transportation Systems*, 2010, pp. 1863–1868.

[44] A. Jazayeri, H. Cai, J. Y. Zheng, and M. Tuceryan, “Vehicle detection and tracking in car video based on motion model,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 2, pp. 583–595, 2011.

[45] Y.-M. Chan, S.-S. Huang, L.-C. Fu, and P.-Y. Hsiao, “Vehicle detection under various lighting conditions by incorporating particle filter,” in *2007 IEEE Intelligent Transportation Systems Conference*, 2007, pp. 534–539.

[46] J.-Y. Bouguet and others, “Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm,” *Intel corporation*, vol. 5, no. 1–10, p. 4, 2001.

[47] A. Geiger and B. Kitt, “Object flow: A descriptor for classifying traffic motion,” in *2010 IEEE Intelligent Vehicles Symposium*, 2010, pp. 287–293.

[48] J.-S. Zhang, J. Cao, and B. Mao, “Application of deep learning and unmanned aerial vehicle technology in traffic flow monitoring,” in *2017 International Conference on Machine Learning and Cybernetics (ICMLC)*, 2017, pp. 189–194.

[49] A. Aboah, “A vision-based system for traffic anomaly detection using deep learning and decision trees,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4207–4212.

[50] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[51] 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.

[52] W. Liu *et al.*, “Ssd: Single shot multibox detector,” in *European conference on computer vision*, 2016, pp. 21–37.

[53] J. Redmon and A. Farhadi, “Yolov3: An incremental improvement,” *arXiv preprint arXiv:1804.02767*, 2018.

[54] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” *Adv Neural Inf Process Syst*, vol. 28, 2015.

[55] R. Girshick, “Fast r-cnn,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[56] J. Arróspide and L. Salgado, “A study of feature combination for vehicle detection based on image processing,” *The Scientific World Journal*, vol. 2014, 2014.

[57] Z. Sun, G. Bebis, and R. Miller, “Monocular precrash vehicle detection: features and classifiers,” *IEEE transactions on image processing*, vol. 15, no. 7, pp. 2019–2034, 2006.

[58] W. Liu, X. Wen, B. Duan, H. Yuan, and N. Wang, “Rear vehicle detection and tracking for lane change assist,” in *2007 IEEE intelligent vehicles symposium*, 2007, pp. 252–257.

[59] C. Stauffer and W. E. L. Grimson, “Adaptive background mixture models for real-time tracking,” in *Proceedings. 1999 IEEE computer society conference on computer vision and pattern recognition (Cat. No PR00149)*, 1999, pp. 246–252.

[60] M. M. Azab, H. A. Shedeed, and A. S. Hussein, “A new technique for background modeling and subtraction for motion detection in real-time videos,” in *2010 IEEE International Conference on Image Processing*, 2010, pp. 3453–3456.

[61] H. Zhang and D. Xu, “Fusing color and texture features for background model,” in *Fuzzy Systems and Knowledge Discovery: Third International Conference, FSKD 2006, Xi’an, China, September 24-28, 2006. Proceedings 3*, 2006, pp. 887–893.

[62] P.-L. St-Charles and G.-A. Bilodeau, “Improving background subtraction using local binary similarity patterns,” in *IEEE winter conference on applications of computer vision*, 2014, pp. 509–515.

[63] O. Barnich and M. Van Droogenbroeck, “ViBe: A universal background subtraction algorithm for video sequences,” *IEEE Transactions on Image processing*, vol. 20, no. 6, pp. 1709–1724, 2010.

[64] J. Levinson and S. Thrun, “Robust vehicle localization in urban environments using probabilistic maps,” in *2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 4372–4378. doi: 10.1109/ROBOT.2010.5509700.

[65] Z. Bai, G. Wu, M. J. Barth, Y. Liu, A. Sisbot, and K. Oguchi, “PillarGrid: Deep Learning-based Cooperative Perception for 3D Object Detection from Onboard-Roadside LiDAR,” *arXiv preprint arXiv:2203.06319*, 2022.

[66] R. Zhang, Z. Zou, S. Shen, and H. X. Liu, “Design, Implementation, and Evaluation of a Roadside Cooperative Perception System,” *Transp Res Rec*, p. 03611981221092402, 2022.

