

UC Berkeley

UC Berkeley Electronic Theses and Dissertations

Title

On Planning and Control for Multi-Agent Robotic Systems

Permalink

<https://escholarship.org/uc/item/4r520469>

Author

Wu, Fangyu

Publication Date

2024

Peer reviewed|Thesis/dissertation

On Planning and Control for Multi-Agent Robotic Systems

By

Fangyu Wu

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering - Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Alexandre Bayen, Chair

Professor Trevor Darrell

Assistant Professor Maria Laura Delle Monache

Summer 2024

On Planning and Control for Multi-Agent Robotic Systems

Copyright 2024
by
Fangyu Wu

Abstract

On Planning and Control for Multi-Agent Robotic Systems

by

Fangyu Wu

Doctor of Philosophy in Engineering - Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Alexandre Bayen, Chair

This dissertation addresses the complexities of planning and control in Multi-Agent Systems (MAS), focusing on optimizing interactions within three configurations: 1) sequentially chained MASs, 2) partially meshed MASs, and 3) centrally controlled MASs. As autonomous technologies increasingly permeate sectors ranging from automotive to industrial applications, the need for robust multi-agent systems capable of efficient coordination among themselves and with human operators becomes paramount. This research provides a comprehensive analysis and development of methodologies that enhance the reliability, efficiency, and safety of these systems through advanced planning and control strategies in the context of connected and automated driving.

The primary objective of this study is to refine the planning and control frameworks used in MAS by leveraging recent advancements in vehicle automation, optimization, and machine learning. By integrating these fields, the dissertation introduces novel solutions that serve real-world applications such as automated car-following, decentralized collision avoidance, and scalable multi-agent planning. The work addresses critical challenges in these systems, including stop-and-go waves, collision avoidance in understructured environments, scalable network-agnostic multi-agent planning, the efficient, stable integration of learning into control frameworks, and hardware and experimental design for MASs.

The main contributions of this research include the development of novel algorithms that reduce error propagation in sequentially chained MAS configurations, both with and without nonlocal measurements, decentralized conflict resolution methods for partially meshed MASs, and scalable scheduling techniques for centrally controlled MASs. These methodologies are validated through a series of simulations and physical experiments, demonstrating their effectiveness and practical applicability.

Moreover, this dissertation explores the integration of machine learning with traditional control frameworks and reflects on the hardware and experimental design utilized in this

study. First, we present the memory-augmented model predictive control method, in which empirical data enhance the amortized running time performance of model predictive control models without compromising stability. Next, we discuss the hardware and experimental design employed in this dissertation to document the lessons learned from empirical hands-on applications.

In summary, this dissertation establishes a theoretical and experimental foundation for the next generation of MAS research in the context of connected and automated driving, aiming at the effective and safe integration of these systems into the fabric of daily and industrial applications.

献给我的母亲，姐姐，以及猫

Contents

Contents	ii
List of Figures	v
List of Tables	xi
1 Introduction	1
1.1 Motivation	1
1.2 Terminology	2
1.3 Problem Definition	5
1.4 Related Works	6
1.5 Summary of Contributions	7
1.6 Organization of the Dissertation	9
2 Sequentially Chained Multi-Agent Systems with Local Observation	10
2.1 Introduction	10
2.2 Related Works	11
2.3 Attenuative Kerner's Model	12
2.4 String Stability Analysis	14
2.5 Numerical Simulation	16
2.5.1 Simulation Setup	16
2.5.2 Numerical Results	18
2.5.3 Energy Efficiency	19
2.5.4 Robustness to Disturbances	21
2.6 Physical Experiments	22
2.6.1 Experimental Setup	22
2.6.2 Experimental Results	23
2.6.3 Energy Efficiency	24
2.7 Discussion	25
2.7.1 Theoretical Remarks	25
2.7.2 Practical Considerations	26

3	Sequentially Chained Multi-Agent Systems with Non-Local Observation	28
3.1	Introduction	28
3.2	Related Works	29
3.3	Problem Statement	30
3.4	Optimal Car-Following Controller	31
3.4.1	Prediction Layer	31
3.4.2	Planning Layer	32
3.4.3	Tracking Layer	36
3.5	Numerical Simulation	37
3.5.1	Simulation Setup	37
3.5.2	Numerical Results	39
3.6	Discussion	39
3.6.1	Headway Constraints Design	39
3.6.2	ETA Estimator Choice	40
4	Partially Meshed Multi-Agent Systems	41
4.1	Introduction	41
4.2	Related Works	43
4.3	Problem Statement	44
4.4	Consensus-Based Conflict Resolution	45
4.4.1	Least-Action Consensus	46
4.4.2	Traversal Policy and Cost Functional	47
4.4.3	Three Phases of Conflict Resolution	48
4.5	Berkeley DeepDrive Drone Dataset	48
4.5.1	Processed Videos	49
4.5.2	Annotated Images	51
4.5.3	Development Kit	54
4.5.4	Areas of Applications	55
4.6	Numerical Simulation	56
4.6.1	Simulation Setup	56
4.6.2	Numerical Results	58
4.7	Discussion	60
5	Centrally Controlled Multi-Agent Systems	62
5.1	Introduction	62
5.2	Related Works	63
5.3	Problem Statement	64
5.4	Mixed-Integer Scheduling Method	65
5.4.1	Optimization Model Design	65
5.4.2	A Two-Agent Example	66
5.5	A Reservation-Based Heuristic	68
5.5.1	Formulation of the Heuristic	68

5.5.2	A Four-Agent Example	69
5.6	Numerical Simulation	70
5.6.1	Simulation Setup	71
5.6.2	Numerical Results	75
5.7	Discussion	77
5.7.1	Nonlocality of Multi-Agent Interactions	77
5.7.2	Robust Real-Time Deployment	77
5.7.3	Extension for Lane-Changing	79
6	Learning-Augmented Planning and Control	80
6.1	Introduction	80
6.2	Related Works	81
6.3	Composing MPC, LQR, and NN	83
6.3.1	Basics of MPC	83
6.3.2	Composition Rules	85
6.4	Theoretical Analysis	90
6.4.1	Stability of MAMPC	90
6.4.2	Remark on Robustness	93
6.4.3	Remark on Necessity of Fail-Safe MPC	94
6.5	Numerical Experiments	95
6.5.1	Pendulum	96
6.5.2	Triple Pendulum	97
6.5.3	Bicopter	97
6.5.4	Quadcopter	99
6.5.5	Robustness Margin	101
6.6	Discussion	106
7	Devices and Experiments for Multi-Agent Research	109
7.1	Actuation Devices and Experiments	109
7.1.1	Unicycle Robot	110
7.1.2	Automated Vehicle	112
7.2	Estimation Devices and Experiments	116
7.2.1	Indoor Object Localization	117
7.2.2	Aerial Video Collection	119
8	Conclusion and Future Work	123
8.1	Chapter Summary	123
8.2	Future Directions	124
8.3	Concluding Remarks	125
	Bibliography	126

List of Figures

1.1	An illustrative example of sequentially chained MAS.	4
1.2	An illustrative example of partially meshed MAS.	4
1.3	An illustrative example of centrally controlled MAS.	5
2.1	Illustration of the two-vehicle car-following problem. The ego vehicle needs to follow the lead vehicle as smoothly as safety permits.	13
2.2	Speed compensation and phase portrait of the AKM with parameters in Table 2.1.	14
2.3	Block diagram of a typical vehicle system integrated with the AKM controller.	15
2.4	Simulated trajectories of the AKM, human driver, and ACC.	20
2.5	Simulated phase portraits of the AKM, human driver, and ACC. The dashed lines in (a) indicate phase motion of different constant time headways.	21
2.6	Evaluation of the impacts of disturbances on AKM. The qualitative efficiency ranking of these models remain unchanged.	22
2.7	An angled view and a trunk view of the automated Cadillac XT5.	23
2.8	Satellite image of the experiment site in Richmond, CA, USA.	23
2.9	Measured trajectories of the AKM, human driver, and ACC.	24
2.10	Measured phase portraits of the AKM, human driver, and ACC.	25
3.1	Scenario of the two-vehicle car-following problem. At t_0 , the considered vehicle is at w_0 and the preceding vehicle at w_1 . The preceding vehicle travels through a set of way points $(w_1, w_2, \dots, w_f)$, where each pair of consecutive way points are spaced Δs apart, barring boundary condition.	30
3.2	Diagram of the MPC controller design. Prediction layer and planning layer are invoked every 1 second. Tracking layer is triggered every 0.1 second.	31
3.3	Specification of the receding-horizon control scheme. The prediction layer has a <i>fixed</i> spatial horizon and a <i>variable</i> temporal horizon. The planning layer has a temporal horizon m . The tracking layer has a fixed temporal horizon n	32
3.4	Headway envelope design. $\hat{s}^0(\cdot)$ is the predicted trajectory of the proceeding vehicle. $\hat{s}_s^-(\cdot), \hat{s}_s^+(\cdot)$ are estimated minimum and maximum <i>space</i> headway envelopes, respectively. $\hat{s}_t^-(\cdot), \hat{s}_t^+(\cdot)$ are estimated minimum and maximum <i>time</i> headway envelopes, respectively. The shaded region represents the feasible set in space and time.	33

3.5	Headway envelope design to handle cut-ins and cut-outs. Left design is for cut-ins and right for cut-outs. The proposed designs aim to to keep a comfortable headway Δs^* without overreaction.	35
3.6	Demonstration of the planning layer. The oracle controller assumes perfect prediction over the entire planning horizon.	38
3.7	Performance heat maps of the MPC controllers. The top heat map measures tracking error e , while the bottom heat map measures fuel saving f	38
4.1	File structure of the B3D dataset. Informational files METADATA, LICENSE, and README are omitted for concision.	50
4.2	Tailgating collisions in hwy00.mp4 and hwy01.mp4. The collided vehicles are circled in black. The first accident involves at least two vehicles, while the second incident involves four vehicles.	51
4.3	Typical Road Topology from the B3D dataset. In alphabetic order, we have (a)-(d) unsignalized intersections, (e)-(g) unsignalized roundabouts, (h)-(i) highways, and (i)-(k) merging bottlenecks. (a)-(d) are roughly on the same scale. (e)-(g) are on another identical scale. (h)-(i) are approximately on a third identical scale. Lastly, (j)-(k) are on a fourth distinct scale.	52
4.4	Timeline of the <i>tailgating accidents</i> . Green indicates regular traffic. Red indicates congestion caused by a collision. Light red indicates the induced congestion starts dissipating.	53
4.5	Timeline of the <i>stop-and-go waves</i> . Green indicates regular traffic. Red indicates congestion caused by a strong stop-and-go wave. Light red indicates congestion caused by a weak stop-and-go wave.	53
4.6	An example annotated image. Colored rectangles are axially aligned bounding boxes of the vehicles in the scene.	53
4.7	An example image with detection results. Colored rectangles are axially aligned bounding boxes. Black numbers are confidence scores of the estimated bounding boxes, with 1.0 being full certainty and 0.0 being full uncertainty.	54
4.8	A demonstration of the masking process. The red polygonal mask is defined via CVAT. Pixels outside of the mask are considered irrelevant and are marked as black.	55
4.9	Visual ambiguity of low resolution objects. (a)(b) are visually alike. (c)(d) are also visually similar.	56
4.10	Road topology of the extracted two-agent scenario from video “jnc07.mp4.” The intersection is unsignalized, where southbound traffic and westbound traffic are in conflict at the crossing.	57
4.11	Example path pairs used for validating consensus model. Red circles indicate the origins of traffic while black circles denote the respective destinations. The blue node at the center marks the only conflicting point at the cross intersection. Green rectangles, labeled with numbers, represent the vehicles as they pass through the intersection. The initial speeds are displayed next to the respective origins. . . .	57

4.12	Road topology of the constructed N -agent simulation scenario. Red circles indicate the origins of traffic while black circles denote the respective destinations. The blue node at the center marks the only conflicting point at the cross intersection. Green rectangles, labeled with numbers, represent the vehicles as they pass through the intersection.	58
4.13	Visualization of the entry and crossing times of vehicles approaching from the southbound and westbound directions, respectively. Each red point represents a southbound vehicle, while each blue point represents a westbound vehicle. It is important to note that although vehicles enter the intersection randomly, they cross the intersection in organized batches, forming queues of alternating directions.	59
5.1	The two-agent setup at a crossing intersection. Each path is discretized into waypoints, at which arrival and departure times are computed through optimization.	67
5.2	The reservation-based heuristic applied to the two-agent setup. The reservation table $\mathcal{R}$ caches existing solutions, which are then used as constraints in subsequent optimizations. This approach allows the mixed-integer method to be scalable in real-time applications.	69
5.3	Visualization of the four-way cross intersection scenario. The red rectangles represent the origins of vehicles in the respective parking lots. Path conflicts primarily happen at the intersection.	71
5.4	Visualization of the four-way roundabout intersection scenario. The red rectangles represent the origins of vehicles in the respective parking lots. Path conflicts primarily happen at roundabout merge points.	72
5.5	Visualization of the single-lane thruway with onramps and offramps scenario. The red rectangles represent the origins of vehicles in the respective parking lots. Path conflicts primarily happen at on-ramp merge points.	72
5.6	Visualization of the 2×2 intersection grid scenario. The red rectangles represent the origins of vehicles in the respective parking lots. Path conflicts primarily happen at 4-way cross intersections and roundabout merge points.	73
5.7	Visualization of the three-way cross intersection mesh scenario. The red rectangles represent the origins of vehicles in the respective parking lots. Path conflicts primarily happen at 3-way cross intersections.	74
5.8	Visualization of the concentric hexagonal rings scenario. The red rectangles represent the origins of vehicles in the respective parking lots. Path conflicts primarily happen at 3-way and 4-way cross intersections.	74
5.9	Visualization of the 8×8 intersection grid scenario. The red rectangles represent the origins of vehicles in the respective parking lots. Path conflicts primarily happen at 4-way cross intersections.	75

5.10	Example snapshots of running simulation of the cross intersection, roundabout intersection, and thruway with onramps and offramps scenarios, optimized by the MILP algorithm. Red rectangles represent vehicles parking at the original parking lots. Blue rectangles represent vehicles travelling between respective origins and destinations. Green rectangles represent vehicles parking at the destination parking lots.	76
6.1	Schematics of the learning processes. (a) Imitation phase: $(\mathbf{x}, u_{\text{MPC}}(\mathbf{x}))$ provides data to train the NN controller u_{NN} in a supervised learning setting. (b) Adaptation phase (optional): u_{NN} interacts with the environment $f(\cdot, \cdot)$ to enhance control effectiveness in a reinforcement learning setting.	86
6.2	An illustrative example of how different variants of MAMPC <i>could</i> steer a system toward equilibrium. (a) Standard MAMPC: a triple-mode hybrid control composed of an MPC, a NN, and a LQR. (b) Alternating-authority MAMPC: a variant of the standard MAMPC, modified for chaotic systems. (c) Way-point MAMPC: a variant of the standard MAMPC, modified for slow systems. Solid red regions represent $\mathbb{X}_0$; solid green regions represent $\mathcal{R}_{\text{NN}}$; <i>green dots</i> indicate periodic controller switching region in alternating-authority MAMPC; and solid blue regions represent $\mathcal{R}_{\text{LQR}}$. A black dot represents a state and an arrow represents a controlled state transition. The color of arrows indicates which sub-controller is invoked during that control step: red is MPC; green is NN; and blue is LQR. The yellow line represents $\partial\mathcal{D}_{\text{WP}}$. Note that the above examples by no means exhaustive. For instance, it is possible that $\mathcal{R}_{\text{NN}} = \emptyset$	89
6.3	Diagram of a pendulum model. A torque between $[-0.05, 0.05]$ N · m is being applied to the hinge to keep the pendulum at its inverted position.	96
6.4	Application of standard MAMPC to the pendulum problem. τ is running time. $\mathbf{x}$ is selected states and $\mathbf{u}$ is selected inputs. t is simulation time. At initialization, the NN mode fails forward verification and thus never get invoked. As the NN is being trained, its likelihood of passing forward verification increases. At last, a well trained NN takes over most of MPC steps, resulting in a more efficient operation.	98
6.5	Diagram of a triple pendulum model. Note that the centers of mass of the first and second torques coincide with the second and third joints. A torque between $[-1, 1]$ N · m is being applied to each joint to keep the triple pendulum at its inverted position.	99
6.6	Application of alternating-authority MAMPC to the triple pendulum problem. τ is running time. $\mathbf{x}$ is selected states and $\mathbf{u}$ is selected inputs. t is simulation time. At initialization, the NN mode fails forward verification and thus never get invoked. As the NN is being trained, its likelihood of passing forward verification increases. At last, a well trained NN takes over most of MPC steps, resulting in a more efficient operation.	100

6.7	Diagram of a bicopter model. Each of the two propellers is capable of generating a thrust between 0.1 N and 9.1572 N to keep the copter hover in air. The thrust limits are chosen such that hovering thrusts are approximately in the center of the thrust limit.	101
6.8	Application of standard MAMPC to the bicopter problem. τ is running time. $\mathbf{x}$ is selected states and $\mathbf{u}$ is selected inputs. t is simulation time. At initialization, the NN mode fails forward verification and thus never get invoked. As the NN is being trained, its likelihood of passing forward verification increases. At last, a well trained NN takes over most of MPC steps, resulting in a more efficient operation.	102
6.9	Diagram of a quadcopter model. Each of the four propellers is capable of rotating at speed between $0 \text{ rad} \cdot \text{s}^{-1}$ and $313.96 \text{ rad} \cdot \text{s}^{-1}$ to keep the copter hover in air. The rotor limits are chosen such that hovering rotor rotations are approximately in the center of the rotation range.	103
6.10	Application of way-point MAMPC to the quadcopter problem. τ is running time. $\mathbf{x}$ is selected states and $\mathbf{u}$ is selected inputs. t is simulation time. At initialization, the NN mode fails forward verification and thus never get invoked. As the NN is being trained, its likelihood of passing forward verification increases. At last, a well trained NN takes over most of MPC steps, resulting in a more efficient operation.	104
6.11	Illustration of robustness margin. The <i>true</i> system $f(\cdot)$ enters $\mathcal{R}_{\text{LQR}}$ when the <i>modeled</i> system $\bar{f}(\cdot)$ enters $\mathcal{R}_{\text{LQR}}^\delta$. Here, δ maybe viewed as a notion of robustness margin.	105
7.1	A schematic diagram of a typical unicycle robot.	110
7.2	Prototype unicycle robot customized from mBot.	111
7.3	Block diagram demonstrating our recent hardware design, which features a time headway follower stopper (THFS) as the safety filter and a PID controller with output saturation as the velocity controller.	112
7.4	Illustration of the ACC bypass system. The OEM ACC module is disabled and replaced with the custom control module at each startup.	114
7.5	Block diagram showing the technical components of the systems developed in the MVT project.	115
7.6	A view of the parking lot facility during a test day of the MVT.	116
7.7	The I24 testbed near Nashville, TN.	117
7.8	Examples of fiducial markers for indoor localization.	118
7.9	Localization based on spatial triangulation.	119
7.10	A view of the single-camera indoor localization system using color thresholding.	119
7.11	An illustration of the experiment setup. The quadcopter hovers over the road of interest, recording the top-view of each traffic participant as they enter and exit the scene.	120
7.12	The computer vision pipeline deployed for producing the B3D dataset.	121

7.13 A typical detection then tracking approach.	121
7.14 Using image segmentation methods to convert an axis-aligned bounding box to an oriented bounding box.	122

List of Tables

2.1	Parameters of the Attenuative Kerner's Model	17
2.2	short	25
4.1	Dates, Lengths, and Locations of Videos	49
4.2	Configuration and Results of Two-Agent Simulation Tests	59
5.1	Trip Count and Fulfillment Time of the Simulated Scenarios	78
5.2	Problem Sizes and Running Times of the Simulated Scenarios	78
6.1	Neural Networks Architecture	103
6.2	Summary of Total Running Time in Millisecond	105
6.3	Summary of Per-Step Running Time in Mircosecond	105
7.1	Parameters of vehicle system dynamics.	114

Acknowledgments

The research presented in this dissertation was made possible by the unwavering support and invaluable contributions of many individuals during my time at Berkeley. I am deeply grateful to each and every one of you for being part of this journey and enriching my experience. Thank you for being part of this journey.