[67] R. Pereira, G. Carvalho, L. Garrote, and U. J. Nunes, “Sort and Deep-SORT Based Multi-Object Tracking for Mobile Robotics: Evaluation with New Data Association Metrics,” *Applied Sciences*, vol. 12, no. 3, 2022, doi: 10.3390/app12031319.

[68] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, “Simple online and realtime tracking,” in *2016 IEEE international conference on image processing (ICIP)*, 2016, pp. 3464–3468.

[69] N. Wojke, A. Bewley, and D. Paulus, “Simple online and realtime tracking with a deep association metric,” in *2017 IEEE international conference on image processing (ICIP)*, 2017, pp. 3645–3649.

[70] X. Zhou, V. Koltun, and P. Krähenbühl, “Tracking objects as points,” in *European Conference on Computer Vision*, 2020, pp. 474–490.

[71] W. Wang, T. Gee, J. Price, and H. Qi, “Real time multi-vehicle tracking and counting at intersections from a fisheye camera,” in *2015 IEEE Winter Conference on Applications of Computer Vision*, 2015, pp. 17–24.

[72] J. Zhu, J. Zhu, X. Wan, C. Wu, and C. Xu, “Object detection and localization in 3D environment by fusing raw fisheye image and attitude data,” *J Vis Commun Image Represent*, vol. 59, pp. 128–139, 2019.

[73] J. Cao *et al.*, “Real-time Adaptive Background Subtraction for Traffic Scenarios at Signalized Intersections Based on Roadside Fish-eye Cameras,” in *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, 2022, pp. 4189–4195.

[74] J. Kannala and S. S. Brandt, “A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses,” *IEEE Trans Pattern Anal Mach Intell*, vol. 28, no. 8, pp. 1335–1340, 2006.

[75] E. Plaut, E. Ben Yaacov, and B. El Shlomo, “3D Object Detection from a Single Fisheye Image Without a Single Fisheye Training Image,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 3659–3667.

[76] H. Rashed *et al.*, “Generalized object detection on fisheye cameras for autonomous driving: Dataset, representations and baseline,” in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021, pp. 2272–2280.

[77] M. Yahiaoui *et al.*, “Fisheyemodnet: Moving object detection on surround-view cameras for autonomous driving,” *arXiv preprint arXiv:1908.11789*, 2019.

[78] P. G. Gipps, “A behavioural car-following model for computer simulation,” *Transportation Research Part B: Methodological*, vol. 15, no. 2, pp. 105–111, 1981.

[79] M. Treiber, A. Hennecke, and D. Helbing, “Congested traffic states in empirical observations and microscopic simulations,” *Phys Rev E*, vol. 62, no. 2, p. 1805, 2000.

[80] G. F. Newell, “A simplified car-following theory: a lower order model,” *Transportation Research Part B: Methodological*, vol. 36, no. 3, pp. 195–205, 2002.

[81] A. Khodayari, A. Ghaffari, M. Nouri, S. Salehinia, and F. Alimardani, “Model Predictive Control system design for car-following behavior in real traffic flow,” in *2012 IEEE International Conference on Vehicular Electronics and Safety (ICVES 2012)*, 2012, pp. 87–92.

[82] B. Gao, K. Cai, T. Qu, Y. Hu, and H. Chen, “Personalized adaptive cruise control based on online driving style recognition technology and model predictive control,” *IEEE Trans Veh Technol*, vol. 69, no. 11, pp. 12482–12496, 2020.

[83] H. Liu, Y.-S. Ong, X. Shen, and J. Cai, “When Gaussian process meets big data: A review of scalable GPs,” *IEEE Trans Neural Netw Learn Syst*, vol. 31, no. 11, pp. 4405–4423, 2020.

[84] W. Wang, D. Zhao, W. Han, and J. Xi, “A learning-based approach for lane departure warning systems with a personalized driver model,” *IEEE Trans Veh Technol*, vol. 67, no. 10, pp. 9145–9157, 2018.

[85] J. Hongfei, J. Zhicai, and N. Anning, “Develop a car-following model using data collected by" five-wheel system",” in *Proceedings of the 2003 IEEE International Conference on Intelligent Transportation Systems*, 2003, pp. 346–351.

[86] X. Wang, R. Jiang, L. Li, Y. Lin, X. Zheng, and F.-Y. Wang, “Capturing car-following behaviors by deep learning,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 3, pp. 910–920, 2017.

[87] L. Chong, M. M. Abbas, and A. Medina, “Simulation of driver behavior with agent-based back-propagation neural network,” *Transp Res Rec*, vol. 2249, no. 1, pp. 44–51, 2011.

[88] X. Huang, J. Sun, and J. Sun, “A car-following model considering asymmetric driving behavior based on long short-term memory neural networks,” *Transp Res Part C Emerg Technol*, vol. 95, pp. 346–362, 2018.

[89] Y. Wu, H. Tan, X. Chen, and B. Ran, “Memory, attention and prediction: a deep learning architecture for car-following,” *Transportmetrica B: Transport Dynamics*, vol. 7, no. 1, pp. 1553–1571, 2019.

[90] K. P. Wabersich and M. N. Zeilinger, “A predictive safety filter for learning-based control of constrained nonlinear dynamical systems,” *Automatica*, vol. 129, p. 109597, 2021.

[91] A. Y. Ng, S. J. Russell, and others, “Algorithms for inverse reinforcement learning.,” in *Icml*, 2000, p. 2.

[92] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, “Maximum entropy inverse reinforcement learning,” in *Aaai*, Jul. 2008, pp. 1433–1438.

[93] H. Gao, G. Shi, G. Xie, and B. Cheng, “Car-following method based on inverse reinforcement learning for autonomous vehicle decision-making,” *Int J Adv Robot Syst*, vol. 15, no. 6, p. 1729881418817162, 2018.

[94] T. Phan-Minh *et al.*, “Driving in real life with inverse reinforcement learning,” *arXiv preprint arXiv:2206.03004*, 2022.

[95] M. F. Ozkan and Y. Ma, “Personalized adaptive cruise control and impacts on mixed traffic,” in *2021 American Control Conference (ACC)*, 2021, pp. 412–417.

[96] L. Guo and Y. Jia, “Inverse Model Predictive Control (IMPC) based Modeling and Prediction of Human-Driven Vehicles in Mixed Traffic,” *IEEE Transactions on Intelligent Vehicles*, 2020.

[97] M. Zhu, X. Wang, and Y. Wang, “Human-like autonomous car-following model with deep reinforcement learning,” *Transp Res Part C Emerg Technol*, vol. 97, pp. 348–368, 2018.

[98] Y. Lin, J. McPhee, and N. L. Azad, “Longitudinal dynamic versus kinematic models for car-following control using deep reinforcement learning,” in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 1504–1510.

[99] G. K. Schmidt and B. Posch, “A two-layer control scheme for merging of automated vehicles,” in *The 22nd IEEE Conference on Decision and Control*, 1983, pp. 495–500.

[100] B. Ran, S. Leight, and B. Chang, “A microscopic simulation model for merging control on a dedicated-lane automated highway system,” *Transp Res Part C Emerg Technol*, vol. 7, no. 6, pp. 369–388, 1999.

[101] T. Awal, L. Kulik, and K. Ramamohanrao, “Optimal traffic merging strategy for communication- and sensor-enabled vehicles,” in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, Oct. 2013, pp. 1468–1474.

[102] J. Rios-Torres and A. A. Malikopoulos, “Automated and cooperative vehicle merging at highway on-ramps,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 4, pp. 780–789, Apr. 2017.

[103] Y. Xie, H. Zhang, N. H. Gartner, and T. Arsava, “Collaborative merging strategy for freeway ramp operations in a connected and autonomous vehicles environment,” *J Intell Transp Syst*, vol. 21, no. 2, pp. 136–147, 2017.

[104] A. Uno, T. Sakaguchi, and S. Tsugawa, “A merging control algorithm based on inter-vehicle communication,” in *Proceedings 199 IEEE/IEEJ/JSAI International Conference on Intelligent Transportation Systems*, Oct. 1999, pp. 783–787.

[105] X.-Y. Lu and K. J. Hedrick, “Longitudinal control algorithm for automated vehicle merging,” in *Proceedings of the 39th IEEE Conference on Decision and Control*, Dec. 2000, pp. 450–455.

[106] T. Dao, C. M. Clark, and J. P. Huissoon, “Distributed platoon assignment and lane selection for traffic flow optimization,” in *IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2008, pp. 739–744.

[107] Z. Wang, G. Wu, and M. Barth, “Distributed consensus-based cooperative highway on-ramp merging using V2X communications,” *SAE Technical Paper*, Apr. 2018.