First and foremost, I extend my heartfelt thanks to my advisor, Professor Alexandre Bayen. His vision directed my PhD research, his boundless energy and intellectual depth inspired me, and his support facilitated numerous collaborations. He made the entire experience immensely enjoyable by promoting a healthy work-life balance. I am deeply grateful for his professional and personal guidance, support, and camaraderie.

I am also thankful to my committee members, Professor Trevor Darrell, Professor Maria Laura Delle Monache, and Professor Yi Ma. Special thanks go to Professor Ronald Fearing and Professor Claire Tomline for chairing my qualifying exam committee and for the enriching discussions on various topics in linear systems theory.

I was fortunate to collaborate with Professor Ion Stoica and his PhD students, Guanhua Wang and Siyuan Zhuang, on amortized optimal control. My gratitude extends to Professor Trevor Darrell and his student, Dequan Wang, for the chance to work on the B3D project and explore the fascinating problem of decentralized trajectory planning. Additionally, I am thankful to Professor Shankar Sastry for inviting me to his weekly group meetings, where I learned extensively from his group's research.

My understanding of control theory, embedded systems, and humanoid robotics owes much to Professor Kameshwar Poolla, Professor Ronald Fearing, Professor Murat Arcak, and Professor Koushil Sreenath. Their teaching deeply influenced my research during my PhD.

I take this special opportunity to wholeheartedly thank Professor Daniel Work at Vanderbilt University, my first research mentor during my undergraduate and master's years at UIUC. His mentorship and guidance during those formative years were invaluable, and I deeply appreciate his continued support.

I am also grateful to Professor Jonathan Sprinkle from Vanderbilt University, Professor Benedetto Piccoli from Rutgers University, and Professor Benjamin Seibold from Temple University for their support during the CIRCLES project.

Throughout my PhD, I had the privilege to collaborate with numerous industry partners. I extend my gratitude to Sharon Hornstein and James Lawrence from General Motors for their continuous support in the CIRCLES project. Their assistance was crucial for the success of our collaboration with General Motors.

During my early years at Berkeley, I was fortunate enough to meet and learn from Alexander Keimer, an extraordinary mentor and a true friend. His methodical and rigorous approach to research, coupled with his work ethic and remarkable humility, are qualities I aspire to embody. Working with Alex has been not only immensely productive but also thoroughly enjoyable.

I have also had the pleasure of collaborating with many PhD and postdoc students. A big thanks to Nathan Lichtlé for the memorable discussions on reinforcement learning, robotics, and control theory, and for all the movies, games, and Reddit memes. Thanks to Joy Carpio for her dedication, patience, and work ethic in the CIRCLES project, and to Arwa AlAnqary for the stimulating discussions on optimal control and game theory. Special thanks to Rahul Bhadani, Matthew Bunting, and Matthew Nice for their insights on the hardware aspects of autonomous driving.

I am also thankful to Saleh Albeaik, Sulaiman Almatrudi, Alben Bagabaldo, Théophile Cabannes, Shaoru Chen, Fang-Chieh Chou, Zhe Fu, Yashar Farid, Kathy Jang, Aboudy Kreidieh, Jessica Lazarus, Jonathan Lee, Yiling You, Cathy Wu, Han Wang, Shuxia Tang, Eugene Vinitsky, Xiaoyun Zhao, and Hossein Zinat for their help and insights.

I would like to acknowledge the interns, master's, and undergraduate researchers I had the pleasure of mentoring and collaborating with during my PhD. Working with Fahd Althukair, Xiangyun Cao, Avikam Chauhan, Chenhui Hao, Minjune Hwang, Jiayi Li, Jiawei Lu, Kehan Wang, Lu Yu, Xiyu Zhai, Changxu Zhang, and Yichen Zhou has been exceptionally rewarding. Special thanks to Jiamu Zhang for the long-standing collaboration that contributed to the final chapters of this dissertation.

Genuine thanks to Helen Bassham, Shirley Salanio, Laman Sadaghiani, and Annie Yeh for their assistance with the administrative and logistical aspects of my PhD.

The hardware and experiment aspect of the research described in this dissertation would not have been possible without the contributions of the staff at the Berkeley Institute of Transportation Studies (ITS), California Partners for the Advanced Transportation Technology (PATH), Berkeley DeepDrive (BDD), and Berkeley Artificial Intelligence Research (BAIR).

Thanks to Jiancong Chen, Jingzhou Zhang, Tianjiao Pu, Wenjing Wu, and Zaiqin Jiang for the memorable times at Berkeley. Special thanks to Xiaoyun Zhao for being an amazing comrade during my first year at Berkeley, and to Han Wang, Tyler Truksa, and Sean McQuade for the fun times playing Age of Empires 4 and Starcraft 2.

Finally, I would like to thank my parents, Peipei, and Soda. Your love and support made this journey possible. Thank you for everything.

Chapter 1

Introduction

1.1 Motivation

The task of *planning* and *controlling* the movement of multiple agents from their points of origin to their respective destinations has long been a fundamental problem in our society. Among its many instances, an ancient multi-agent planning and control task is the transportation problem. During the era of Julius Caesar, regulations were already in place to ban private vehicles from the streets of Rome during the daytime to alleviate congestion [125]. In seventeenth-century Belgium, early forms of modern traffic rules, such as speed limits and intersection right-of-way, as well as the presence of traffic police, began to emerge [103]. With the advent of the automobile, the efficient automation of road coordination became essential, leading to the installation of the first electric traffic lights in Cleveland, USA in 1914 [108]. Similar developments can be observed in railway, water, and air transportation, paralleling those seen in vehicle-based ground transportation.

Reflecting on our history, we can identify two fundamental forces behind the development of multi-agent planning and control methods: 1) the growth in risk and complexity, and 2) the progress in theory and technology. For instance, the mass adoption of automobiles in the early nineteenth century made the public roads significantly more dangerous and harder to regulate via traditional means. This increased risk and complexity in ground transportation necessitated a new approach to traffic regulation. In response to this challenge, decades of experience in traffic management, combined with the newly developed electricity grid, provided a viable solution: electricity-powered automated traffic lights.

In contemporary times, the problem of multi-agent planning and control has emerged in numerous new areas that were unimaginable even a few decades ago. For example, connected and automated vehicles should interact with human drivers to enhance traffic throughput and driving efficiency. Logistics robots must operate efficiently and safely in relation to one another and to nearby human operators within a controlled warehouse environment. Humanoid robots require rapid and precise coordination among their limbs to manipulate objects, navigate physical spaces, and convey intent in social contexts. Additionally, swarms

of drones need to be intricately and gracefully coordinated to perform artistic formations in nighttime drone shows.

These new multi-agent challenges are more complex than traditional transportation problems and can pose significant risks to society if not properly addressed. For instance, early commercial adaptive cruise control (ACC) systems were designed without considering multi-agent interactions [38]. An unintended consequence of this design approach is a phenomenon known as string instability [48, 119], where ACC systems amplify the tracking errors between the velocities of lead vehicles and their reference velocities. When multiple ACC-controlled vehicles form a platoon behind a lead vehicle, this error amplification multiplies, resulting in increasingly aggressive car-following behaviors, which are fuel inefficient and potentially unsafe.

At the same time, we have witnessed remarkable recent progress in automation, computing, and artificial intelligence (AI). On one hand, existing mathematical approaches, including control theory, convex optimization, and integer optimization, are becoming increasingly powerful due to the growing capabilities of sensors, actuators, and computing and networking infrastructure. On the other hand, cutting-edge developments in machine learning, such as reinforcement learning (RL), large language models (LLM), and diffusion models, have also unlocked numerous new avenues to synergize the empirical performance of neural networks with the rigor of mathematical models.

The motivation of this dissertation is to explore how the latest developments in theory and technology can be integrated to provide practical or paradigm-shifting solutions to a significant subset of these new problems, with an emphasis on transportation problems.

1.2 Terminology

We begin by informally describing a general definition of a multi-agent system (MAS) and the class of MAS problems we aim to address in this dissertation. Although a MAS can be analyzed through various technical fields, we adopt the language and general framework of optimal control for the remainder of the thesis. Generally speaking, a MAS problem consists of two foundational components: 1) the control system and 2) the cost functional. These two components are defined below.

In this thesis, we refer to a CONTROL SYSTEM as a set of ordinary differential equations (ODEs) of the following form:

$$\begin{aligned}\dot{\mathbf{x}}(t) &= f(t, \mathbf{x}(t), \mathbf{u}(t)), & \mathbf{x}(t_0) &= \mathbf{x}_0, \\ \mathbf{y}(t) &= h(\mathbf{x}(t)),\end{aligned}\tag{1.1}$$

where $\mathbf{x}(t) : \mathbb{R} \rightarrow \mathbb{R}^n$ represents the state, $\mathbf{u}(t) : \mathbb{R} \rightarrow \mathbb{R}^p$ the control input, $\mathbf{y}(t) : \mathbb{R} \rightarrow \mathbb{R}^q$ the output, $t \in \mathbb{R}$ denotes the time, t_0 the initial time, and $\mathbf{x}_0$ the initial state. For the remainder of the dissertation, we may simplify notation by omitting the t arguments and using $\mathbf{x}$, $\mathbf{u}$, and $\mathbf{y}$ for conciseness, provided there is no ambiguity.

Assuming well-posedness of the ODE system outlined above, the system can be fully parametrized by its control input and initial conditions. That is, given a control input $\mathbf{u}(t)$ from t_0 to a specified terminal time t_f and an initial condition $(t_0, \mathbf{x}_0)$, the system state trajectory $\mathbf{x}(t)$ can be fully determined. Therefore, we can associate a COST FUNCTIONAL J with each combination of control input $\mathbf{u}$ and initial condition, as defined below:

$$J(\mathbf{u}; t_0, \mathbf{x}_0, t_f) = \int_{t_0}^{t_f} \ell(t, \mathbf{x}, \mathbf{u}) dt + \ell_f(t_f, \mathbf{x}_f), \quad (1.2)$$

where $\ell(\cdot) : \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^p \rightarrow \mathbb{R}$ is the stage cost or running cost, $\ell_f(\cdot) : \mathbb{R} \times \mathbb{R}^n \rightarrow \mathbb{R}$ the terminal cost, and $\mathbf{x}_f := \mathbf{x}(t_f)$ the terminal state. Here, $J(\cdot)$ is a function of the control function $\mathbf{u}(t)$, hence the term *functional*. Additionally, while the terminal time t_f may be free, within the scope of this thesis, we will consider it fixed.

A MULTI-AGENT SYSTEM (MAS), as referred to in this thesis, is a special subclass of the general control system defined in (1.2), where the whole system can be subdivided into relatively independent subsystems. Each subsystem functions as the control system for an agent within the environment. Concretely, a system comprising N agents can be specified as:

$$\begin{aligned} \dot{\mathbf{x}}_1(t) &= f_1(t, \mathbf{x}_1(t), \mathbf{u}_1(t)), & \mathbf{x}_1(t_0) &= \mathbf{x}_{1|0}, \\ \dot{\mathbf{x}}_2(t) &= f_2(t, \mathbf{x}_2(t), \mathbf{u}_2(t)), & \mathbf{x}_2(t_0) &= \mathbf{x}_{2|0}, \\ &\vdots & \\ \dot{\mathbf{x}}_N(t) &= f_N(t, \mathbf{x}_N(t), \mathbf{u}_N(t)), & \mathbf{x}_N(t_0) &= \mathbf{x}_{N|0}, \\ \mathbf{y}_1(t) &= h_1(\mathbf{x}_1(t), \mathbf{x}_2(t), \dots, \mathbf{x}_N(t)), \\ \mathbf{y}_2(t) &= h_2(\mathbf{x}_2(t), \mathbf{x}_2(t), \dots, \mathbf{x}_N(t)), \\ &\vdots & \\ \mathbf{y}_N(t) &= h_N(\mathbf{x}_1(t), \mathbf{x}_2(t), \dots, \mathbf{x}_N(t)), \end{aligned} \quad (1.3)$$

where the n th subscript denotes the n th subsystem, $\mathbf{x} := [\mathbf{x}_1^\top, \mathbf{x}_2^\top, \dots, \mathbf{x}_N^\top]^\top$, and $\mathbf{y} := [\mathbf{y}_1^\top, \mathbf{y}_2^\top, \dots, \mathbf{y}_N^\top]^\top$. A MULTI-AGENT ROBOTIC SYSTEM (MARS) is a term used herein to describe a MAS where each agent's state captures essential mechanical or electronic properties of a robotic device.

When constructing control inputs by feedbacking the output, i.e., $\mathbf{u}(t) = g(\mathbf{y}(t)) = g \circ h(\mathbf{x}(t))$, the effectiveness of such feedback control depends on the extent of access it has to the full state. Therefore, it is particularly useful to categorize problems based on the structure of the observation function h , especially in the context of MASs.

Perhaps the two most trivial types of observation functions are the identity function and the null function, i.e., $h(\mathbf{x}) = \mathbf{x}$ and $h(\mathbf{x}) = \mathbf{0}$, respectively. In the context of MASs, we define a system as *fully meshed* if $h_i(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N) = [\mathbf{x}_1^\top, \mathbf{x}_2^\top, \dots, \mathbf{x}_N^\top]^\top$ for each $i = 1, 2, \dots, n$. Conversely, we term a system *strictly local* if $h_i(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N) = \mathbf{x}_i$ for each $i = 1, 2, \dots, n$.

In real-world MASs, three types of observation functions are widely adopted: 1) sequentially chained MASs, 2) partially meshed MASs, and 3) centrally controlled MASs.

A SEQUENTIALLY CHAINED MULTI-AGENT SYSTEMS is a special type of locally meshed MAS with a observation function of the following form:

$$\begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \mathbf{y}_3 \\ \vdots \\ \mathbf{y}_N \end{bmatrix} = \begin{bmatrix} \mathbf{I}_{n_1} \\ \hline \mathbf{I}_{n_1} & \mathbf{I}_{n_2} \\ \hline & \mathbf{I}_{n_2} & \mathbf{I}_{n_3} \\ \hline & & \vdots \\ \hline & & & \mathbf{I}_{n_{N-1}} & \mathbf{I}_{n_N} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \vdots \\ \mathbf{x}_N \end{bmatrix}, \quad (1.4)$$

where, except for the first agent, each agent in the system has access to the states of both itself and its preceding agent. Information exchange occurs in a directed line, as illustrated in Figure 1.1, resembling a chain of daisies. Therefore, this system is also referred to as a “daisy-chained” system.

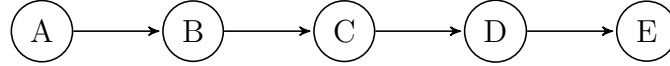


Figure 1.1: An illustrative example of sequentially chained MAS.

A general PARTIALLY MESHED MULTI-AGENT SYSTEM is a MAS that is not fully meshed, where information exchange typically occurs only locally and heterarchically, as illustrated in Figure 1.2.

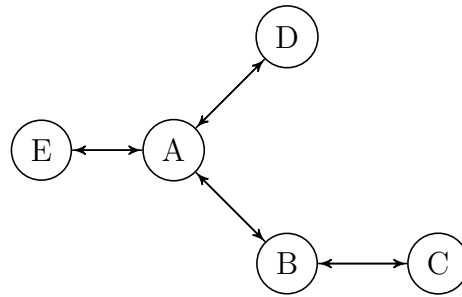


Figure 1.2: An illustrative example of partially meshed MAS.

A CENTRALLY CONTROLLED MULTI-AGENT SYSTEM is a MAS with an observation func-

tion of the following form:

$$\begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_N \end{bmatrix} = \begin{bmatrix} \mathbf{I}_{n_1} & & & \\ & \mathbf{I}_{n_2} & & \\ & & \ddots & \\ & & & \mathbf{I}_{n_N} \\ \hline \mathbf{I}_{n_1} & \mathbf{I}_{n_2} & & \\ & & \ddots & \\ \hline \mathbf{I}_{n_1} & & \dots & \mathbf{I}_{n_N} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_N \end{bmatrix}, \quad (1.5)$$

where the first agent has access to the full state information, whereas the other agents only have access to their own state and that of the first agent. Here, the structure of information exchange resembles a star, as illustrated in Figure 1.3, hence the term “star-connected” system is also commonly used.

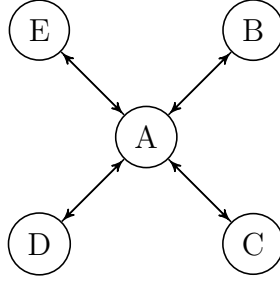


Figure 1.3: An illustrative example of centrally controlled MAS.

1.3 Problem Definition

In this dissertation, we address the problems of planning and control within the context of MASs, the definitions of which are presented in detail below.

A **PLANNING PROBLEM** involves determining a *feedforward* control input $\mathbf{u}$ that navigates a system from its initial condition $(t_0, \mathbf{x}_0)$ to a specified terminal set X_f , while adhering to path constraints $\mathbb{X}$ and input constraints $\mathbb{U}$. When multiple unique control inputs can solve the planning problem, the objective is to identify the one(s) that minimize the cost functional, denoted by $\mathbf{u}^* := \min_{\mathbf{u}} J(\mathbf{u}; t_0, \mathbf{x}_0)$. The optimal trajectory produced by $\mathbf{u}^*$ is represented as $\mathbf{x}^*$. Formally, we state the planning problem as the following optimization problem:

Problem 1. (Optimization-Based Planning Problem) Provided $t_0, \mathbf{x}_0, t_f$, and X_f solve

$$\begin{aligned} \mathbf{u}^*(t) &:= \min_{\mathbf{u}(t)} J(\mathbf{u}; t_0, \mathbf{x}_0, t_f) \\ \text{s.t. } \quad &\mathbf{x}(t_0) = \mathbf{x}_0, \quad \mathbf{x}(t_f) \in X_f, \\ &\dot{\mathbf{x}}(t) = f(t, \mathbf{x}(t), \mathbf{u}(t)), \quad t \in [t_0, t_f], \\ &(\mathbf{x}(t), \mathbf{u}(t)) \in \mathbb{X} \times \mathbb{U}, \quad t \in [t_0, t_f]. \end{aligned} \tag{1.6}$$

A CONTROL PROBLEM involves determining a *feedback* control input $\mathbf{u}$ that accurately tracks either a static or dynamic reference trajectory $\mathbf{x}^{\text{ref}}$, typically the $\mathbf{x}^*$ derived from a corresponding planning problem, while adhering to the same set of path and input constraints as the planning problem. This control input can be implemented using straightforward PID control or a more complex model predictive control (MPC), with a formulation similar to Problem (1.6), as follows:

Problem 2. (Optimization-Based Control Problem) Provided $\mathbf{x}^{\text{ref}}$ and T , for every time τ with $\mathbf{x}(\tau) = \mathbf{x}_\tau$, solve

$$\begin{aligned} \mathbf{u}^*(t) &:= \min_{\mathbf{u}(t)} \int_{\tau}^{\tau+T} \|\mathbf{x}^{\text{ref}}(t) - \mathbf{x}(t)\| dt \\ \text{s.t. } \quad &\mathbf{x}(\tau) = \mathbf{x}_\tau, \\ &\dot{\mathbf{x}}(t) = f(t, \mathbf{x}(t), \mathbf{u}(t)), \quad t \in [\tau, \tau + T], \\ &(\mathbf{x}(t), \mathbf{u}(t)) \in \mathbb{X} \times \mathbb{U}, \quad t \in [\tau, \tau + T]. \end{aligned} \tag{1.7}$$

and let the input control be $\mathbf{u}(\tau) = \mathbf{u}^*(\tau)$. Repeat the process each time step until $\mathbf{x} \in X_f$.

1.4 Related Works

Research on MASs has its origins tracing back to the 1980s, emerging from engineering fields such as robotics, computer networks, transportation, and electrical grids, and intersecting with applied mathematics disciplines like control theory, optimization, game theory, and graph theory [29]. This interdisciplinary foundation has set the stage for developments in the field for the upcoming decades.

In the 1990s, Dudek significantly advanced the conceptualization of MASs by proposing a classification system based on seven dimensions: 1) agent ability, 2) communication range, 3) communication topology, 4) communication bandwidth, 5) collective size, 6) collective reconfigurability, and 7) collective composition [29]. This classification has provided a structured framework that guides the research and development within the MAS community, facilitating a more systematic exploration of agent-based systems and their capabilities.

Moving into the 2000s, Ota introduced a further categorization of tasks within MARSs, distinguishing between point-reaching and region-sweeping tasks, and between tasks that are performed once versus those that are repetitive [87]. This categorization addresses the

operational diversity of robotic tasks, highlighting the adaptability and application-specific configurations required in modern robotic systems.