[108] Z. Wang, B. Kim, H. Kobayashi, G. Wu, and M. J. Barth, “Agent-based modeling and simulation of connected and automated vehicles using game engine: A cooperative on-ramp merging study,” *arXiv preprint arXiv:1810.09952*, 2018.

[109] Z. Wang *et al.*, “Cooperative ramp merging system: Agent-based modeling and simulation using game engine,” *SAE International Journal of Connected and Automated Vehicles*, vol. 2, no. 2, May 2019.

[110] A. Henao and W. E. Marshall, “The impact of ride-hailing on vehicle miles traveled,” *Transportation (Amst)*, vol. 46, pp. 2173–2194, 2019.

[111] A. Tirachini and A. Gomez-Lobo, “Does ride-hailing increase or decrease vehicle kilometers traveled (VKT)? A simulation approach for Santiago de Chile,” *Int. J. Sustain. Transp.*, vol. 14, pp. 187–204, 2020, doi: 10.1080/15568318.2018.1539146.

[112] “UberX Affordable, Everyday Rides.”

[113] A. N. Pratt, E. A. Morris, Y. Zhou, S. Khan, and M. Chowdhury, “What do riders tweet about the people that they meet? Analyzing online commentary about UberPool and Lyft Shared/Lyft Line,” *Transp. Res. Part F Traffic Psychol. Behav.*, vol. 62, pp. 459–472, 2019, doi: 10.1016/j.trf.2019.01.015.

[114] C. S. Helvig, G. Robins, and A. Zelikovsky, “The moving-target traveling salesman problem,” *J. Algorithms*, vol. 49, pp. 153–174, 2003.

[115] C. Gambella, J. Naoum-Sawaya, and B. Ghaddar, “The vehicle routing problem with floating targets: Formulation and solution approaches,” *Inf. J. Comput.*, vol. 30, pp. 554–569, 2018.

[116] A. Tirachini, “Ride-hailing, travel behaviour and sustainable mobility: An international review,” *Transportation (Amst)*, vol. 47, pp. 2011–2047, 2020, doi: 10.1007/s11116-019-10070-2.

[117] N. Haglund, M. N. Mladenović, R. Kujala, C. Weckström, and J. Saramäki, “Where did Kutsuplus drive us? Ex post evaluation of on-demand micro-transit pilot in the Helsinki capital region,” *Res. Transp. Bus. Manag.*, vol. 32, 2019, doi: 10.1016/j.rtbm.2019.100390.

[118] A. Wegener, M. Piórkowski, M. Raya, H. Hellbrück, S. Fischer, and J.-P. Hubaux, "TraCI: An interface for coupling road traffic and network simulators," in *Proceedings of the 11th Communications and Networking Simulation Symposium*, Ottawa, ON, Canada, 2008, pp. 155–163.

[119] S. Shaheen and A. Cohen, "Shared Mobility Policy Briefs: Definitions, Impacts, and Recommendations," Berkeley, CA, 2018.

[120] S. A. Shaheen *et al.*, "Mobility on Demand Planning and Implementation: Current Practices, Innovations, and Emerging Mobility Futures," 2020.

[121] S. Shaheen and A. Cohen, "Shared ride services in North America: Definitions, impacts, and the future of pooling," *Transp. Rev.*, vol. 39, pp. 427–442, 2019, doi: 10.1080/01441647.2018.1497728.

[122] N. D. Chan and S. A. Shaheen, "Ridesharing in North America: Past, present, and future," *Transp. Rev.*, vol. 32, pp. 93–112, 2012.

[123] Y. Dumas, J. Desrosiers, and F. Soumis, "The pickup and delivery problem with time windows," *Eur. J. Oper. Res.*, vol. 54, pp. 7–22, 1991.

[124] K. Braekers, K. Ramaekers, and I. Van Nieuwenhuyse, "The vehicle routing problem: State of the art classification and review," *Comput. Ind. Eng.*, vol. 99, pp. 300–313, 2016.

[125] Z. Yang *et al.*, "Dynamic vehicle routing with time windows in theory and practice," *Nat. Comput.*, vol. 16, pp. 119–134, 2017.

[126] J. Alonso-Mora, S. Samaranayake, A. Wallar, E. Frazzoli, and D. Rus, "On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment," *Proc. Natl. Acad. Sci. USA*, vol. 114, pp. 462–467, 2017.