The core methodologies in MAS research have traditionally been grounded in control theory, optimization, and game theory [85, 109, 91]. Specifically for MARS, control theory and optimization have served as the backbone for solving a wide range of robotic tasks, offering solutions that enhance precision, efficiency, and reliability in complex agent interactions.

In recent years, the field has experienced a resurgence of interest with the integration of deep reinforcement learning, bringing a fresh perspective to traditional multi-agent challenges. Huh’s recent survey on the use of reinforcement learning in multi-agent systems underscores this shift, highlighting the potential for these methods to revolutionize how agents learn and interact in shared environments [44].

Despite the well-established challenges of planning and control in MASs, the current literature has yet to fully address the emerging issues pertinent to the robotics community in the face of rapid technological evolution. Automated systems such as autonomous vehicles, unmanned aerial vehicles, and warehouse robots are becoming more ingrained in our daily lives, pushing the boundaries of current MAS applications and demanding novel solutions to new problems.

As these systems increasingly coexist with humans and potentially replace human labor in routine, labor-intensive tasks, engineering these systems for effective human cooperation becomes paramount. Particularly, exploring MASs within the framework of heterogeneous and homogeneous autonomy in transportation is essential, addressing significant challenges that arise from interactions among humans and robots as well as between robots themselves. This exploration is crucial for ensuring the safe, efficient, and ethical integration of autonomous systems into society and represents the main theme of this dissertation.

1.5 Summary of Contributions

The primary contributions of this thesis address contemporary and emergent planning and control problems in sequentially chained, partially meshed, and centrally connected MASs.

Attenuation of Error Propagation for Sequentially Chained MASs with Only Local Observation (Chapter 2)

- We have developed a novel ACC controller, which is a modification of Kerner’s model [55] and is compatible with a wide range of ACC systems.
- We demonstrate the *local* string stability of the Attenuative Kerner’s Model through frequency-domain analysis.
- We benchmark our controller against both a human driver and an OEM ACC controller, employing both simulation and field tests for comprehensive comparison.

Attenuation of Error Propagation for Sequentially Chained MASs with Additional Non-Local Observation (Chapter 3)

- We present a novel, efficient tri-layer MPC design for highway car-following based on a quadratic programming formulation.
- We introduce a practical prediction method utilizing an ETA estimator.
- We offer a comprehensive numerical study demonstrating the method's effectiveness and robustness against prediction errors.

Decentralized Conflict Resolution for Partially Meshed MASs (Chapter 4)

- We propose a decentralized conflict resolution method based on a consensus control strategy.
- We develop and release an open-source dataset that captures rich dynamics of human driving behaviors in understructured road environments.
- We validate our method numerically through simulations and using the collected dataset.

Scalable Network-Agnostic Scheduling for Centrally Connected MASs (Chapter 5)

- We propose a novel mixed-integer scheduling algorithm that works with arbitrary road network topologies.
- We introduce a reservation-based heuristic that enables the mixed-integer scheduling algorithm to scale to larger problem sizes.
- We evaluate the performance and efficiency of our method through a diverse set of numerical simulations.

Additionally, this dissertation also presents two additional contributions in the areas of 1) design of hardware and experiments for MAS method validation and 2) integration of deep learning and optimization for efficient and stable control.

Integration of Learning and Optimization for Planning and Control (Chapter 6)

- We present a novel controller design, i.e., Memory-Augmented MPC, in its standard form and two modified forms.
- We prove stability of Memory-Augmented MPC without imposing any condition on the approximation errors of the neural network.

- We demonstrate amortized computational efficiency of Memory-Augmented MPC via four numerical experiments.

Actuation and Estimation Approaches in MAS Experiments (Chapter 7)

- We present the modeling, hardware design, and experimental setup aspects of indoor robotic research, focusing on unicycle robots.
- We discuss the modeling and hardware aspects of autonomous driving, supplemented by a detailed case study on the experimental aspects of autonomous driving research.
- We describe three alternative approaches for setting up an estimation system dedicated to indoor robot localization.
- We provide detailed documentation on how to collect, annotate, and process aerial drone videos, including practical tips, an algorithmic roadmap, and descriptions of software tools.

1.6 Organization of the Dissertation

The remainder of this dissertation is organized as follows. Chapter 2 introduces the problem of string stability in sequentially chained MASs and proposes a solution known as the attenuative Kerner’s model for highway adaptive cruise control with only local observations. Chapter 3 continues the discussion on string stability in transportation and presents a hierarchical MPC controller that utilizes both local and non-local observations for optimal adaptive cruise control. Chapter 4 addresses conflict resolution in locally meshed MASs and proposes a set of consensus algorithms for collision avoidance within understructured road networks. Chapter 5 considers the problem of optimal scheduling in centrally connected MASs and introduces a scalable mixed-integer planning algorithm designed for large-scale, network-agnostic multi-agent scheduling. Chapter 7 discusses the hardware aspects and experimental design of MAS research, detailing two case studies as experimental works. Chapter 6 explores how black-box neural networks can enhance the amortized efficiency of planning and control methods, ensuring safety is not compromised. Finally, the dissertation concludes with final remarks and a discussion of potential future work in Chapter 8.

Chapter 2

Sequentially Chained Multi-Agent Systems with Local Observation

Sequentially chained MASs are arguably some of the simplest yet most prevalent MAS configurations in robotic applications, especially in ground transportation. A sequentially chained MAS typically emerges from the formation of a queue, such as a platoon of vehicles cruising on highways. In such scenarios, each agent interacts directly only with its immediate predecessor in a one-dimensional arrangement. For example, in a highway platoon, a following vehicle must maintain a safe distance from the vehicle ahead while attempting to drive as close to its preferred speed as possible. A common problem in such sequentially chained MASs is stop-and-go waves, which often arise from the so-called string instability. This chapter investigates methods to mitigate stop-and-go waves for highway platooning using controllers that rely exclusively on local observations.

2.1 Introduction

Recent traffic monitoring efforts have revealed the prevalence of stop-and-go waves on freeways. Initial investigations examined these waves [70], albeit with limitations due to the dataset’s length and duration, which constrained understanding of their prevalence. Wu [134] demonstrated the propagation and dissipation of stop-and-go waves over extended periods using aerial drone videos in afternoon rush hours on commute highways. Gloudemans [36] showed that during congestion periods, stop-and-go waves persist, moving at regular speeds and propagating through the traffic in a direction opposite to the flow, consistently observed throughout the congestion periods on all monitored days.

This phenomenon can be attributed to human driving dynamics, even in the absence of bottlenecks [114], or it can be induced by the unstable behavior of commercial Adaptive Cruise Control (ACC) systems [38]. When equipped with properly designed ACC systems, particularly those that are string stable [119], one can attenuate stop-and-go waves, leading to a reduction in fuel consumption and emissions. For example, previous studies have shown

a reduction in fuel consumption of nearly 40% as reported in [113], and a reduction in emissions ranging between 15% and 73% as quantified in [112].

Such results underscore the importance of considering the role that Adaptive Cruise Control (ACC) might play in promoting sustainable highway transportation practices. Mercedes was the first automaker to introduce ACC in the U.S. market in 1999, yet until recently, string stability was not a design criterion considered by major automobile manufacturers. Given that a significant portion of ACC systems on the road are potentially string instable [38], there is merit in retrofitting these legacy ACC systems on the roads.

However, existing literature has not given enough consideration in this aspect. Common ACC methods involve the manipulation of acceleration commands [66, 79, 9, 55]. However, by design, ACC systems are intended to accept velocity commands. In addition, optimization-based and flow-based methods frequently require wireless communications and costly onboard computing [99, 128, 132, 37, 82]. Such hardware requirements are often unattainable for earlier automobiles. Furthermore, learning-based methods inherently function as black boxes [94, 130, 51]. This characteristic complicates the verification of string stability, a crucial factor given the safety-critical nature of driving.

Therefore, developing a method that can provably stabilize stop-and-go waves and is widely compatible with a broad range of commercial vehicles equipped with ACC is crucial. To address these challenges, we introduce a string stable ACC method that overcomes issues of backward compatibility. Building upon the acceleration-based Kerner’s model [55], we propose a novel velocity-based controller, named the *Attenuative Kerner’s Model* (AKM).

We summarize the main contributions of our work below.

- We have developed a novel ACC controller, which is a modification of Kerner’s model [55] and is compatible with a wide range of ACC systems.
- We demonstrate the *local* string stability of the Attenuative Kerner’s Model through frequency-domain analysis.
- We benchmark our controller against both a human driver and an OEM ACC controller, employing both simulation and field tests for comprehensive comparison.

2.2 Related Works

Stop-and-go waves are a traffic phenomenon commonly emerged on congested highways. They can be triggered by the presence of a road bottleneck, e.g., a drop in speed limit or a reduction in lane capacity, or arise naturally due to the inherent instability in human car-following behavior. String stability has been introduced and is utilized to qualitatively assess the presence of instability in car-following dynamics [119]. The oscillatory dynamics observed in this experiment were later modeled as analogs to gas-dynamical detonation waves [33], enabling researchers to explore how these waves could be dampened. Sugiyama [114] provided the first closed-course experimental validation, demonstrating the existence of stop-and-go

waves resulting from unstable human car-following behavior. Gunter [38] demonstrated through field experiments that such unstable car-following dynamics could also be present in commercial ACC systems.

Traditionally, control theory researchers have been exploiting the robotic nature of cars as both sensors and actuators to stabilize car-following. On a microscopic level, a classical approach to stabilizing car-following involves optimizing parameters of car-following models as demonstrated in [66, 116]. Another strategy involves construction of explicit feedback control mechanisms focusing on gap regulation and speed tracking [79, 9, 55]. Furthermore, there are studies that apply model predictive control to design implicit feedback controllers by leveraging additional non-local information [99, 132, 37]. On a macroscopic level, alternative methods to mitigate traffic congestion include the adaptation of infrastructure compliance, such as dynamically setting speed limits based on downstream traffic conditions [128, 82].

Meanwhile, machine learning, particularly reinforcement learning, has emerged as an alternative approach to wave smoothing, with many instances showing promising empirical performance. The early adoption of reinforcement learning for adaptive cruise control is documented in [81]. More recent explorations in this area are detailed in [94, 130, 51]. While these modern methods excel at utilizing both synthetic and real-world data to enhance performance, their certification for string stability remains challenging due to the opaque nature of black-box neural networks.

Parallel to controller design, experimental work is also essential, particularly in assessing the robustness of proposed solutions. The challenge of implementing precise acceleration-based feedback control arises from the often noisy nature of odometry or IMU measurements [111]. Additionally, Rajamani [95] demonstrated that the integration of various time delays from each sensor could lead to instabilities if the controller is not designed to accommodate them. Therefore, conducting experimental evaluations is vital to confirm the practicality of a proposed ACC method.

On the side of field experimentation, an early experiment by Stern [113] aimed to mitigate stop-and-go waves through the control of a single vehicle on a circular, single-lane road. More recently, the significant decrease in equipment costs has enabled coordinated experiments on multi-lane highways, such as those described in [77, 67].

2.3 Attenuative Kerner’s Model

Consider the two-vehicle car-following setup, as depicted in Figure 2.1. At any given moment t , the lead vehicle’s position, velocity, and acceleration are denoted by $s_0(t)$, $v_0(t)$, and $a_0(t)$, respectively, while those for the ego vehicle are similarly represented by $s_1(t)$, $v_1(t)$, and $a_1(t)$. The relative distance and velocity between the vehicles are defined as $\Delta s(t) := s_0(t) - s_1(t)$ and $\Delta v(t) := v_0(t) - v_1(t)$. The goal is to control the ego vehicle’s behavior to ensure it follows the lead vehicle safely and smoothly in stop-and-go traffic conditions.

Given that all ACC systems on the market accept a set speed as the control input, designing a controller that outputs a velocity command ensures the highest compatibility.

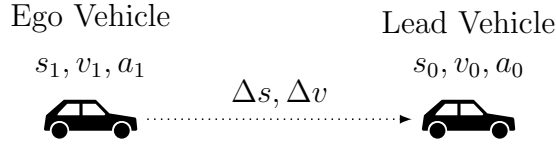


Figure 2.1: Illustration of the two-vehicle car-following problem. The ego vehicle needs to follow the lead vehicle as smoothly as safety permits.

Therefore, building upon Kerner's model [55], we introduce the AKM as follows:

$$u_{\text{AKM}}(\Delta s, v_0, v_1) = \begin{cases} v_0 + \delta_-(\Delta s, v_1), & \Delta s \in S_-(v_1), \\ v_0, & \Delta s \in S_{\text{eq}}(v_1), \\ v_0 + \delta_+(\Delta s, v_1), & \Delta s \in S_+(v_1), \end{cases} \quad (2.1)$$

where

$$\begin{aligned} \delta_-(\Delta s, v_1) &= \max(a_1 \Delta s / q(v_1) + b_1, d_1), \\ \delta_+(\Delta s, v_1) &= \min(a_2 \Delta s / q(v_1) + b_2, d_2), \\ S_-(v_1) &= \{\Delta s \in \mathbb{R}_{\geq 0} : \Delta s / q(v_1) < h_-\}, \\ S_+(v_1) &= \{\Delta s \in \mathbb{R}_{\geq 0} : \Delta s / q(v_1) > h_+\}, \\ S_{\text{eq}}(v_1) &= \mathbb{R}_{\geq 0} \setminus (S_-(v_1) \cup S_+(v_1)), \\ q(v_1) &:= \max(v_1, v_{\min}), \end{aligned} \quad (2.2)$$

for some parameters $a_1, a_2, b_1, b_2, d_1, d_2, h_-, h_+$, and $v_{\min}$. At start-up, we set $u_{\text{AKM}}(t_0) = v_0(t_0)$.

The parameters a_1, b_1 , and d_1 in $\delta_-(\cdot)$ govern the negative velocity compensation, prompting the ego vehicle to decelerate and increase the space headway. Conversely, the parameters a_2, b_2 , and d_2 in $\delta_+(\cdot)$ govern the positive velocity compensation, encouraging the ego vehicle to accelerate and decrease the space headway. A graphical representation of the velocity compensation terms is depicted in Figure 2.2a. The parameters h_- and h_+ represent the minimum and maximum space headways, respectively, where $h_- < h_+$. These parameters are used to track the lead vehicle's speed. Finally, the parameter $v_{\min}$ defines the minimum velocity threshold critical for a meaningful time headway.

The three switch conditions of the AKM aim to smooth oscillations in the lead vehicle's speed while maintaining a reasonable space headway between the two vehicles. When $\Delta s \in S_-$, indicating that the space headway is too small, the controller commands a speed lower than that of the lead vehicle to increase the gap. When $\Delta s \in S_+$, indicating that the space headway is too large, the controller commands a speed higher than the speed of the lead vehicle to reduce the gap. When $\Delta s \in S_{\text{eq}}$, indicating that the space headway is optimal—neither too small nor too large—the ego vehicle adopts the lead vehicle's speed, attenuating oscillations in the lead vehicle's speed through natural attenuation property of cruise control speed tracking.

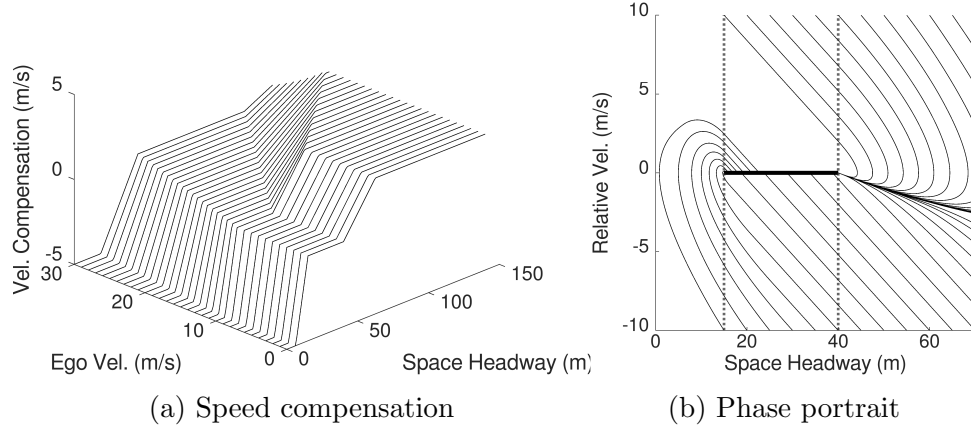


Figure 2.2: Speed compensation and phase portrait of the AKM with parameters in Table 2.1.

Note that the AKM model (2.1) is formulated in continuous-time. For implementation on a physical device, the following discrete-time adaptation is proposed:

$$u_{\text{AKM}}[k+1] = \begin{cases} v_0[k] + \delta_-(\Delta s[k], v_1[k]), & \text{if } \Delta s[k] \in S_-(v_1[k]), \\ \alpha \cdot v_0[k] + (1 - \alpha) \cdot u_{\text{AKM}}[k], & \text{if } \Delta s[k] \in S_{\text{eq}}(v_1[k]), \\ v_0[k] + \delta_+(\Delta s[k], v_1[k]), & \text{if } \Delta s[k] \in S_+(v_1[k]), \end{cases} \quad (2.3)$$

where $\alpha \in (0, 1]$ is a tuning parameter that adjusts the responsiveness of the cruise control's speed tracking. Should the onboard cruise control's speed tracking prove overly responsive, leading to minimal attenuation effects, it is recommended to decrease α towards zero. If this issue does not arise, α should remain at 1.

To demonstrate the integration of the AKM model within standard vehicle control systems, a typical system block diagram is depicted in Figure 2.3.

2.4 String Stability Analysis

In this section, we establish the local string stability of the proposed controller, illustrating the use of the vehicle's inherent speed tracking to dampen speed oscillations.

Assuming the ego vehicle's cruise control system functions as a proportional controller, the dynamics of the two-vehicle car-following system are described as follows:

$$\dot{v}_1 = k_p(u_{\text{AKM}}(\Delta s, v_0, v_1) - v_1), \quad (2.4)$$

where $k_p > 0$ represents the proportional gain for velocity tracking. Let $v^* > 0$ represent the desired cruising speed targeted by all vehicles. We define the velocity errors as $e_0 := v_0 - v^*$ and $e_1 := v_1 - v^*$.

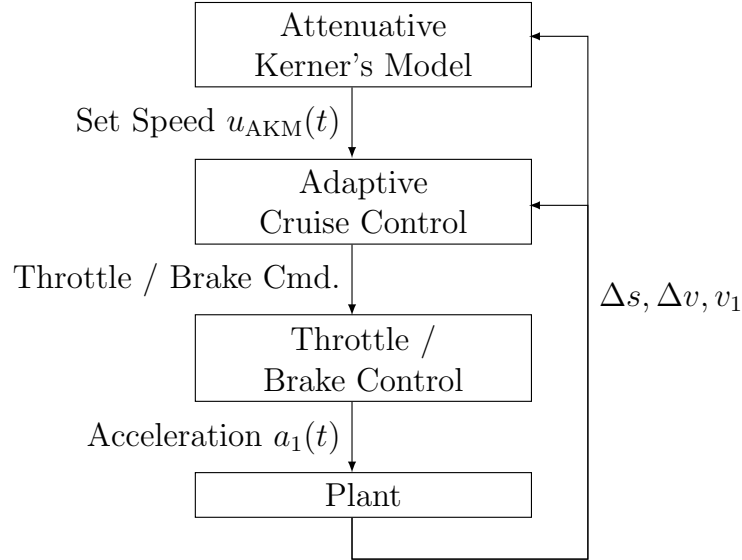


Figure 2.3: Block diagram of a typical vehicle system integrated with the AKM controller.

A phase portrait of the system (2.4), with parameters chosen from Table 2.1 and $v_0 \equiv 10$ m/s, is depicted in Figure 2.2b. This phase portrait considers only trajectories originating from feasible initial conditions. The vertical dashed lines divide the space into three segments: from left to right, S_- , S_{eq} , and S_+ . Qualitatively, the system exhibits asymptotic convergence to a one-dimensional equilibrium set, indicated by the bold black horizontal line segment $\{(\Delta s, \Delta v) : 15 \leq \Delta s \leq 40, \Delta v = 0\}$. Additionally, the system converges to the equilibrium set via a sliding surface when within S_+ , i.e., when the space headway exceeds 40 m.