[127] S. Ropke and D. Pisinger, "An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows," *Transp. Sci.*, vol. 40, pp. 455–472, 2006.

[128] X. Wang, "Optimizing Ride Matches for Dynamic Ride-Sharing Systems." Atlanta, Georgia, 2013.

[129] A. Simonetto, J. Monteil, and C. Gambella, "Real-time city-scale ridesharing via linear assignment problems," *Transp. Res. Part C Emerg. Technol.*, vol. 101, pp. 208–232, 2019, doi: 10.1016/j.trc.2019.01.019.

[130] V. Pandey, J. Monteil, C. Gambella, and A. Simonetto, “On the needs for MaaS platforms to handle competition in ridesharing mobility,” *Transp. Res. Part C Emerg. Technol.*, vol. 108, pp. 269–288, 2019, doi: 10.1016/j.trc.2019.09.021.

[131] B. Coltin and M. Veloso, “Ridesharing with passenger transfers,” in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Chicago, IL, USA, 2014, pp. 3278–3283.

[132] M. Stiglic, N. Agatz, M. Savelsbergh, and M. Gradisar, “The benefits of meeting points in ride-sharing systems,” *Transp. Res. Part B Methodol.*, vol. 82, pp. 36–53, 2015.

[133] X. Li, S. Hu, W. Fan, and K. Deng, “Modeling an enhanced ridesharing system with meet points and time windows,” *PLoS One*, vol. 13, 2018, doi: e0195927.

[134] M. Zhao, J. Yin, S. An, J. Wang, and D. Feng, “Ridesharing Problem with Flexible Pickup and Delivery Locations for App-Based Transportation Service: Mathematical Modeling and Decomposition Methods,” *J. Adv. Transp.*, 2018, doi: 6430950.

[135] “Multimodal Transportation Software—PTV Group.”

[136] L. Zhu, Z. Zhao, and G. Wu, “Shared automated mobility with demand-side cooperation: A proof-of-concept microsimulation study,” *Sustainability*, vol. 13, no. 5, p. 2483, 2021.

[137] L. Zhu, J. Wang, V. Garikapati, and S. Young, “Decision support tool for planning neighborhood-scale deployment of low-speed shared automated shuttles,” *Transp. Res. Rec.*, vol. 2674, pp. 1–14, 2020.

[138] Y. Huang, K. M. Kockelman, V. Garikapati, L. Zhu, and S. Young, “Use of Shared Automated Vehicles for First-Mile Last-Mile Service: Micro-Simulation of Rail-Transit Connections in Austin, Texas,” *Transp. Res. Record*, 2020, doi: 10.1177/0361198120962491.

[139] D. J. Fagnant, K. M. Kockelman, and P. Bansal, “Operations of a shared autonomous vehicle fleet for the Austin, Texas market,” *Transp. Res. Board*, vol. 2563, pp. 98–106, 2015.

[140] C. Wang, Y. Hou, and M. Barth, “Data-Driven Multi-step Demand Prediction for Ride-Hailing Services Using Convolutional Neural Network,” in *Advances in Computer Vision*, in Advances in Intelligent Systems and Computing, vol. 944. Cham: Springer, 2019, pp. 11–22. doi: 10.1007/978-3-030-17798-0_2.

[141] C. G. Petersen, “An evaluation of order picking routeing policies,” *Int. J. Oper. Amp; Prod. Manag.*, vol. 17, pp. 1098–1111, 1997, doi: 10.1108/01443579710177860.

[142] T. Schneider, “Analyzing 1.1 Billion NYC Taxi and Uber Trips, with a Vengeance.” 2021.

[143] C. S. Helvig, G. Robins, and A. Zelikovsky, “The moving-target traveling salesman problem,” *Journal of Algorithms*, vol. 49, no. 1, pp. 153–174, 2003.

[144] J. F. Bard and others, “A branch and cut algorithm for the VRP with satellite facilities,” *IIE transactions*, vol. 30, no. 9, pp. 821–834, 1998.

[145] M. Keskin and B. Çatay, “Partial recharge strategies for the electric vehicle routing problem with time windows,” *Transp Res Part C Emerg Technol*, vol. 65, pp. 111–127, 2016.