Translating the above observation into velocity error dynamics, we find that equilibrium in the coupled velocity error dynamical system occurs when $v_0 = v_1 = v^*$ and $\Delta s \in S_{eq}(v^*)$, a compact set with a nonempty interior. The dynamics of the coupled error within this equilibrium set are described by:

$$\dot{e}_1 = k_p(e_0 - e_1), \quad (2.5)$$

forming a linear time-invariant system. Applying the Laplace transform to both sides yields the transfer function:

$$T(s) := \frac{E_1(s)}{E_0(s)} = \frac{k_p}{s + k_p}.$$

As introduced in [48, 66, 80], string stability is a property that describes that a harmonic disturbance experienced by a vehicle driving in equilibrium in a platoon will be attenuated by the subsequent vehicle over time. Following the convention in [129], we formalize the notion of locally string stability below.

Definition 1 (Local String Stability). The system (2.4) is locally string stable around an

equilibrium if $T(s)$ exists and

$$\max_{\forall \omega \in \mathbb{R}} \|T(j\omega)\| \leq 1.$$

Next, we show that the controller is locally string stable within the equilibrium set.

Theorem 1. The system (2.4) is locally string stable for all $k_p > 0$.

Proof. Locally within the equilibrium set of (2.4), the error dynamical system takes the form of (2.5). Since $k_p > 0$, the local homogeneous response of (2.4) is stable, which implies existence of steady states. Looking at frequency response of the steady state for ω , we have

$$\|T(j\omega)\| = \left\| \frac{k_p}{j\omega + k_p} \right\| = \frac{k_p}{\sqrt{\omega^2 + k_p^2}} \leq 1,$$

for all $\omega \in \mathbb{R}$. By definition, system (2.4) is locally string stable. $\square$

In fact, for $\omega \neq 0$, we have $\|T(j\omega)\| < 1$, indicating that the controller invariably attenuates nonzero harmonic disturbances. The proof demonstrates that the AKM's second mode consistently acts as an attenuator, attributed to the intrinsic damping effect of cruise control speed tracking. To avert potential collisions or lagging caused by excessive dampening in the second mode, the first and third modes function as headway regulators.

2.5 Numerical Simulation

To assess the performance of the AKM, we conduct numerical simulations of the model and contrast its outcomes with those from two benchmark approaches: a model of human driver and a model of a commercial ACC.

2.5.1 Simulation Setup

Imitating real-world stop-and-go traffic, we consider the lead vehicle's speed to follow an oscillatory profile, defined as

$$v_0^{\text{ref}}(t) = A \sin(\omega t) + B, \quad (2.6)$$

where $A = 3.35$ m/s, $\omega = \frac{2\pi}{20}$ s⁻¹, and $B = 5.59$ m/s. This profile causes the speed to oscillate between 2.24 m/s (5 mph or 8.05 km/h) and 8.94 m/s (20 mph or 32.19 km/h), achieving a periodicity of 20 seconds.

We selected this harmonic oscillation profile to closely mimic the low-speed stop-and-go waves observed on US highways, while simplifying it to facilitate the comparison of wave attenuation effects. Specifically, we referenced the stop-and-go wave patterns that occur daily during afternoon rush hours on CA State Route 242 near Concord, CA.

We model the vehicle dynamics with a first-order, time-invariant system. This system takes a set speed, u , as input and produces an actuated speed, v , as its output. The dynamics are represented by the equation

$$\dot{v}(t) = k_p(u(t) - v(t)), \quad (2.7)$$

where, by system identification, k_p is set to 0.32.

Furthermore, for our numerical analysis and subsequent field experiments, we select a specific set of parameters, as detailed in Table 2.1. With the speed profile of the lead

Table 2.1: Parameters of the Attenuative Kerner's Model

a_1 (m/s ²)	a_2 (m/s ²)	b_1 (m/s)	b_2 (m/s)	d_1 (m/s)	d_2 (m/s)	h_- (s)	h_+ (s)	$v_{\min}$ (m/s)	α —
5.71	1.33	-8.57	-5.33	-5.0	3.0	1.5	4.0	10	0.2

vehicle, the dynamics of the ego vehicle, and the controller parameters described above, we can simulate the proposed AKM, enabling it to follow the lead vehicle at the specified speed.

To evaluate the effectiveness of the AKM, we simulate a human driver using the Intelligent Driver Model (IDM) [123], a widely utilized model for representing human driving behavior. Likewise, the Adaptive Cruise Control (ACC) is represented through a modified Optimal Velocity Model (OVM) [66, 79], which is frequently selected for simulating cruise control systems.

To ensure the accuracy of the model, we collect driving data recorded by a human driver and a commercial ACC. We calibrate the human and ACC models by minimizing the discrepancy between the models and the data collected from real-world human and ACC driving, respectively. The calibration procedure, adapted from [38], for a $\mathbf{k}$ -parameterized model $f(\mathbf{x} \mid \mathbf{k})$, is presented below:

$$\mathbf{k}^* = \min_{\mathbf{k}, X} w_1 \ell_1(X, \Delta \hat{S}) + w_2 \ell_2(X, \hat{V}) + w_3 \ell_3(X) \quad (2.8a)$$

$$\text{s.t. } \mathbf{x}(0) = \hat{\mathbf{x}}_0 \quad (2.8b)$$

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ \dot{v}_0(t) \\ 0 \\ f(\mathbf{x} \mid \mathbf{k}) \end{bmatrix}, \quad (2.8c)$$

where $\mathbf{x} := [s_0 \ v_0 \ s_1 \ v_1]^\top$, $X := \{\mathbf{x}(t) : \forall t \in T\}$, $\Delta \hat{S} := \{\Delta \hat{s}(t) : \forall t \in T\}$, and

$\hat{V} := \{\hat{v}_1(t) : \forall t \in T\}$ for a specified duration T and

$$\begin{aligned}\ell_1(X, \Delta \hat{S}) &= \int (\mathbf{x}_1(t) - \mathbf{x}_3(t) - \Delta \hat{s}(t))^2 dt, \\ \ell_2(X, \hat{V}) &= \int (\mathbf{x}_4(t) - \hat{v}_1(t))^2 dt, \\ \ell_3(X) &= \int (\dot{\mathbf{x}}_4(t))^2 dt.\end{aligned}$$

Note that we use $\hat{\cdot}$ to distinguish data collected from real-world experiments from data computed from simulations.

The three penalty terms are used for minimizing model-data discrepancy and for model regulation. With $\ell_1(\cdot)$, we penalize the difference between simulated and actual space headway. With $\ell_2(\cdot)$, we penalize the difference between simulated and actual velocity. With $\ell_3(\cdot)$, we penalize large accelerations from the calibrated model to avoid overfitting.

The calibrated IDM for the human driver is

$$a_{\text{HUM}}(\Delta s, v_1, \Delta v) = a \cdot \left(1 - \left(\frac{v_1}{v^*} \right)^\delta - \left(\frac{s^*(v_1, \Delta v)}{\Delta s} \right)^2 \right), \quad (2.9)$$

where

$$s^*(v_1, \Delta v) = s_{\min} + v_1 T + \frac{v_1 \Delta v}{2\sqrt{ab}}, \quad (2.10)$$

with $a = 2.0 \text{ m/s}^2$, $b = 2.0681 \text{ m/s}^2$, $\delta = 4$, $T = 0.7254 \text{ s}$, $s_{\min} = 6.5489 \text{ m}$, and $v^* = 11.08 \text{ m/s}$.

The calibrated OVM for commercial ACC is

$$a_{\text{ACC}}(\Delta s, v_1, \Delta v) = k_\alpha(\Delta s + \gamma_1 v_1 + \gamma_0) + k_\beta \Delta v, \quad (2.11)$$

with $k_\alpha = 0.1222 \text{ s}^{-2}$, $k_\beta = 2.5094 \text{ s}^{-1}$, $\gamma_0 = -1.6423 \text{ m}$, and $\gamma_1 = -0.7925 \text{ s}$.

2.5.2 Numerical Results

We simulate the AKM, the human driving model, and the ACC model in low-speed stop-and-go traffic, as characterized by Equation 2.6. The results are presented as time series in Figure 2.4 and as phase portraits in Figure 2.5. From Figures 2.4 and 2.5, we observe that

- The human driving model, as expected, does not reduce the oscillations present in the lead vehicle's trajectory;
- The ACC model moderately dampens the oscillations in the lead vehicle's trajectory; and
- The AKM attenuates the oscillations more effectively than the ACC model.

The above observation is particularly evident in Figure 2.5b, where an oscillatory trajectory manifests as a periodic orbit. The more oscillatory the trajectory, the larger the radius of the periodic orbit appears in the phase portrait. As depicted in the plot:

- The periodic orbit of the human driving model almost coincides with that of the lead vehicle;
- The periodic orbit of the ACC model is encompassed within that of the lead vehicle; and
- The periodic orbit of the AKM is situated within that of the ACC model.

Examining the headway plots in Figures 2.4a and 2.5a, we observe that the AKM achieves smoother car-following behavior at the expense of a larger distance between the ego vehicle and the lead vehicle. The human driving model's oscillation occurs within a 2 to 3 s time headway interval; the ACC model oscillates within a 2.5 to 4 s interval; and the AKM model oscillates within a 4 to 5 s interval.

We note that the trade-off between maximum headway and driving smoothness represents the most critical design consideration in developing a wave-dampening controller. The closer a controller is designed to follow the lead vehicle, the smaller the buffer space becomes for mitigating the impacts of stop-and-go behavior exerted by the lead vehicle.

2.5.3 Energy Efficiency

Beyond qualitative analysis of the trajectories, it is beneficial to quantitatively compare the three models in terms of their energy consumption efficiency. To this end, we employed an energy consumption model suitable for a typical midsize SUV. This energy model is one of six energy models [58] derived and validated through Autonomie [39], a simulation tool developed by Argonne National Laboratory that includes libraries of different vehicle types.

The model is represented by a function $e(v, a, \theta)$ that maps speed v , acceleration a , and road grade θ to fuel consumption rate e . This empirical model leverages the concept that the instantaneous engine speed N and engine torque T can be directly related to fuel rate. Furthermore, transmission output speed N_{output} and wheel force F_{wheel} are mapped to engine speed and torque. Consequently, both N_{output} and F_{wheel} can be expressed as functions of speed, acceleration, and road grade.

The energy consumption model $e(v, a, \theta)$ is given by

$$e(v, a, \theta) = \max \{ \ell(v, a, \theta), f_p(v, a, \theta) \}, \quad (2.12)$$

where

$$\begin{aligned} \ell(v, a, \theta) &= \begin{cases} \beta, & \text{if } v \leq v_c, \\ 0, & \text{if } v > v_c \text{ and } a < a_c(v, \theta), \end{cases} \\ f_p(v, a, \theta) &= C(v) + P(v)a + Q(v)(a_+)^2 + Z(v)\theta. \end{aligned}$$

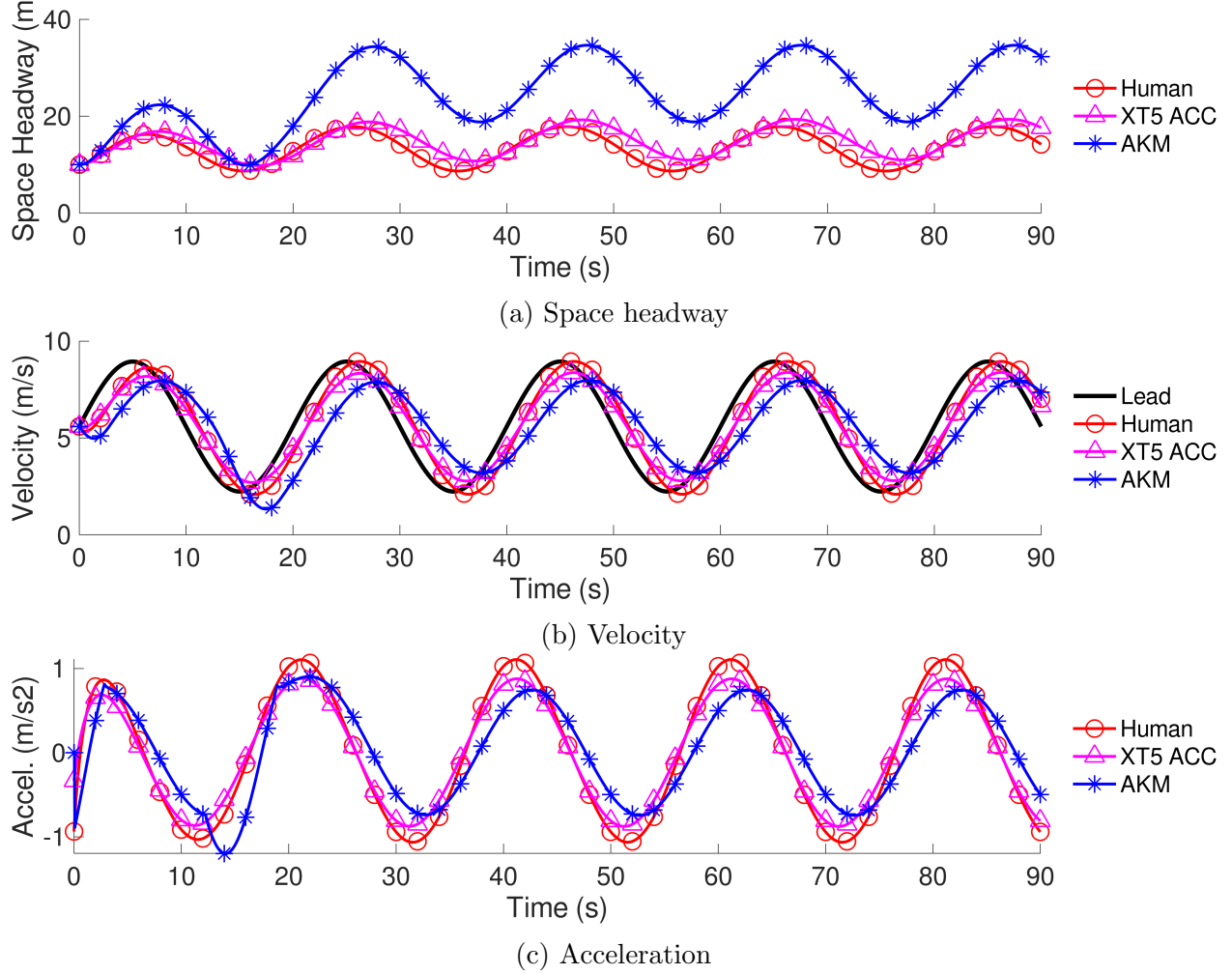


Figure 2.4: Simulated trajectories of the AKM, human driver, and ACC.

The function $f_p(v, a, \theta)$ denotes the fuel rate and is a polynomial function of speed v , acceleration a , and road grade θ . The term $\ell(v, a, \theta)$ serves as a lower bound for the fuel rate consumed by the vehicle. The critical speed for fuel-cut activation is denoted by v_c , while the acceleration threshold for fuel-cut, dependent on speed and road grade, is defined by the polynomial $a_c(v, \theta)$. The coefficients $C(v)$, $P(v)$, and $Q(v)$ are polynomial functions of speed designed to ensure that $f_p(v, a, 0)$ is monotonically increasing with respect to a , by setting a constraint that the minimum $f_p(v, a_{\min}(v), 0)$ remains positive.

Despite its simplicity, this energy consumption model facilitates the convenient calculation of the vehicle's fuel consumption rate in grams-per-second of *gasoline*. In this section and the subsequent ones, we assume the road grade to be zero, as this represents the most common scenario in the real world. Utilizing this energy consumption model, we compute the energy efficiency of the three car-following models. Sorting the efficiency of the three

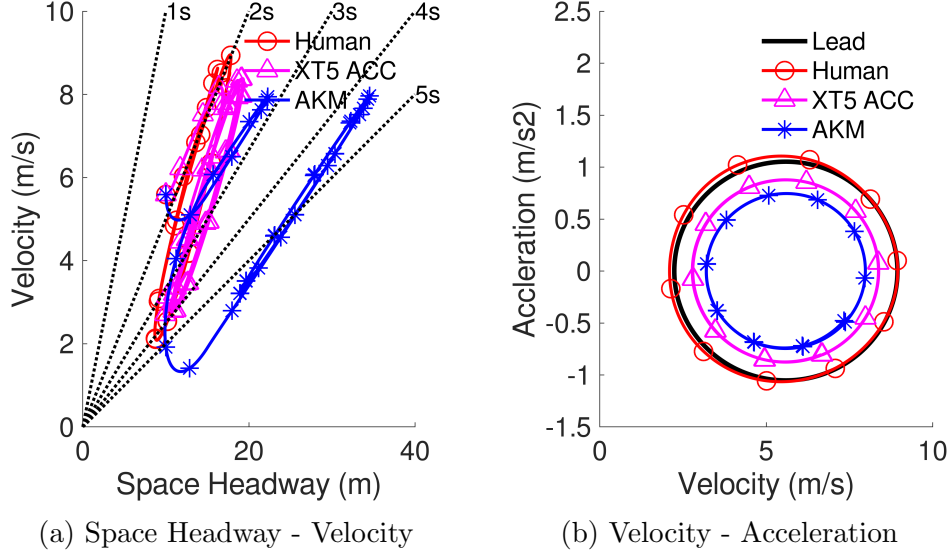


Figure 2.5: Simulated phase portraits of the AKM, human driver, and ACC. The dashed lines in (a) indicate phase motion of different constant time headways.

models in ascending order yields the fuel rate of the human driver 0.5697 g/s, the ACC 0.5090 g/s, and the most efficient AKM 0.4900 g/s. This quantitative result aligns with the qualitative analysis presented earlier.

2.5.4 Robustness to Disturbances

Because human drivers never adheres to the idealistic sinusoidal velocity in (2.6), we further examine the performance of all models under a more realistic velocity profile with disturbances. To emulate real-world disturbances, we have augmented (2.6) with two additional harmonic waves, as shown below:

$$v_0^{\text{ref}}(t) = A_1 \sin(\omega_1 t) + A_2 \sin(\omega_2 t) + A_3 \sin(\omega_3 t) + B, \quad (2.13)$$

where $A_1 = 3.35$ m/s, $A_2 = 0.509$ m/s, $A_3 = 0.0159$ m/s, $\omega_1 = \frac{2\pi}{20}$ s⁻¹, $\omega_2 = \frac{2\pi}{8}$ s⁻¹, $\omega_3 = 2\pi$ s⁻¹, and $B = 5.59$ m / s.

Simulations with this disturbed velocity profile have yielded largely consistent results. The acceleration-velocity phase portrait, illustrating the robustness of our method, is displayed in Figure 2.6. Again sorting the efficiency of the three models, the fuel rates are as follows: the human driver at 0.5695 g/s, the ACC at 0.5095 g/s, and the most efficient AKM at 0.4881 g/s. As demonstrated, the performance of the AKM does not deteriorate under disturbances, suggesting its robustness to such conditions.

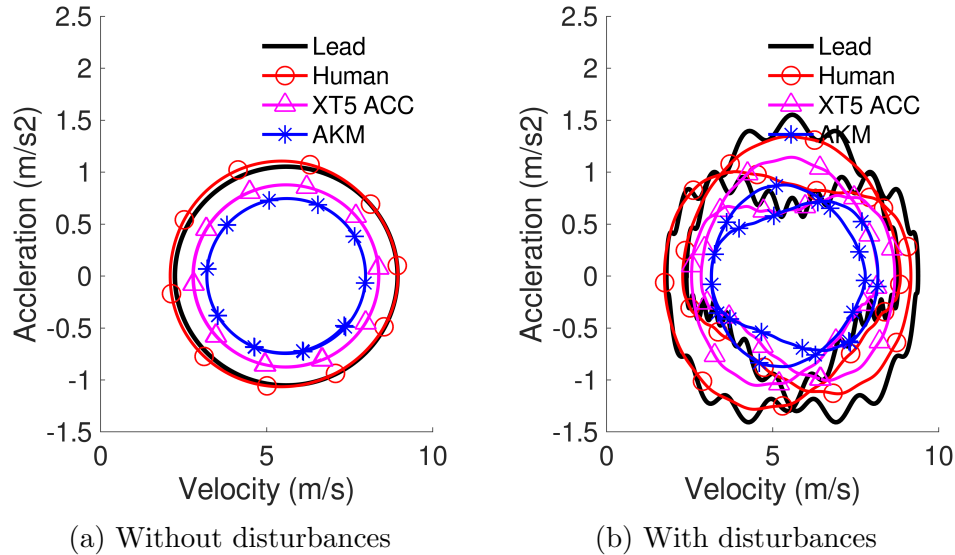


Figure 2.6: Evaluation of the impacts of disturbances on AKM. The qualitative efficiency ranking of these models remain unchanged.