[146] A. Omidvar and R. Tavakkoli-Moghaddam, “Sustainable vehicle routing: Strategies for congestion management and refueling scheduling,” in *2012 IEEE international energy conference and exhibition (ENERGYCON)*, IEEE, 2012.

[147] A. E. Rizzoli and others, “Ant colony optimization for real-world vehicle routing problems,” *Swarm Intelligence*, vol. 1, no. 2, pp. 135–151, 2007.

[148] G. Hiermann and others, “Routing a mix of conventional, plug-in hybrid, and electric vehicles,” *Eur J Oper Res*, vol. 272, no. 1, pp. 235–248, 2019.

[149] D. Peng, Z. Zhao, G. Wu, and K. Boriboonsomsin, “Bi-Level Fleet Dispatching Strategy for Battery-Electric Trucks: A Real-World Case Study,” *Sustainability*, vol. 15, no. 2, p. 925, 2023.

[150] Z. Zhao, G. Wu, K. Boriboonsomsin, and A. Kailas, “Vehicle Dispatching and Scheduling Algorithms for Battery Electric Heavy-Duty Truck Fleets Considering En-route Opportunity Charging,” in *2021 IEEE Conference on Technologies for Sustainability (SusTech)*, 2021, pp. 1–8.

[151] Z. Zhao, G. Wu, Z. Wang, and M. J. Barth, “Optimal Control-Based Eco-Ramp Merging System for Connected and Automated Electric Vehicles,” *arXiv preprint arXiv:1910.07620*, 2019.

[152] Z. Zhao, G. Wu, and M. Barth, “Corridor-Wise Eco-Friendly Cooperative Ramp Management System for Connected and Automated Vehicles,” *Sustainability*, vol. 13, no. 15, p. 8557, 2021.

[153] N. Geroliminis, A. Srivastava, and P. Michalopoulos, “Development of the Next Generation Stratified Ramp Metering Algorithm Based on Freeway Design,” 2011.

[154] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, “The explicit linear quadratic regulator for constrained systems,” *Automatica*, vol. 38, no. 1, pp. 3–20, 2002.

[155] M. A. S. Kamal, M. Mukai, J. Murata, and T. Kawabe, “Ecological vehicle control on roads with up-down slopes,” *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 3, pp. 783–794, 2011.

[156] F. Ye, G. Wu, K. Boriboonsomsin, and M. J. Barth, “A hybrid approach to estimating electric vehicle energy consumption for ecodriving applications,” in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, 2016, pp. 719–724.

[157] A. Kesting, M. Treiber, and D. Helbing, “Enhanced intelligent driver model to access the impact of driving strategies on traffic capacity,” *Philos. Trans. R. Soc. A Math. Phys. Eng. Sci.*, vol. 368, no. 1928, pp. 4585–4605, 2010.

[158] Q. Jin, G. Wu, K. Boriboonsomsin, and M. J. Barth, “Power-based optimal longitudinal control for a connected eco-driving system,” *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 10, pp. 2900–2910, 2016.

[159] S. Hussain, Z. Peng, and M. I. Hayee, “Development and Demonstration of Merge Assist System Using Connected Vehicle Technology,” 2019.

[160] E.-H. Choi, “Crash factors in intersection-related crashes: An on-scene perspective,” 2010.

[161] S. Soleimaniamiri *et al.*, “Cooperative Automation Research: CARMA Proof-of-Concept Transportation System Management and Operations Use Case 2,” 2021.

[162] E. T. S. Institute, “Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service,” *ETSI: Sophia-Antipolis*, 2019.

[163] J. B. Kenney, “Dedicated short-range communications (DSRC) standards in the United States,” *Proceedings of the IEEE*, vol. 99, no. 7, pp. 1162–1182, 2011.

[164] E. T. S. Institute, “Intelligent Transport Systems (ITS); Cooperative Perception Services (CPS),” *ETSI: Sophia-Antipolis*, 2019.

[165] E. T. S. Institute, “Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Analysis of the Collective Perception Service (CPS); Release 2,” *ETSI: Sophia-Antipolis*, 2019.

[166] Z. Zhang, “A flexible new technique for camera calibration,” *IEEE Trans Pattern Anal Mach Intell*, vol. 22, no. 11, pp. 1330–1334, 2000.