2.6 Physical Experiments

Given that sensing and actuation in actual vehicles are typically far from perfect [106, 13], verifying the performance of a controller in a real vehicle is crucial to attest its robustness. Unlike most prior studies, we conducted field experiments to validate the performance of our proposed AKM method. To achieve this, we customized a 2022 Cadillac XT5 and carried out the same set of experiments as described in the simulation on this custom SUV.

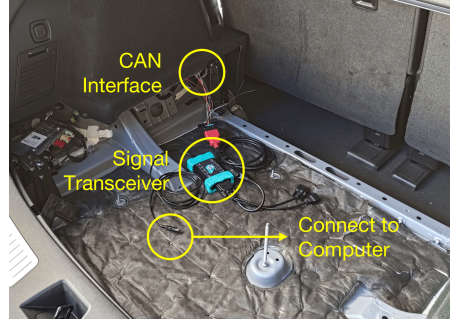
2.6.1 Experimental Setup

To achieve this goal, we deployed the AKM model on a 2022 Cadillac XT5, as illustrated in Figure 2.7a. To interface with the vehicle, we utilized a commercial CAN bus transceiver named neoVI FIRE2 serving as the interface device to query sensor estimates and issue actuation commands. A trunk view of the hardware integration is shown in Figure 2.7b. The XT5 is configured to accept velocity commands, in accordance with the block diagram presented in Figure 2.3. This customization allows the vehicle to track both constant and variable velocity inputs provided by the AKM.

The experiments were conducted at the Richmond Field Station in California, US, as illustrated by the satellite image in Figure 2.8. The test track starts at the exit of the left parking lot and concludes at the entrance of the right parking lot, spanning a total length of approximately 640 meters. In each test, a lead human driver was tasked with adhering as closely as possible to the velocity profile specified in Equation (2.6). Simultaneously, the ego vehicle, under the supervision of a safety driver, followed the lead vehicle employing



(a) Angled view



(b) Trunk view

Figure 2.7: An angled view and a trunk view of the automated Cadillac XT5.

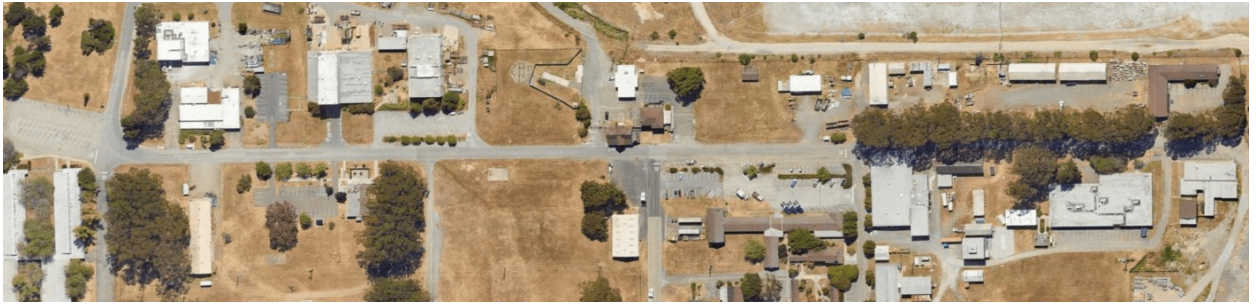


Figure 2.8: Satellite image of the experiment site in Richmond, CA, USA.

either manual driving, the onboard ACC, or the AKM. To account for variations in human execution of the lead reference trajectory, we conducted five repeated runs for each of the three car-following modes.

2.6.2 Experimental Results

Sample trajectories of human driving, the XT5 ACC, and the AKM are depicted in Figures 2.9 and 2.10. When comparing the experimental results to those from simulations, we observe that the high-level car-following characteristics of the three methods remain consistent: human driving exhibits the least effectiveness in wave dampening; the AKM performs the best; and the XT5 ACC ranks in the middle.

The primary distinction between numerical simulations and physical experiments lies in the inability of the human driving and ACC models to replicate short bursts of acceleration, as evidenced in Figure 2.9c. This suggests that, in practice, both actual human driving and the XT5 ACC adopt a more aggressive approach towards closing any significant gaps between the ego vehicle and the lead vehicle, compared to their simulated counterparts.

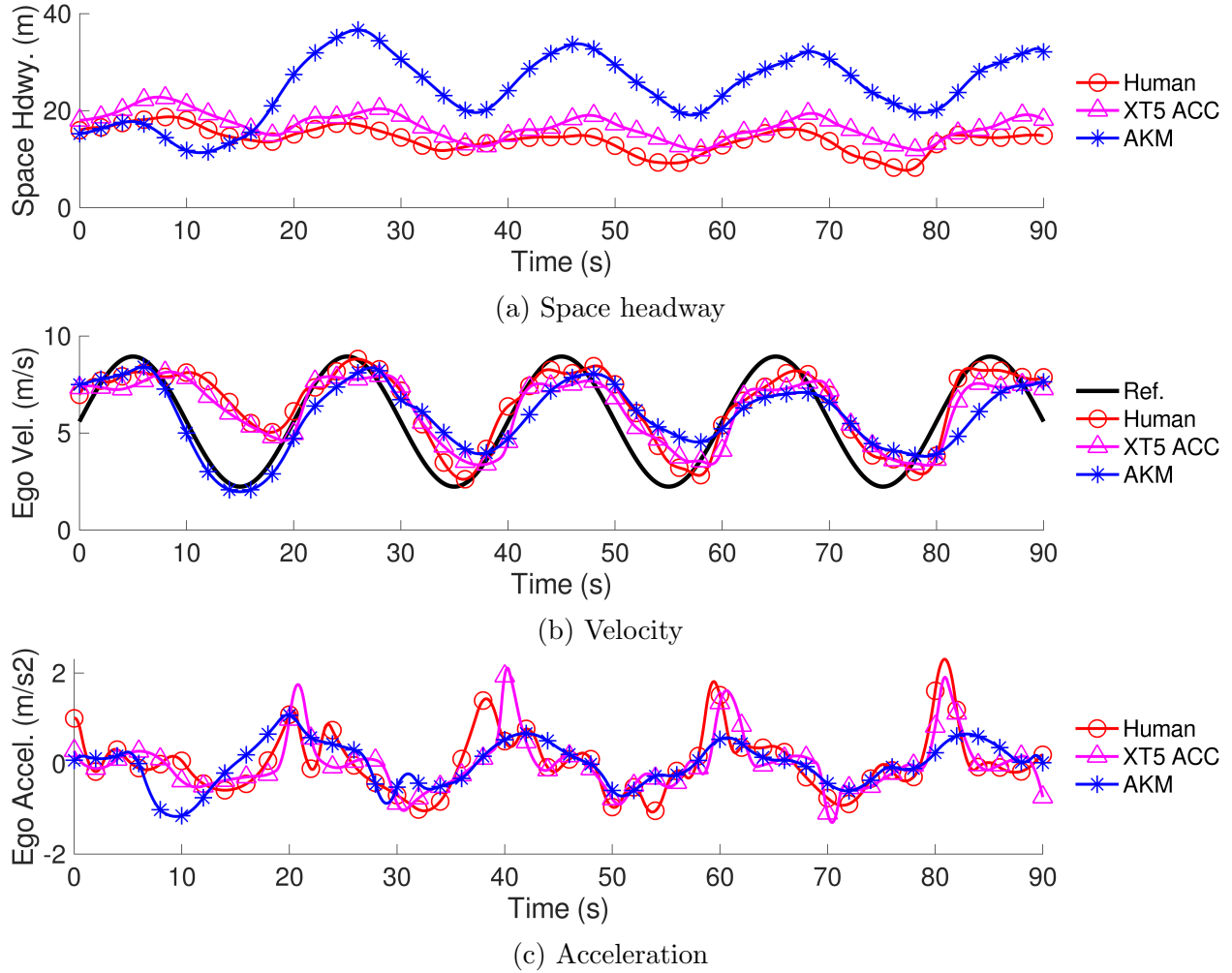


Figure 2.9: Measured trajectories of the AKM, human driver, and ACC.

2.6.3 Energy Efficiency

Utilizing the energy consumption model presented in Equation (2.12), we can further qualitatively evaluate the efficiency of the three car-following methods. The fuel consumption rates for each method across all test runs are succinctly summarized in Table 2.2. Analysis of the data reveals that although the numerical values differ from those obtained in simulations, the relative order of energy efficiency remains consistent: the AKM is the most efficient; human driving is the least efficient; and the XT5 ACC occupies the middle ground.

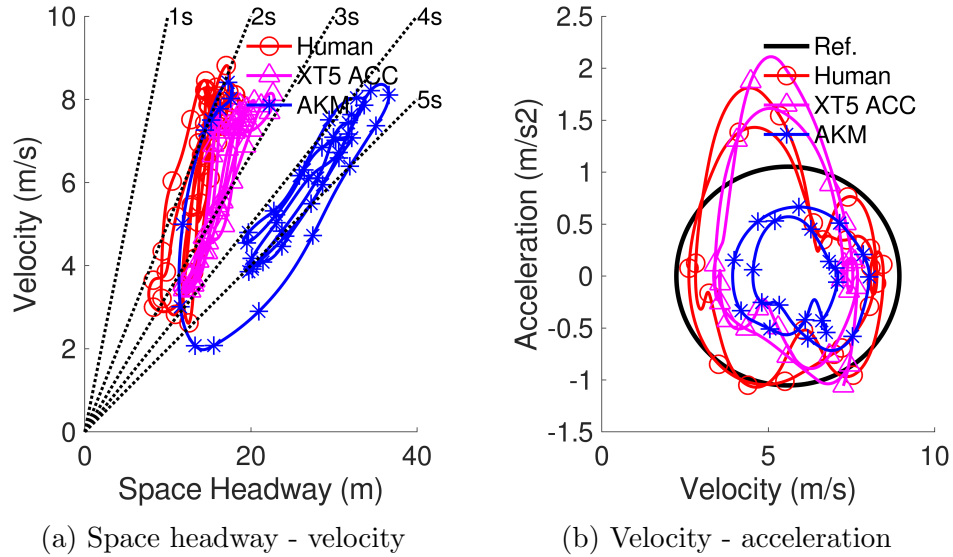


Figure 2.10: Measured phase portraits of the AKM, human driver, and ACC.

Table 2.2: Measured Fuel Rates (g/s Gasoline) in the Physical Experiments

	Run 1	Run 2	Run 3	Run 4	Run 5	Mean
Human	0.5182	0.5134	0.5136	0.5277	0.5277	0.5201
XT5 ACC	0.5043	0.4748	0.4874	0.4424	0.4472	0.4712
AKM	0.4246	0.4270	0.4450	0.4324	0.4167	0.4291

2.7 Discussion

Before concluding, we want to remark on several important theoretical and practical aspects of our works.

2.7.1 Theoretical Remarks

In Section 2.4, we presented a qualitative illustration of the global convergence property to an equilibrium set and the local string stability of the AKM within that equilibrium set. However, a proof of such convergence and an examination of global string stability remain pending. Given that the full system exhibits piecewise affine characteristics, rendering it nonlinear, traditional approaches for linear systems are inapplicable. This complexity makes the analysis challenging.

To demonstrate global convergence to the equilibrium set, we must carefully analyze the transitions among the three jump sets S_- , S_{eq} , and S_+ . Alternatively, we could attempt to

numerically solve for a Lyapunov function using the procedure described in [53].

To probe into the string stability of this nonlinear dynamic system, a potential avenue is to assess the bounded-input-bounded-output (BIBO) stability of the error dynamics. Beyond BIBO stability, it is imperative to also demonstrate that error propagation behaves as a contraction map. Pursuing these two propositions necessitates exploring outside the confines of linear systems theory.

2.7.2 Practical Considerations

First and foremost, we emphasize the necessity of precise calibration of AKM parameters, including a_1, a_2, h_-, h_+ , and α . In scenarios where the ACC exhibits sluggishness due to delays or quantization, adjustments to a_1, a_2, h_-, h_+ are critical to optimize response characteristics. Specifically, if the vehicle tends to drive too close to the lead vehicle, one should consider decreasing a_1 while increasing h_- . Conversely, if the vehicle consistently lags behind the lead vehicle, increasing a_2 while decreasing h_+ should be considered. In the second mode, where the ACC response to reference speed is overly rapid, reducing α will improve the dampening effect. Lastly, imposing additional constraints on u_{AKM} may be required to maintain speed commands within legal limits.

When operating on highways, it is essential to consider the potential side effects of increased space headway due to the AKM. A primary disadvantage of increased space headway is the resultant decrease in traffic flow density, particularly during rush hours. Consequently, this reduction may lead to decreased highway capacity, potentially causing greater congestion on local roads and highway onramps. Another shortcoming occurs when the AKM activates in traffic flows where the density is below the critical threshold, potentially reducing overall traffic throughput. However, this scenario is unlikely since stop-and-go waves typically occur above the critical density.

Furthermore, we acknowledge that our model does not account for two critical scenarios, which require human intervention. The first involves the lead vehicle initiating an emergency stop. Under such circumstances, we expect human drivers to intervene promptly by applying brakes to avert a collision. The second scenario occurs when the lead vehicle accelerates beyond the speed limit and exits the radar sensors' detection range. To prevent the following vehicle from engaging in unsafe pursuit, we recommend that a human driver take over or that supplementary measures be incorporated.

Lastly, we comment on the most salient practical challenges encountered during the project. Among these, customizing the Cadillac XT5 for automated driving presented significant difficulties. Part of the challenge stemmed from developing the driver to interface with the vehicle CAN buses. Additionally, mastering the temporal requirements of each CAN bus signal proved complex: certain signals must be transmitted at specific times or frequencies to be registered by the CAN buses. Another challenge involved designing experiments with the appropriate level of complexity to demonstrate the effectiveness of our method. Conducting full-scale field tests on highways introduced too many variables, complicating

the isolation of our controller's impacts. Instead, emulating repeatable highway scenarios in controlled environments struck an optimal balance between complexity and control.

Chapter 3

Sequentially Chained Multi-Agent Systems with Non-Local Observation

The reliability and ubiquity of wireless communication, including the Internet and local WLAN networks, enable modern robotic devices to access extensive data beyond local sensor feedback. In scenarios such as car-following on highways, the onboard ACC system not only monitors the speed and relative distance between the ego and lead vehicles but also gathers downstream traffic metrics—speed, density, throughput, and estimated travel times—via cellular networks. Such non-local data may potentially enhance the decision-making capabilities of adaptive cruise controllers. This chapter, therefore, broadens the discussion to include non-local observations, aiming to further improve the mitigation of stop-and-go waves and stabilize highway traffic flow presented in Chapter 2. The controller, rather than solely depending on local observations from the ego and the preceding vehicle, now also integrates information about downstream traffic conditions. Specifically, it is assumed that the controller connects to the Internet, enabling real-time access to updates from an estimated time of arrival (ETA) service provider.

3.1 Introduction

Autonomous driving technology has advanced significantly in recent years, yet it continues to face complex challenges that impede its full integration into our society. Among the issues it encounters, the dynamics of single-lane car-following represent a foundational and subtly difficult problem in the field of autonomous vehicular control. This is especially crucial as it affects traffic flow efficiency, safety, and energy consumption.

Human drivers typically exhibit car-following behaviors that are sub-optimal, leading to issues such as increased traffic congestion and higher accident rates. The concept of *string instability*, popularized in the seminal work by Swaroop [119], highlights the tendency of vehicles in a platoon to amplify speed variations, thereby generating oscillatory behaviors known as stop-and-go waves. Such phenomena have been observed in various studies, in-

cluding those by Sugiyama and Tadaki, who showed that these waves could emerge even without physical bottlenecks or overt lane changes [114, 120].

Addressing these challenges, this dissertation proposes a novel hierarchical Model Predictive Control (MPC) approach. Following conventional control architecture, this section incorporates a tri-layer structure consisting of prediction, planning, and tracking layers, significantly enhancing robustness to prediction errors. This approach is grounded in a linearly constrained quadratic programming (LCQP) framework, which offers both theoretical rigor and practical applicability.

We summarize the main contributions of the work below.

- We present a novel, efficient tri-layer MPC design for highway car-following based on a quadratic programming formulation.
- We introduce a practical prediction method utilizing an ETA estimator.
- We offer a comprehensive numerical study demonstrating the method’s effectiveness and robustness against prediction errors.

The rest of this chapter is organized as follows. In Section 3.2, we briefly present the literature context of this method. We state the problem of the two-vehicle car-following in Section 3.3. Next, we formally define and analyze the tri-layer hierarchical MPC control scheme in Section 3.4. We evaluate the performance of our MPC controller via numerical simulations in Section 3.5 and discuss the numerical results in Section 3.6.

3.2 Related Works

The quest to improve vehicle automation has led to the development of various control strategies, particularly focusing on enhancing car-following behaviors through localized and non-localized control systems. Local control methods, which typically involve direct interactions between consecutive vehicles, have seen extensive development and deployment. Pioneering works in this area, such as those by Martinez [74] and Kesting [56], have laid the groundwork for subsequent adaptive cruise control systems that prioritize safety and fuel efficiency.

Further advancements have been made by Luo, who explored a model predictive control (MPC) framework for creating a fixed-headway adaptive cruise control that optimizes fuel consumption [71]. Similarly, Kerner et al. introduced a variable-headway approach that effectively mitigates the propagation of stop-and-go waves, showcasing the potential of adaptive systems to enhance overall traffic flow [55]. Stern et al. contributed by demonstrating experimentally that a vehicle equipped with a proportional-integral controller could dampen oscillations induced by human drivers, further proving the viability of these technologies in a circular platoon setting [113].

In addition to local methods, the literature also discusses non-local control strategies that utilize broader traffic data to anticipate and adjust to conditions well ahead of the immediate

traffic scenario. Such approaches are advantageous in that they allow for anticipatory actions that can smooth overall traffic flow and reduce oscillations over larger areas. Examples include the work of Schmied and Jia, who have developed variable-headway MPC methods that leverage data from downstream traffic signals to refine car-following tactics [105, 50].

Despite their potential, non-local methods require more sophisticated infrastructure and communication systems, such as vehicle-to-infrastructure and vehicle-to-vehicle interactions, which introduce new layers of complexity and potential points of failure [17, 105, 50]. The performance of these systems is highly dependent on the accuracy of the prediction models used, and the literature indicates a need for improved methods to handle prediction errors effectively [46].

This dissertation builds on these foundational studies but advances the field by proposing a more integrated and robust approach to managing the inherent uncertainties of prediction in dynamic traffic environments.

3.3 Problem Statement

Consider a two-vehicle setup, where a considered vehicle follows a preceding vehicle that travels along a path of length L . Denote the initial and final times of the trip as t_0 and t_f , respectively.

Let the state of a vehicle be $\mathbf{x} := [s \ v]^\top$, where s and v represent the position and speed of a vehicle, respectively. Define a as the acceleration of a vehicle. Let ℓ be the length of a vehicle. We use superscript to identify to which vehicle a variable belongs, where the considered vehicle is indexed as one and the preceding vehicle zero. Thus, $\mathbf{x}^1$ refers to the state of the considered vehicle. Let the initial position of the considered vehicle be w_0 , that is, $s^1(t_0) := w_0$. Let the initial position of the preceding vehicle be w_1 , that is, $s^0(t_0) := w_1$. Let $w_f := w_1 + L$. Then, $s^0(t_f) := w_f$. The two-vehicle scenario is illustrated in Figure 3.1.

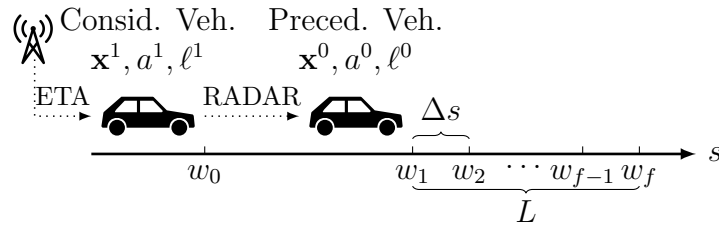


Figure 3.1: Scenario of the two-vehicle car-following problem. At t_0 , the considered vehicle is at w_0 and the preceding vehicle at w_1 . The preceding vehicle travels through a set of way points $(w_1, w_2, \dots, w_f)$, where each pair of consecutive way points are spaced Δs apart, barring boundary condition.

For the considered vehicle to properly follow the preceding vehicle, we impose the following five requirements. First and foremost, to prevent collision, we constrain the space head-

way between the two vehicles to be greater than some minimal headway $h_{\min}(t) : \mathbb{R} \rightarrow \mathbb{R}_{\geq 0}$. Next, to *follow* the preceding vehicle, we constrain the space headway to be smaller than some maximum headway $h_{\max}(t) : \mathbb{R} \rightarrow \mathbb{R}_{\geq 0}$. The third constraint requires the speed of the considered vehicle to fall within some speed limits $[v_{\min}, v_{\max}]$. Next, we constrain the acceleration of the considered vehicle to be bounded within some range $[a_{\min}, a_{\max}]$. Last but not least, among all trajectories that satisfies the constraints above, we select one that is the smoothest, as measured by ℓ^2 -norm of the acceleration.

We summarize the above specifications in Problem 3.

Problem 3. Provided $s^0(t)$ for $t \in [t_0, t_f]$ with some finite t_f , determine a trajectory of least acceleration in ℓ^2 -norm for the considered vehicle $s^1(t)$ for $t \in [t_0, t_f]$ such that 1) $\min_{\forall t} s^0(t) - s^1(t) \geq h_{\min}(t)$, 2) $\max_{\forall t} s^0(t) - s^1(t) \leq h_{\max}(t)$, 3) $v^1(t) \in [v_{\min}, v_{\max}]$ for all t , and 4) $a^1(t) \in [a_{\min}, a_{\max}]$ for all t .

With the problem stated above, we present the proposed controller in the next section.

3.4 Optimal Car-Following Controller

In this section, we present a hierarchical MPC scheme consisting of 1) a prediction layer, 2) a planning layer, and 3) a tracking layer. The layered controller design is illustrated in Figure 3.2. We present the details of the three layers below.

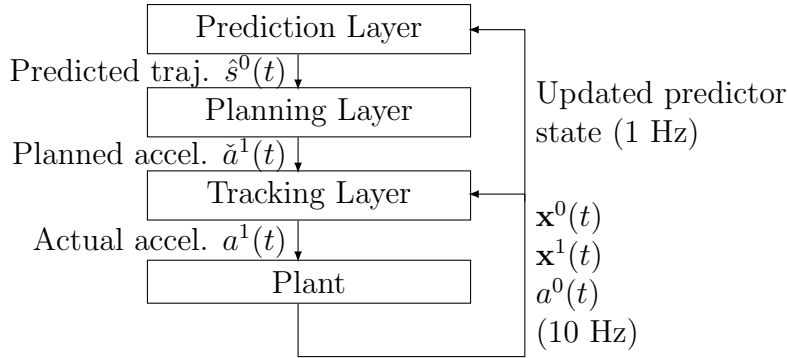


Figure 3.2: Diagram of the MPC controller design. Prediction layer and planning layer are invoked every 1 second. Tracking layer is triggered every 0.1 second.

3.4.1 Prediction Layer

The prediction layer predicts the trajectory of the preceding vehicle based on an ETA estimator. Because the ETA estimator uses downstream information, it renders the controller non-local.

Define a way point every Δs from $s^0(t)$ to $s^0(t) + l \cdot \Delta s$ at some time $t \in [t_0, t_f]$ and for some spatial receding horizon $l \cdot \Delta s > 0$. Non-integer l indicates that the spatial horizon cannot be evenly divided by Δs . Collect all way points into a sequence, we have $\mathcal{W}(t) := (w_1(t), w_2(t), \dots, w_{l+1}(t)) = (s^0(t) + i \cdot \Delta s \mid i = 0, 1, \dots, l)$. For convenience, we assume $w_{l+1} \leq w_f$.

Define $\hat{\mathcal{T}}^0(t)$ as the corresponding ETA set for the preceding vehicle, that is, $\hat{\mathcal{T}}^0(t) := (t_1^0(t), \hat{t}_2^0(t), \dots, \hat{t}_{l+1}^0(t))$, where $t_i^0, \hat{t}_i^0$ are the *true* and *estimated* arrival times of the preceding vehicle at the way point $w_i \in \mathcal{W}(t)$. For the problem to be well-posed, we require both true and estimated arrival time sequences to be strictly increasing.

We assume there exists an ETA estimator¹ $f : \mathcal{W}(t) \rightarrow \hat{\mathcal{T}}^0(t)$ that estimates the *arrival* time of the preceding vehicle at each way point in $\mathcal{W}(t)$. To model prediction errors, we assume that the relative estimation error is *uniformly* distributed with an average at one and a radius of σ , that is,

$$\frac{\hat{t}_{i+1}^0 - \hat{t}_i^0}{t_{i+1}^0 - t_i^0} \sim \mathcal{U}(1 - \sigma, 1 + \sigma), \quad i = 1, 2, \dots, l,$$

where $\hat{t}_1^0 = t_1^0$. We use f_σ to denote that an ETA estimator f has an error radius of σ .

With $\mathcal{W}(t)$ and $\hat{\mathcal{T}}^0(t) = f_\sigma(\mathcal{W}(t))$, we can then generate a *predicted* trajectory for the preceding vehicle $\hat{s}^0 : [t_1^0, \hat{t}_l^0] \rightarrow [s^0(t), s^0(t) + l \cdot \Delta s]$ using a standard interpolation method. The above receding horizon design is illustrated in the s and $\hat{t}^0$ axes of Figure 3.3.

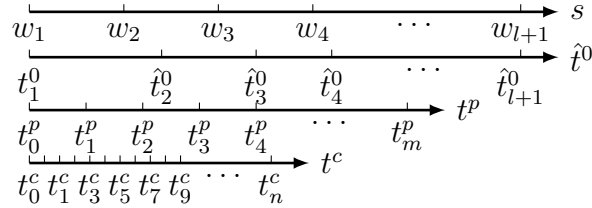


Figure 3.3: Specification of the receding-horizon control scheme. The prediction layer has a *fixed* spatial horizon and a *variable* temporal horizon. The planning layer has a temporal horizon m . The tracking layer has a fixed temporal horizon n .

3.4.2 Planning Layer

The planning layer plans a reference trajectory for the considered vehicle based on the predicted trajectory of the preceding vehicle. With $\hat{s}^0(t)$ predicted by f_σ , we can derive the following four trajectories: 1) $\hat{s}_t^-(t) := \hat{s}^0(t - \Delta t^-)$: predicted envelope of some minimal time headway Δt^- , 2) $\hat{s}_t^+(t) := \hat{s}^0(t - \Delta t^+)$: predicted envelope of some maximal time headway

¹Common ETA service providers include Google Maps, Waze, INRIX, and Garmin.

Δt^+ , 3) $\hat{s}_s^-(t) := \hat{s}^0(t) - \Delta s^-$: predicted envelope of some minimal space headway Δs^- , and 4) $\hat{s}_s^+(t) := \hat{s}^0(t) - \Delta s^+$: predicted envelope of some maximal space headway Δs^+ . The above four trajectories are illustrated in Figure 3.4.

While many possible designs are possible, in this study we select the minimal and maximal headway envelopes as follows

$$\begin{aligned}\hat{s}_{\min}(t | \mathcal{W}, \hat{\mathcal{T}}^0) &:= \max(\min(\hat{s}_s^-(t), \hat{s}_t^-(t)), \hat{s}_s^+(t)), \\ \hat{s}_{\max}(t | \mathcal{W}, \hat{\mathcal{T}}^0) &:= \min(\max(\hat{s}_s^+(t), \hat{s}_t^+(t)), \hat{s}_s^-(t)).\end{aligned}\quad (3.1)$$

The area between $\hat{s}_{\min}(t)$ and $\hat{s}_{\max}(t)$ is shaded in green in Figure 3.4. Consequently, the minimum and maximum headways are

$$\begin{aligned}h_{\min}(t) &= \hat{s}^0(t) - \hat{s}_{\min}(t | \mathcal{W}, \hat{\mathcal{T}}^0), \\ h_{\max}(t) &= \hat{s}^0(t) - \hat{s}_{\max}(t | \mathcal{W}, \hat{\mathcal{T}}^0).\end{aligned}\quad (3.2)$$

The objective of the design in (3.1) is to generate smooth car-following without falling far behind nor getting too close. Compared to [105, 50], our formulation is stricter, which leads to a tighter yet still safe and smooth car-following.

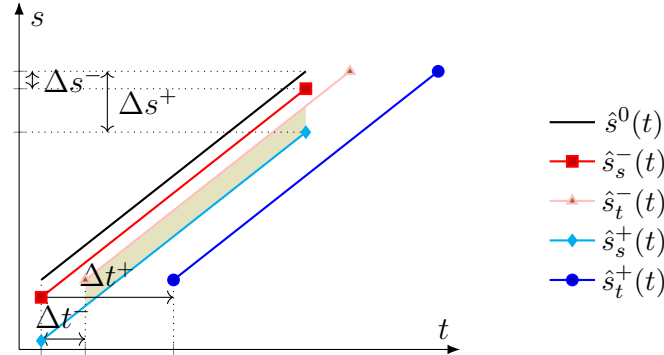


Figure 3.4: Headway envelope design. $\hat{s}^0(\cdot)$ is the predicted trajectory of the proceeding vehicle. $\hat{s}_s^-(\cdot), \hat{s}_s^+(\cdot)$ are estimated minimum and maximum *space* headway envelopes, respectively. $\hat{s}_t^-(\cdot), \hat{s}_t^+(\cdot)$ are estimated minimum and maximum *time* headway envelopes, respectively. The shaded region represents the feasible set in space and time.

Now, let m be the temporal planning horizon and Δt_p be the temporal planning resolution. For all $i_p = 0, \dots, m$, denote $t_{i_p}^p := t_1^0 + i_p \Delta t_p$. Non-integer m indicates that $\hat{t}_m^0 - t_1^0$ cannot be evenly divided by Δt_p .

Define the planned states $\tilde{\mathcal{X}}^{1p}$ and the planned accelerations $\tilde{\mathcal{U}}^{1p}$ of the considered vehicle as follows:

$$\begin{aligned}\tilde{\mathcal{X}}_m^{1p} &:= \{\tilde{\mathbf{x}}^1(t_0^p), \tilde{\mathbf{x}}^1(t_1^p), \dots, \tilde{\mathbf{x}}^1(t_m^p)\}, \\ \tilde{\mathcal{U}}_{m-1}^{1p} &:= \{\tilde{a}^1(t_0^p), \tilde{a}^1(t_1^p), \dots, \tilde{a}^1(t_{m-1}^p)\}.\end{aligned}\quad (3.3)$$

Here, we introduce the check accent $\check{x}$ to denote that a variable x is a *planned* decision variable.

With everything defined above, we plan for the states and accelerations of the considered vehicle via the following LCQP:

$$\min_X \quad \alpha \sum_{i_p=0}^{m-1} \left(\check{a}^1(t_{i_p}^p) \right)^2 + \beta \sum_{j_p=1}^m \left(\xi_{j_p}^p \right)^2 + \gamma \sum_{j_p=1}^m \left(\zeta_{j_p}^p \right)^2 \quad (3.4a)$$

$$\text{s.t.} \quad \check{\mathbf{x}}^1(t_0^p) = \mathbf{x}_0^1, \quad (3.4b)$$

$$\check{\mathbf{x}}^1(t_{i_p+1}^p) = \mathbf{A}_{i_p}^p \cdot \check{\mathbf{x}}^1(t_{i_p}^p) + \mathbf{B}_{i_p}^p \cdot \check{a}^1(t_{i_p}^p), \quad (3.4c)$$

$$0 \leq \xi_{j_p}^p, \quad \check{s}^1(t_{j_p}) - \hat{s}_{\min}(t_{j_p} \mid \mathcal{W}, \hat{\mathcal{T}}^0) \leq \xi_{j_p}^p, \quad (3.4d)$$

$$0 \leq \zeta_{j_p}^p, \quad -\check{s}^1(t_{j_p}) + \hat{s}_{\max}(t_{j_p} \mid \mathcal{W}, \hat{\mathcal{T}}^0) \leq \zeta_{j_p}^p, \quad (3.4e)$$

$$v_{\min} \leq \check{v}^1(t_{j_p}^p) \leq v_{\max}, \quad (3.4f)$$

$$a_{\min} \leq \check{a}^1(t_{i_p}^p) \leq a_{\max}, \quad (3.4g)$$

where

$$\begin{aligned} X &:= \left\{ \check{\mathcal{X}}_m^{1p}, \check{\mathcal{U}}_{m-1}^{1p}, \{\xi_{j_p}^p\}_1^m, \{\zeta_{j_p}^p\}_1^m \right\}, \\ \mathbf{A}_{i_p}^p &:= e^{\mathbf{A}(t_{i_p+1}^p - t_{i_p}^p)}, \quad \mathbf{A} = \begin{bmatrix} 0 & 0 \end{bmatrix}^\top \begin{bmatrix} 1 & 0 \end{bmatrix}^\top, \\ \mathbf{B}_{i_p}^p &:= \int_{t_{i_p}^p}^{t_{i_p+1}^p} e^{\mathbf{A}(t_{i_p+1}^p - \tau)} d\tau \cdot \mathbf{B}, \quad \mathbf{B} = \begin{bmatrix} 0 & 1 \end{bmatrix}^\top, \end{aligned}$$

for $i_p = 0, \dots, m-1$ and $j_p = 1, \dots, m$. The receding horizon design is illustrated in the t^p axis of Figure 3.3. We have $\mathcal{W}, \hat{\mathcal{T}}^0, \alpha, \beta, \gamma$, and $\mathbf{x}_0^1$ as the inputs to the optimization program, where $\alpha, \beta, \gamma \in \mathbb{R}_{>0}$, $\alpha + \beta + \gamma = 1$, and $\mathbf{x}_0^1$ is the initial state of the considered vehicle.

In (3.4a), the first term penalizes large accelerations, while the second and third term, coupled with (3.4d) and (3.4e), regulate the vehicle inside the admissible space headway ranges. Constraints (3.4b) and (3.4c) impose kinematic constraints using zero-order hold. Lastly, constraints (3.4f) and (3.4g) impose speed and acceleration limits.

Proposition 1. If the feasible set is nonempty, problem (3.4) has unique global optimum.

Proof. Because $\alpha, \beta, \gamma > 0$, the objective function (3.4a) is strongly convex. By construction, the feasible set is a polyhedron, which is also convex. Therefore, a direct application of Lemma 8.2 and Theorem 8.6 in [18] reveals that if the feasible set is nonempty, problem (3.4) has an unique global optimum. $\square$

Proposition 2. For some initial time t_0 , if the feasible set of problem (3.4) is nonempty, then its corresponding receding-horizon policy has a solution for all $t > t_0$, that is, it is persistently feasible.

Proof. To show that problem (3.4) is persistently feasible, it suffices to show that it has a solution at every future time step. Because the feasible set is initially nonempty, the problem has a solution at t_0 . For every other time step $t > t_0$, it is easy to verify that zero acceleration is a feasible point due to the fact that constraints (3.4d) and (3.4e) are soft. Therefore, it follows that problem (3.4a), when repeatedly applied in receding horizon, always stays feasible. $\square$

Remark 1. When the initial velocity is too high or too low, the feasible set of (3.4) will become empty. For example, when the initial velocity is greater than $v_{\max} + a_{\max}\Delta t^p$, there will be no acceleration within the actuation limits that can steer the velocity below $v_{\max}$ in the next time step. To prevent any chance of infeasibility, we can replace the hard constraints on velocity with suitable soft constraints.

Remark 2. Because the constraints on minimum headway are soft, it is possible for the considered vehicle to violate constraint (3.4d). Therefore, it is theoretically permissible for the preceding vehicle to collide with the preceding vehicle. Note that this is chosen by design because the hard constraint version of (3.4d) can never be guaranteed in practice due to irregularity of human driving and inevitable errors in prediction. Therefore, the presented soft constraints in (3.4d) should be viewed as a “best-effort” attempt to respect the minimum headway requirement.

Furthermore, we briefly comment on how constraints (3.4d) and (3.4e) can be extended to account for cut-ins and cut-outs. For example, a possible design is illustrated in Figure 3.5. Under this design, the considered vehicle will not overreact when a discontinuous change occurs in the position of the preceding vehicle. Rather, it gradually recovers to a comfortable headway over time. With the help a switching condition, this policy can be combined with that of (3.1) to form a hybrid MPC controller to perform car-following with cut-ins and cut-outs.

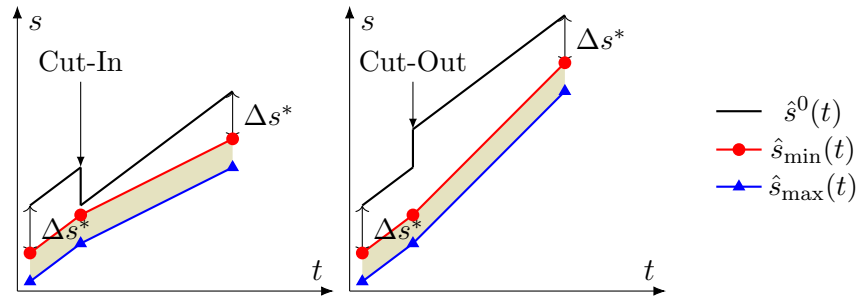


Figure 3.5: Headway envelope design to handle cut-ins and cut-outs. Left design is for cut-ins and right for cut-outs. The proposed designs aim to keep a comfortable headway Δs^* without overreaction.

3.4.3 Tracking Layer

To handle prediction errors, an additional tracking layer is introduced to track the planned accelerations and to guard the considered vehicle from imminent collisions.

Concretely, the layer takes the following form, which is a modified version of (3.4).

$$\min_Z \quad \lambda \sum_{i_c=0}^{n-1} (\bar{a}^1(t_{i_c}^c) - \bar{a}^1(t_{i_c}^c))^2 + \mu \sum_{j_c=1}^n (\xi_{j_c}^c)^2 \quad (3.5a)$$

$$\text{s.t.} \quad \bar{\mathbf{x}}^1(t_0^c) = \mathbf{x}_0^1, \quad (3.5b)$$

$$\bar{\mathbf{x}}^1(t_{i_c+1}^c) = \mathbf{A}_{i_c}^c \cdot \bar{\mathbf{x}}^1(t_{i_c}^c) + \mathbf{B}_{i_c}^c \cdot \bar{a}^1(t_{i_c}^c), \quad (3.5c)$$

$$0 \leq \xi_{j_c}^c, \quad s^1(t_{j_c}^c) - \tilde{s}_{\min}(t_{j_c}^c \mid a^0(t_0^c)) \leq \xi_{j_c}^c, \quad (3.5d)$$

$$v_{\min} \leq \bar{v}^1(t_{j_c}^c) \leq v_{\max}, \quad (3.5e)$$

$$a_{\min} \leq \bar{a}^1(t_{i_c}^c) \leq a_{\max}, \quad (3.5f)$$

where

$$t_{i_c}^c := t + i_c \Delta t^c, \quad Z := \{\bar{\mathbf{x}}_n^{1c}, \bar{\mathbf{u}}_{n-1}^{1c}, \{\xi_{j_c}^c\}_1^n\},$$

$$\mathbf{A}_{i_c}^c := e^{\mathbf{A}(t_{i_c+1}^c - t_{i_c}^c)}, \quad \mathbf{B}_{i_c}^c := \int_{t_{i_c}^c}^{t_{i_c+1}^c} e^{\mathbf{A}(t_{i_c+1}^c - \tau)} d\tau \cdot \mathbf{B},$$

for some horizon n , resolution Δt^c , $i_c = 0, \dots, n-1$, and $j_c = 1, \dots, n$. $\bar{\mathbf{x}}_n^{1c}, \bar{\mathbf{u}}_{n-1}^{1c}$ are defined similar to those in (3.3). Note that here we use accent $\bar{x}$ to indicate that a variable x is an internal variable of the tracking controller. The receding horizon design is illustrated in the t^c axis of Figure 3.3. Unlike the planning layer, the minimum headway envelope $\tilde{s}_{\min}(\cdot \mid a^0(\cdot))$ is generated by assuming that the preceding vehicle accelerates *constantly* from t_0^c to t_n^c . Similar to the planning layer once again, we have $a^0(t_0)$, λ, μ , and $\mathbf{x}_0^1$ as inputs to the optimization program, where $\lambda, \mu \in \mathbb{R}_{>0}$, $\lambda + \mu = 1$.