[167] A. G. Jain and N. Saunier, “Autocamera Calibration for traffic surveillance cameras with wide angle lenses,” *arXiv preprint arXiv:2001.07243*, 2020.

[168] G.-A. Bilodeau, J.-P. Jodoin, and N. Saunier, “Change detection in feature space using local binary similarity patterns,” in *2013 International conference on computer and robot vision*, 2013, pp. 106–112.

[169] P.-L. St-Charles, G.-A. Bilodeau, and R. Bergevin, “SuBSENSE: A universal change detection method with local adaptive sensitivity,” *IEEE Transactions on Image Processing*, vol. 24, no. 1, pp. 359–373, 2014.

[170] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, “CARLA: An Open Urban Driving Simulator,” in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[171] MathWorks, “RoadRunner.” https://www.maathworks.com/products/roadrunner.html

[172] Toyota, “Toyota Safety Sense: The Standard for Safety.” 2021. [Online]. Available: https://www.toyota.com/safety-sense/

[173] Volkswagen, “Adaptive Cruise Control.” 2021. [Online]. Available: https://www.volkswagen-newsroom.com/en/adaptive-cruise-control-acc-3664

[174] Ford, “Adaptive Cruise Control.” 2021. [Online]. Available: https://www.ford.com/technology/driver-assist-technology/adaptive-cruise-control/

[175] S. E. Shladover *et al.*, “Automated vehicle control developments in the PATH program,” *IEEE Trans Veh Technol*, vol. 40, no. 1, pp. 114–130, 1991.

[176] M. Hasenjäger, M. Heckmann, and H. Wersing, “A survey of personalization for advanced driver assistance systems,” *IEEE Transactions on Intelligent Vehicles*, vol. 5, no. 2, pp. 335–344, 2019.

[177] Z. Zhao *et al.*, “Personalized car following for autonomous driving with inverse reinforcement learning,” in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 2891–2897.

[178] J. M. Wang, D. J. Fleet, and A. Hertzmann, “Gaussian process dynamical models for human motion,” *IEEE Trans Pattern Anal Mach Intell*, vol. 30, no. 2, pp. 283–298, 2007.

[179] Z. Wang *et al.*, “Mobility digital twin: Concept, architecture, case study, and future challenges,” *IEEE Internet Things J*, vol. 9, no. 18, pp. 17452–17467, 2022.

[180] C. Miyajima *et al.*, “Driver modeling based on driving behavior and its evaluation in driver identification,” *Proceedings of the IEEE*, vol. 95, no. 2, pp. 427–437, 2007.

[181] OpenWeatherMap, “Weather API.” 2021. [Online]. Available: https://openweathermap.org/api

[182] R. S. Sutton, A. G. Barto, and others, *Introduction to reinforcement learning*, vol. 135. MIT press Cambridge, 1998.

[183] C. E. Rasmussen, “Gaussian processes in machine learning,” in *Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2-14, 2003, Tübingen, Germany, August 4-16, 2003, Revised Lectures*, Springer, 2004, pp. 63–71.

[184] M. Bando, K. Hasebe, A. Nakayama, A. Shibata, and Y. Sugiyama, “Dynamical model of traffic congestion and numerical simulation,” *Phys. Rev. E*, vol. 51, no. 2, pp. 1035–1042, Feb. 1995, doi: 10.1103/PhysRevE.51.1035.

[185] C. and W. Z. and E. M. and R. G. and F. Y. Ma Jiaqi and Schwarz, “New Simulation Tools for Training and Testing Automated Vehicles,” in *Road Vehicle Automation 7*, S. Meyer Gereon and Beiker, Ed., Cham: Springer International Publishing, 2020, pp. 111–119.

[186] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, “CARLA: An open urban driving simulator,” in *Conference on robot learning*, 2017, pp. 1–16.

[187] G. Rong *et al.*, “Lgsvl simulator: A high fidelity simulator for autonomous driving,” in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, 2020, pp. 1–6.

[188] Z. Wang *et al.*, “Driver behavior modeling using game engine and real vehicle: A learning-based approach,” *IEEE Transactions on Intelligent Vehicles*, vol. 5, no. 4, pp. 738–749, 2020.

[189] X. Zhao *et al.*, “Co-simulation platform for modeling and evaluating connected and automated vehicles and human behavior in mixed traffic,” *SAE Int. J. of CAV*, vol. 5, no. 4, 2022.