The modified LCQP in the tracking layer in (3.5) is structurally identical to (3.4) except for: 1) we remove the soft constraint on maximum headway; 2) we adopt a shorter planning horizon n and a smaller temporal resolution Δt^c ; and 3) we predict the trajectory of the preceding vehicle by assuming that it keeps its current acceleration for the entire tracking horizon.

Note that the last change above is made to ensure safety. Because constraints (3.5d) only takes vehicular RADAR measurements, which are generally highly reliable, the collision avoidance of the car-following is mostly *decoupled* from the ETA estimation. Hence, a bad prediction from the ETA estimator cannot severely impact the safety of the vehicle.

Remark 3. Because the optimization problem (3.5) may be viewed as a special case of problem (3.4) where $\gamma = 0$, it has the same properties as those highlighted in Propositions 1,2 and Remarks 1,2.

Remark 4. We assume that we can directly control the acceleration of the vehicle. Consequently, dynamic constraints (3.5c) are purely *kinematic*.

Lastly, we briefly describe how our proposed method can be extended for multi-vehicle platooning. For example, consider a platoon of vehicles with indices $0, 1, \dots, k$, where index 0 indicates the preceding human vehicle and indices $1, \dots, k$ indicate the k following automated vehicles. As before, we can use an ETA estimator to generate a prediction for vehicle zero. For all the vehicles that *follow*, we can substitute the predicted trajectories of the preceding vehicles by the corresponding planned trajectories. That is, for $i = 1, 2, \dots, k$, we can use $\check{a}^{i-1}(t)$ to compute $\check{s}^{i-1}(t)$ and approximate $\hat{s}^{i-1}(t)$ with $\check{s}^{i-1}(t)$. Integrating the above prediction method with the planning and tracking methods presented in this paper, we have consequently produced a complete platoon controller design.

3.5 Numerical Simulation

To evaluate the proposed controller, we test its performance through an array of numerical experiments, each equipped with an ETA estimator of an unique combination of spatial resolution Δs and noise level σ .

3.5.1 Simulation Setup

For all simulations, we emulate the preceding vehicle by replaying a recorded drive from [83]. The velocity and acceleration of the recorded drive are shown in the second and third rows of Figure 3.6.

Spatial horizon of the ETA estimators are fixed to $l \cdot \Delta s = 3000$ m. We run an array of simulations by varying the spatial resolution Δs between 10 m and 500 m and noise level σ between 0.01 and 0.25. To establish a performance baseline, we run an additional simulation with an intelligent driver model (IDM) controller [123]. Parameters of the IDM are chosen to roughly match that of the MPC controller: $a = 1.5$, $b = 3$, $\delta = 4$, $s_0 = 3.5$, $\ell = 4.65$. Likewise, to establish a performance upper bound, we run one more simulation with a controller that can perfectly foresee the trajectory of the preceding vehicle. We call this controller the oracle controller.

Shared parameters of the planning and tracking layers are: $v_{\min} = 0$, $v_{\max} = 35 \text{ m} \cdot \text{s}^{-1}$, $a_{\min} = -1.5 \text{ m} \cdot \text{s}^{-2}$, $a_{\max} = 3 \text{ m} \cdot \text{s}^{-2}$, $\Delta s^- = 5 \text{ m}$, $\Delta s^+ = 100 \text{ m}$, $\Delta t^- = 0.6 \text{ s}$, and $\Delta t^+ = 3.0 \text{ s}$. Unique parameters of the planning layer are: $\alpha = 0.2$, $\beta = 0.7$, $\gamma = 0.1$, $\Delta t^p = 1 \text{ s}$, and $m = 60$. Unique parameters of the tracking layer are: $\lambda = 0.1$, $\mu = 0.9$, $\Delta t^c = 0.1 \text{ s}$, and $n = 30$.

To evaluate performance, we use two measures e and f to quantify tracking error and fuel saving, respectively. The measure e is defined to be the standard deviation of $\check{v}^{1*} - \bar{v}^1$ for all sampled time instants, where $\check{v}^{1*}$ denotes the planned velocity under *perfect* prediction. The measure f is defined to be the ratio of the fuel consumption of the baseline IDM controller to that of the MPC controller, using the energy model described in [65].

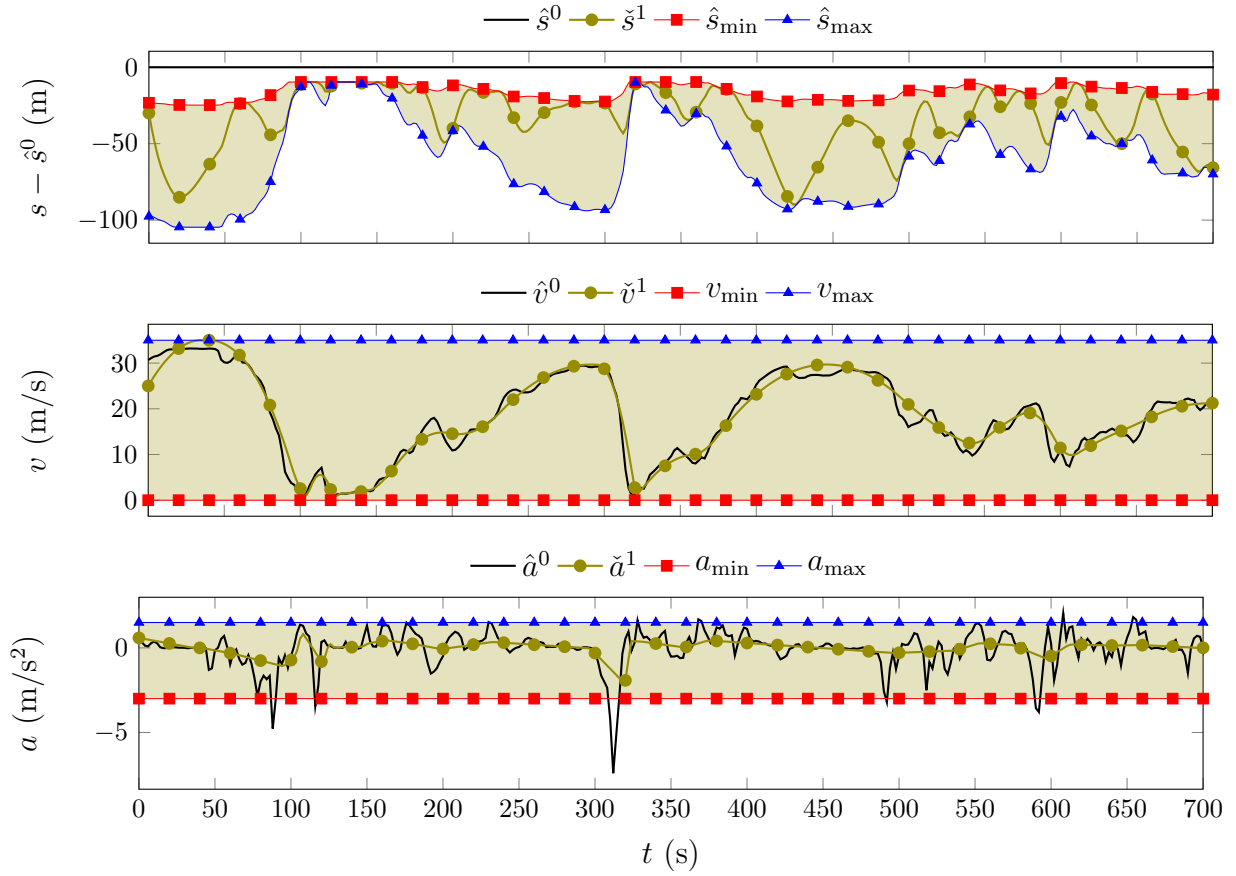


Figure 3.6: Demonstration of the planning layer. The oracle controller assumes perfect prediction over the entire planning horizon.

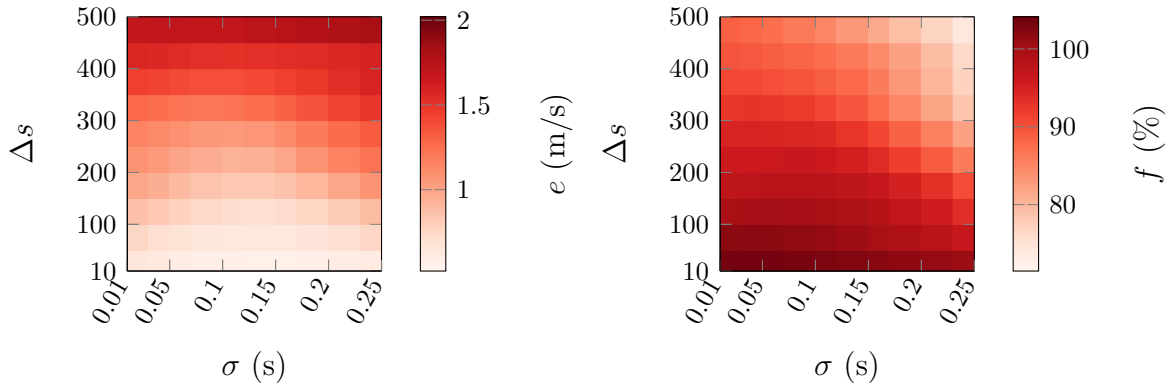


Figure 3.7: Performance heat maps of the MPC controllers. The top heat map measures tracking error e , while the bottom heat map measures fuel saving f .

3.5.2 Numerical Results

Position, velocity, and acceleration of the oracle controller are shown in Figure 3.6. As the oracle vehicle has demonstrated in Figure 3.6, the optimal behavior is to avoid accelerating or decelerating unless there is an imminent violation of at least one of the two headway constraints.

Tracking error and fuel saving are visualized in the top and bottom heat maps in Figure 3.7. As expected, tracking error e is proportional to spatial resolution Δs and noise level σ . Likewise, fuel saving f is *inversely* proportional to spatial resolution Δs and noise level σ . Note that both performance measures plateau near the origin, indicating that the MPC controller is robust to prediction errors and especially so to that induced by σ .

Lastly, we comment on the running times of the planning and tracking layers. The optimization solver is part of a python package called CVXOPT [5]. Benchmarked on an Intel i7-6700K CPU, the average running time of the planning layer is about 0.12 s and of the tracking layer about 0.04 s. From Figure 3.2, we find that both times fit comfortably into their allocated time budgets.

3.6 Discussion

In this section, we discuss two of the most consequential features in the design of the controller, namely, 1) the space between the maximum and minimum headway constraints and 2) the resolution and noise level of the ETA estimators.

3.6.1 Headway Constraints Design

It is clear that the larger the space between the maximum and minimum headway constraints, the smoother the car-following could be. For example, if the considered car is allowed to be arbitrarily far behind the preceding vehicle without incurring any penalties, it can simply wait for the preceding vehicle to exit the road and then accelerate to a very small constant velocity to complete the trip. Nevertheless, we know from common sense that such behavior is not acceptable in most of the real-world applications.

On the contrary, when the space between the two headway constraints is small, the MPC controller will become sensitive to prediction errors. Consider a situation where the allowable headway gap is thin and the prediction errors are significant. Because the considered vehicle is already close to the boundaries of the headway constraints, an error in prediction could easily mislead the planning layer to falsely believe that it will soon violate one of the headway constraints. In an effort to steer away from imminent constraint violation, the planning layer overreacts, leading to undesirably large acceleration or deceleration.

To resolve the above problem, one could modify the headway constraints proposed in (3.1). One possible solution is to enforce a minimum space gap between the maximum and minimum headway envelopes and to add a small penalty to encourage the considered vehicle to drive at the center of the allowable headway constraints.

3.6.2 ETA Estimator Choice

Last but not least, we comment on how to choose an ETA estimator. As illustrated in Figure 3.7, the performance of the controller changes significantly with the resolution and noise level of the ETA estimator. In practice, one may first define a desirable tracking error e_{des} and a desirable fuel saving f_{des} . With e_{des} and f_{des} defined, one can then draw two level sets in the two heat maps of Figure 3.7. Any ETA estimator with e and f that simultaneously fall within the two level sets may be deemed sufficient to meet the design specifications.

Chapter 4

Partially Meshed Multi-Agent Systems

The setup of sequentially chained MASs typically occurs in highly structured environments, where a line of agents is artificially constructed by design protocols. In contrast, partially meshed MASs often emerge in unstructured or understructured environments, where connections organically form based on the real-time vicinity of neighboring agents. Examples of partially meshed MASs include interactions among pedestrians, both artificial and biological swarming behaviors, and ad-hoc networks. This chapter explores planning and control problems in partially meshed MASs, focusing on decentralized collision avoidance through the use of a consensus-based framework. The chapter begins by proposing a consensus-based framework designed to model negotiative driving behaviors in understructured environments. Subsequently, we introduce a novel aerial video dataset that captures the intricate human dynamics in these environments. Finally, we validate our consensus-based framework by testing a consensus control method within the framework through both numerical simulations and the dataset.

4.1 Introduction

Navigation in built environments, such as driving, cycling, and walking, is a crucial area of research in autonomous driving and human-robot interaction. The development of autonomous agents capable of operating in *structured* environments is a well-established field, with roots that extend back to the early days of control theory. Meanwhile, researchers have extensively studied human behaviors in structured environments, including free-flow highways and signalized urban streets.

As detailed in [88], the modern control and planning hierarchy in an autonomous vehicle consists of four layers of abstraction, from top to bottom: 1) routing, 2) behavioral decision-making, 3) motion planning, and 4) vehicle control. Particularly in the second behavioral layer, a prediction module is often employed to forecast the motion of surrounding vehi-

cles, around which the motion planning layer then plans to avoid collisions. This classical predict-then-plan paradigm, prevalent in automated driving as exemplified in Chapter 3, is inadequate for collision resolution in understructured road environments where there are no explicit signs or central authority to coordinate conflicting traffic. For instance, at an unsignalized intersection, drivers negotiate the right-of-way dynamically: when two conflicting vehicles approach the intersection at similar times, the one perceived as more aggressive often “wins” the priority of passage. This negotiative process fundamentally deviates from the predict-then-plan approach.

Modeling such a negotiative process holds significant practical value for the autonomous driving community. Given the vast population of drivers in developing countries, where roads are often unsignalized, understanding how humans operate in these environments is crucial for enhancing the generalizability of autonomous driving technologies. This knowledge enables autonomous vehicles to mimic human drivers, facilitating navigation through understructured road environments. Furthermore, it may inspire innovative designs of decentralized motion planning algorithms, applicable not only to autonomous vehicles but also to other forms of mobile robots.

In this work, we propose a novel planning approach for the behavioral decision-making layer, replacing the traditional predict-then-plan approach with a negotiate-then-commit strategy. Central to this new method is a consensus model that arbitrates the priority of agents with conflicting paths. This consensus model is founded on the principle of least effort, a universal law observed across physical and biological worlds.

In addition to the modeling method, another essential ingredient for understanding human behaviors is the study of empirical data. In this context, camera videos are particularly effective because 1) they capture rich dynamics on roads at a relatively low cost, 2) they allow for quantitative assessment through direct inspection, and 3) they are suitable for qualitative analysis using computer vision techniques. Despite extensive past research, most existing video datasets focus solely on human behaviors in structured environments. Behaviors in *understructured* road environments, such as highways with collisions and unsignalized intersections, have rarely been surveyed.

As mentioned previously, understanding how humans coordinate, i.e., the implicit “social etiquette,” has profound impacts. The scarcity of data in this domain undoubtedly hinders its development. To bridge this gap in empirical data, we introduce the Berkeley DeepDrive Drone (B3D) dataset. This inertial-framed dataset records rich dynamics of human driving behaviors in understructured road environments, including unsignalized intersections, unsignalized roundabouts, highways with collisions, highways with stop-and-go waves, and highways with merging bottlenecks. To the best of our knowledge, it is the first inertial-framed dataset that extensively covers understructured driving behaviors.

To summarize, we list the main contributions of our work below.

- We propose a decentralized conflict resolution framework based on a consensus-based strategy.

- We develop and release an open-source dataset that captures rich dynamics of human driving behaviors in understructured road environments.
- We validate our method numerically through simulations and using the collected dataset.

The rest of the article is organized as follows. We provide a brief literature review in Section 4.2. We then formulate our problem in Section 4.3, describe the consensus-based conflict resolution framework in Section 4.4, and present the details of the dataset in Section 4.5. Next, we validate our method through simulation studies in Section 4.6. Lastly, we remark on several theoretical and practical aspects of our work in Section 4.7.

4.2 Related Works

Consensus problems are a fundamental area of study in various engineering disciplines, playing a critical role in enabling distributed systems to operate efficiently and effectively. These problems are ubiquitous in practical applications, such as formation control [97], distributed sensing [127], decentralized task allocation [23], and multi-agent coordination [49].

In the specific context of MARSs, consensus models are useful for solving complex modeling and coordination challenges. These models facilitate the understanding and simulation of diverse behaviors and interactions, ranging from pedestrian behavior modeling to air traffic control and swarm behavior modeling. For instance, the social force model has proven effective in capturing and describing real-world pedestrian behaviors, offering insights that are critical for autonomous driving and crowd management [41, 104]. In the field of air traffic management, consensus auction models have been employed to enhance efficiency and safety, demonstrating their utility in high-stakes environments [121, 24]. Furthermore, consensus algorithms are applied to describe and control swarm behaviors in natural settings, as well as in engineered systems, thus supporting both biological research and technological development [98, 124].

These diverse applications underscore the versatility and significance of consensus models in MARSs, highlighting their role in advancing both theoretical understanding and practical implementations in various engineering domains.

Most of the existing camera datasets can be classified into either *body-framed* dataset or *inertial-framed* dataset. The body-framed datasets have cameras placed on the traffic participants, e.g., on top a survey vehicle, to observe the movements of the surrounding traffic. The inertial-framed datasets have cameras installed at some overhead positions above some roads of interest to observe all road occupants within its field of view.

Body-framed datasets are useful to study behaviors of the surrounding traffic participants with respect to the host. It is arguably the most commonly used method for autonomous driving perception research. Many well-known datasets fall within this category, such as the KITTI dataset [35], the Oxford RoboCar dataset [72], the Waymo Open dataset [117], and the Waymo Open Motion dataset [30]. Nevertheless, the body-framed placement makes it

difficult to study persistent traffic patterns within a fixed spatial range. Moreover, placing the sensors on a low-profile moving object also results in undesirable occlusion.

Inertial-framed datasets are better suited to observe traffic within a fixed spatial range. By construction, the inertial-framed placement makes it very easy to estimate the recorded vehicle and human movements with respect to the ground. Because the cameras are often placed on a vantage point, occlusion is also not as problematic. In this category, we also find many seminal datasets, including the NGSIM dataset [1], the ARED dataset [133], the Stanford Drone dataset [104], the highD dataset [62], and the INTERACTION dataset [136].

4.3 Problem Statement

We consider a setup with N agents and potentially M pairs of conflicting paths. The objective is to determine a traversal policy for each agent that commands them to reach their destinations efficiently and without collisions using only states of neighboring agents.

Denote the road network by $G = (V, E)$, where V represents the set of vertices and E the set of edges. A path p is defined as a finite sequence of homogeneously directed edges connecting a sequence of unique vertices. Assuming $v_1, v_2, \dots, v_n$ is the vertex sequence of the path, with v_1 as the origin and v_n the destination, we denote the path as $p = \{v_1, v_2, \dots, v_n\}$. A trajectory $\{\mathbf{x}(t) \mid \forall t\}$ is defined as the evolution of an agent's state over time.

Let $s \in \mathbb{R}_{\geq 0}$ represent the traveled distance along path p , and assume the state of the agent can be uniquely indexed by s , i.e., $\mathbf{x} = \phi(s)$, where $\phi : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}^n$ is the index function. An traversal policy π for the path p describes the evolution of an agent's distance traveled along the path over time. Assuming the agent begins its path at t_0 and completes it at t_f , the traversal policy can be defined as a function $s = \pi(t)$, where $\pi : [t_0, t_f] \rightarrow \mathbb{R}_{\geq 0}$ maps time to the distance traveled. Consequently, the agent's state at any time t is given by $\mathbf{x}(t) = \phi(\pi(t))$.

Two traversal policies $\pi_1 : [t_{0|1}, t_{f|1}] \rightarrow \mathbb{R}_{\geq 0}$ and $\pi_2 : [t_{0|2}, t_{f|2}] \rightarrow \mathbb{R}_{\geq 0}$ are considered collision-free if the resulting trajectories $\mathbf{x}_1(t)$ and $\mathbf{x}_2(t)$ remain sufficiently separated according to some distance function $d : \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \rightarrow \mathbb{R}_{\geq 0}$ for all relevant times. Specifically, we require that $d(\mathbf{x}_1(t), \mathbf{x}_2(t)) \geq \epsilon$ for a certain separation margin $\epsilon > 0$ and for all t in the overlap of their time intervals $[t_{0|0}, t_{f|0}] \cap [t_{0|1}, t_{f|1}]$.

If two paths, p_1 and p_2 , intersect at only a single vertex $v^\times$, a set of collision-free policies, π_1 and π_2 , will both necessarily and sufficiently mandate a priority order. In this priority order, exactly one of the two agents will pass through the vertex $v^\times$ first. Similarly, if the two paths intersect at a path $p^\times$, establishing a priority order throughout the entire $p^\times$ is the necessary and sufficient condition to ensure zero collisions. Therefore, a crucial component for achieving decentralized conflict resolution is a consensus mechanism that can establish a consistent priority order among conflicting agents solely based on local observations.

With the preceding definition, we are now equipped to present the two-agent collision avoidance problem as follows:

Problem 4. (Two-Agent Collision Avoidance Problem) Consider two paths p_1 and p_2 , which intersect at K , a set comprising either one or several *isolated* vertices or *isolated* paths. Given the initial conditions $(t_{0|1}, \mathbf{x}_{0|1})$ and $(t_{0|2}, \mathbf{x}_{0|2})$, find a pair of traversal policies, π_1 and π_2 , such that they establish a priority order at each conflicting location $k \in K$ with finite arrival times $t_{f|1}$ and $t_{f|2}$.

The above two-agent problem is an special case of the more general N -agent problem. One can prove by contradiction that the N -agent problem requires that every unique pair of two agents be collision-free, as formally stated below:

Problem 5. (N -Agent Collision Avoidance Problem) Consider N agents of conflicting paths $P = \{p_1, p_2, \dots, p_N\}$. Given the initial conditions $\{(t_{0|i}, \mathbf{x}_{0|i}) : \forall i = 1, \dots, N\}$, find a set of traversal policies $\{\pi_i : \forall i = 1, \dots, N\}$, such that they solve the corresponding two-agent collision avoidance problem for each unique pair of paths in P .

4.4 Consensus-Based Conflict Resolution

The central idea of this work is a consensus-based approach to achieve decentralized conflict resolution. A consensus control strategy, informally, refers to a local control policy that, despite initial disagreements, steers each individual agent towards consensus on certain critical quantities. Particularly in the considered collision avoidance scenario, the local control method should receive the states of the ego and neighboring agents and output a traversal policy for the ego agent. The critical quantity of interest here is the priority order at all conflicting locations. In this section, we present a general framework consensus-based conflict resolution.

As an N -agent problem in 5 can be decomposed into $\binom{N}{2}$ two-agent problems in 4, it is important to first consider the two-agent problem. Without loss of generality, label the two agents as Agent 1 and Agent 2. Define the state of an agent $\mathbf{x} \in \mathbb{R}^2$ as $\mathbf{x} := [s, v]$, where s represents the distance traveled along the path, and v the speed. Let k denote the conflicting location and define a binary indicator $\kappa \in [0, 1]$ for each agent such that $\kappa = 1$ when the agent has the priority and $\kappa = 0$ otherwise.

Furthermore, given κ , we assume the availability of the traversal policies for Agent 1 and Agent 2, $\pi_1(\mathbf{x}_1 | \kappa_1)$ and $\pi_2(\mathbf{x}_2 | \kappa_2)$, respectively, as follows:

$$\begin{aligned}\dot{\mathbf{x}}_1 &= \pi_1(\mathbf{x}_1 | \kappa_1), \\ \dot{\mathbf{x}}_2 &= \pi_2(\mathbf{x}_2 | \kappa_2),\end{aligned}\tag{4.1}$$

with $\kappa_1, \kappa_2 \in [0, 1]$. A priority order is established if and only if $\kappa_1 + \kappa_2 = 1$. To facilitate the ensuing discussion, we define a two-argument function $f : A \times B \rightarrow [0, 1]$ with domain $A \times B$ as *collisionless* if for any $a, b \in A \times B$, $f(a, b) + f(b, a) = 1$.

Consequently, a consensus control function $\rho : \mathbb{R}^2 \rightarrow [0, 1]$ can take the following form:

$$\kappa = \rho(\mathbf{x}, \mathbf{x}_{\text{cpl}}),\tag{4.2}$$

where $\mathbf{x}$ and $\mathbf{x}_{\text{cpl}}$ are the states of the ego agent and the coupling agent, respectively. By definition, a valid consensus control function must be collisionless, meaning it satisfies $\rho(\mathbf{x}_1, \mathbf{x}_2) + \rho(\mathbf{x}_2, \mathbf{x}_1) = 1$.

Next, we propose a specific form of collisionless consensus function based on the principle of least action.

4.4.1 Least-Action Consensus

The principle of least action is ubiquitous in physical and biological systems. Systems adhering to this principle take the path of least resistance when presented with multiple forward-moving paths. For example, a ball falling under gravity follows the shortest path to the ground. Similarly, people traveling from one point to another tend to choose the route that minimizes travel time or costs. Inspired by this principle, we propose a novel consensus control function, denoted as ρ^* , to model behaviors of decentralized conflict resolution in unstructured road environments.

Again, consider the two-agent problem, where decisions must be made regarding κ_1 and κ_2 . This simple scenario presents two possibly valid solutions: $(\kappa_1, \kappa_2) = (1, 0)$ or $(\kappa_1, \kappa_2) = (0, 1)$. If one can associate a cost functional with each valid choice of (κ_1, κ_2) , then a least-action consensus model is feasible, which simply favors the choice associated with the least cost. As a result, once both agents concur on the same priority ordering, they are enabled to traverse the conflicting location without incident.

Note the disparity in costs between the choices can serve as a measure of confidence. The more pronounced the difference in cost between the two options, the more robust the consensus will be. Moreover, a slack variable, denoted as ι , may be incorporated to model the internal defensiveness or aggressiveness of the agents. For instance, consider two priority plans with respective costs J_1 and J_2 . One can introduce a slack variable ι such that the plan with cost J_1 is deemed favorable if and only if $J_1 < J_2 + \iota$, and vice versa. Within this framework, a negative value of ι represents a defensive agent, while a positive ι indicates an aggressive agent.

Furthermore, it is important to consider all edge cases within the consensus model. Should there be only one viable option, the model must select that particular plan. In the absence of any valid choices, the model must revert to an emergency stop as a safety measure. Additionally, in instances where the costs are identical, it becomes necessary to implement an additional mechanism for tie-breaking. Within the driving context, this may entail both vehicles simultaneously coming to a stop, thereafter proceeding based on established social norms or sign language cues.

Next, we comment on how the least-action consensus model can be generalized to N -agent setups. Assume there are M conflicting locations, each involving N_k conflicting agents for $k = 1, 2, \dots, M$. Therefore, for each conflicting location k , there are $N_k!$ possible orderings of priority. With a cost functional and traversal policies in place, the consensus model can enumerate the costs for each possible ordering at every conflicting locations and select the

ones with the least amount of costs. Similar to the two-agent setups, additional logic is necessary to handle tie-breaking, which likely depends on social conventions.

4.4.2 Traversal Policy and Cost Functional

The least-action consensus model described above relies on the selection of a traversal policy and a cost functional. While numerous designs are possible, we present a simple yet effective combination here, where the traversal policy is based on a standard car-following model, and the cost functional incorporates a quadratic cost on acceleration.

Among the many candidates for constructing the traversal policy π , a viable option is through a car-following model. A car-following model describes how a human driver behaves longitudinally on roads. In this work, we consider the following car-following model named Intelligent Driver Model (IDM) [123], $f_{\text{IDM}} : \mathbb{R}^3 \rightarrow \mathbb{R}$, as formulated below:

$$f_{\text{IDM}}(\Delta s, v_1, \Delta v) = a \cdot \left(1 - \left(\frac{v_1}{v^*} \right)^\delta - \left(\frac{s^*(v_1, \Delta v)}{\Delta s} \right)^2 \right), \quad (4.3)$$

with

$$\begin{aligned} s^*(v_1, \Delta v) &= s_{\min} + v_1 T + \frac{v_1 \Delta v}{2\sqrt{ab}}, \\ \Delta s &= s_0 - s_1 - \ell_0, \\ \Delta v &= v_1 - v_0, \end{aligned} \quad (4.4)$$

for some parameters a, b, v^*, δ, T , and $s_{\min}$, where s_0 is the lead vehicle position, s_1 the ego vehicle position, ℓ_0 the length of the lead vehicle, v_0 the speed of the lead vehicle, and v_1 the velocity of the ego vehicle.

With the car-following model defined above, one can construct traversal policies for the two conflicting agents:

- For the agent with priority that has a lead vehicle, the default car-following model can be directly applied. If the prioritized agent does not have a lead vehicle, the default car-following model can be modified for fixed-speed cruising. In the case of the IDM, this modification involves removing the $(s^*(\cdot)/\Delta s)^2$ term, as shown below:

$$f_{\text{IDM}}^{\text{cruise}}(\Delta s, v_1, \Delta v) = a \cdot \left(1 - \left(\frac{v_1}{v^*} \right)^\delta \right). \quad (4.5)$$

- For the agent without priority, a virtual lead vehicle can be introduced at the conflicting location until the prioritized vehicle has cleared that location. This virtual vehicle acts as a barrier to prevent the non-prioritized agent from passing the conflicting location, thereby preventing a potential collision.

Once the traversal policy is defined, we can adopt the standard quadratic cost to describe the effort associated with that specific policy. Let $\pi_1(\cdot \mid 1)$ and $\pi_2(\cdot \mid 0)$ be the traversal

policies associated with $(\kappa_1, \kappa_2) = (1, 0)$, and $\pi_1(\cdot \mid 0)$ and $\pi_2(\cdot \mid 1)$ with $(\kappa_1, \kappa_2) = (0, 1)$. For each combination of traversal policies, the quadratic cost is defined as:

$$J(\pi_1, \pi_2; t_{0|1}, t_{0|2}, \mathbf{x}_{0|1}, \mathbf{x}_{0|2}, t_{\times|1}, t_{\times|2}) = \int_{t_{0|1}}^{t_{\times|1}} a_1^2(t) dt + \int_{t_{0|2}}^{t_{\times|2}} a_2^2(t) dt, \quad (4.6)$$

where a_1 and a_2 are the accelerations of the two agents, $t_{0|1}$ and $t_{0|2}$ are the initial departure times on their paths, $\mathbf{x}_{0|1}$ and $\mathbf{x}_{0|2}$ are the initial states, and $t_{\times|1}$ and $t_{\times|2}$ are the times when the agents clear the conflicting location k .

4.4.3 Three Phases of Conflict Resolution

Lastly, we contextualize the consensus model within real-time, decentralized conflict resolution, detailing the process by which conflicts are recognized, priorities negotiated, and resolutions concurred. In this regard, we present the three phases of conflict resolution: 1) the handshake phase, 2) the negotiation phase, and 3) the delivery phase. The details of these three phases are described below.

The *handshake phase* constitutes the initial stage, wherein multiple agents converge upon a common area and engage in the process of conflict resolution. This handshake is initiated based on a predefined observation radius and the potential for conflict. Owing to the localized nature of the interaction, agents interact exclusively with others situated within this observation radius. Moreover, among those within the radius, only those agents that present a potential conflict with the ego agent are considered relevant for engagement.

The *negotiation phase* represents the second stage, during which conflicting agents interactively converge to a consensus regarding the priority order for road passage. This phase commences immediately following the completion of the handshake phase. In an asynchronous and iterative manner, the coupled agents employ the consensus control model, utilizing local observations to execute a traversal policy that aligns with the presumed priority order. Each agent continues this process until the disparity in costs surpasses a predefined confidence threshold or until a change in plan becomes infeasible. It is important to note that during the consensus phase, an agent may potentially manipulate the consensus model to its own advantage by setting a particularly high disparity slack ι . This behavior effectively simulates the aggressive driving behaviors frequently observed in real-world road scenarios.

The *delivery phase* marks the final stage, where the negotiation period concludes and all agents are required to proceed in accordance with the agreed-upon priority order. During this phase, each agent is obligated to adhere to the established priority sequence regardless of any subsequent negotiation phases.

4.5 Berkeley DeepDrive Drone Dataset

To validate the least-cost consensus model, we collect an aerial video dataset called the Berkeley DeepDrive Drone dataset. The dataset consists of 20 post-processed videos, 16002

annotated images, and a development kit. It can be accessed from <https://github.com/b3d-project/b3d>. The sizes of the videos, the annotated images, and the development kit are 84.5 GB, 1.55 GB, and 225 MB, respectively, totaling to about 86.3 GB. The file structure of the dataset is illustrated in Figure 4.1. The 20 aerial videos were recorded with a DJI Mavic 2 Pro quadcopter between December 11 and December 21 of 2019 in China. Specific dates, lengths, and locations of the shootings are tabulated in Table 4.1. We briefly describe the three dataset components below.

Table 4.1: Dates, Lengths, and Locations of Videos

Name	Date	Length	Location
jnc00	11Dec19 10:34:49	21:31.13	(27.682343, 111.434406)
jnc01	11Dec19 11:12:24	21:31.33	(27.700318, 111.436217)
jnc02	11Dec19 11:54:14	18:47.90	(27.691014, 111.430694)
jnc03	13Dec19 16:08:34	20:02.67	(27.758277, 111.997546)
jnc04	13Dec19 16:34:26	21:01.80	(27.758274, 111.997575)
jnc05	14Dec19 14:34:13	20:29.87	(27.748028, 111.986952)
jnc06	15Dec19 14:59:56	20:47.62	(26.905471, 112.553030)
jnc07	15Dec19 16:21:29	20:55.42	(26.896411, 112.573248)
hwy00	20Dec19 10:39:47	16:31.26	(23.195657, 113.365076)
hwy01	20Dec19 11:04:03	15:01.90	(23.195657, 113.365076)
hwy02	20Dec19 11:27:21	15:01.80	(23.195657, 113.365076)
hwy03	20Dec19 15:31:57	15:03.84	(23.180680, 113.367367)
hwy04	20Dec19 15:54:14	14:30.87	(23.180680, 113.367367)
hwy05	20Dec19 16:20:23	15:04.77	(23.180680, 113.367367)
hwy06	21Dec19 12:38:35	17:01.46	(23.107395, 113.345787)
hwy07	21Dec19 13:01:55	17:02.66	(23.107395, 113.345787)
hwy08	21Dec19 13:26:57	18:01.18	(23.107395, 113.345787)
hwy09	21Dec19 15:08:24	16:00.83	(23.103120, 113.345733)
hwy10	21Dec19 15:32:38	15:30.80	(23.103120, 113.345733)
hwy11	21Dec19 15:58:53	10:45.21	(23.103120, 113.345733)

4.5.1 Processed Videos

Among the 20 processed videos, eight were recorded on top of junctions and 12 on top of highways. An overview of the topology of the roads covered in the videos is illustrated in Figure 4.3. The videos are stored in mp4 format, which can be opened by all common video players.

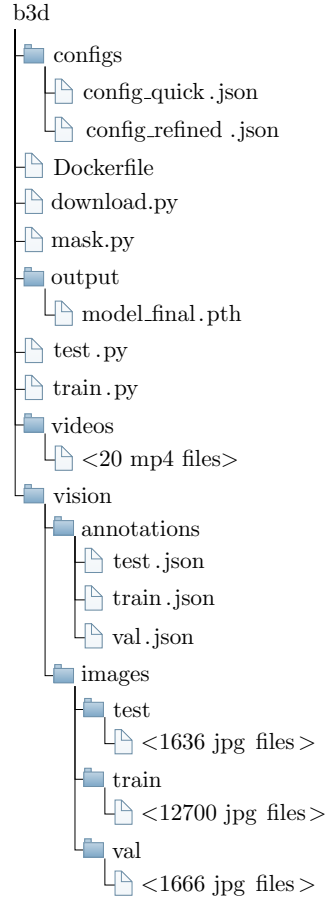


Figure 4.1: File structure of the B3D dataset. Informational files METADATA, LICENSE, and README are omitted for concision.

Scenarios recorded in the video can be classified into the following six categories: 1) unsignalized intersections, 2) unsignalized roundabouts, 3) tailgating accidents, 4) stop-and-go waves, 5) roadwork-induced merging, and 6) ramp-induced merging.

Unsignalized intersections can be found in videos jnc00.mp4, jnc01.mp4, jnc02.mp4, and jnc07.mp4. Videos jnc00.mp4 and jnc01.mp4 are two variants of three-way intersections, as shown in Figure 4.3a and Figure 4.3b, respectively. Videos jnc02.mp4 and jnc07.mp4 are two variants of four-way intersections, as shown in Figure 4.3c and Figure 4.3d, respectively.

Unsignalized roundabouts are captured in jnc03.mp4, jnc04.mp4, jnc05.mp4, and jnc06.mp4. Videos jnc03.mp4 and jnc04.mp4 are two recordings of a five-way roundabout, as shown in Figure 4.3e. Videos jnc05.mp4 and jnc06.mp4 are two variants of four-way roundabouts, as shown in Figure 4.3f and Figure 4.3g, respectively. Compared to video jnc03.mp4, video jnc04.mp4 has slightly more traffic.

The *tailgating accidents* consist of two collision events, first in hwy00.mp4 and then in

hwy01.mp4. At 00:45 of hwy00.mp4, we observe the first accident near the left margin of the frame, as shown in Figure 4.2a. At 13:10 of hwy01.mp4, we find another traffic accident in the middle of the frame, as shown in Figure 4.2b. Video hwy02.mp4 captures the resulting congested traffic induced by the second incident. The timestamps of the collision events are visualized in Figure 4.4.

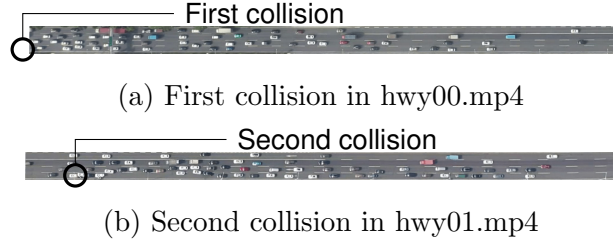


Figure 4.2: Tailgating collisions in hwy00.mp4 and hwy01.mp4. The collided vehicles are circled in black. The first accident involves at least two vehicles, while the second incident involves four vehicles.

Stop-and-go waves are recorded in hwy04.mp4 and hwy05.mp4. The first stop-and-go wave forms between 02:30 and 05:07 of hwy04.mp4. The second stop-and-go wave emerges between 06:06 and 08:10 of hwy04.mp4. The third stop-and-go wave is observed between 10:26 and 12:25 of hwy04.mp4. The fourth wave happens between 00:00 and 01:33 of hwy05.mp4. The last visible wave occurs between 05:19 and 06:07 of hwy05.mp4. For comparison, we provide video hwy03.mp4 as a free-flow baseline. The formation and dissipation events of the stop-and-go waves are visualized in Figure 4.5.

Roadwork-induced merging is recorded in hwy06.mp4, hwy07.mp4, and hwy08.mp4. The topology of the scenario is a four-lane-to-two-lane bottleneck, as shown in Figure 4.3j. Persistent congestion is observed before the merge point, while free-flow traffic is formed after the merge point.

Ramp-induced merging is recorded in hwy09.mp4, hwy10.mp4, and hwy11.mp4. The topology of the ramp is shown in Figure 4.3k, where a three-lane on-ramp is merging into a four-lane congested highway. The traffic stays congested *before and after* the merge point.

4.5.2 Annotated Images

To qualitatively analyze the vehicle behaviors observed in the videos, it is necessary to obtain a precise trajectory of every relevant vehicle in the scene. To construct the trajectory of a vehicle, one common approach is to first detect its position in every frame and then to use a tracking algorithm to associate each detected position across frames.

To this end, we build an image dataset, which can be used to train a vehicle detection model. The dataset consists of 16002 annotated images, 80% of which is split for training,

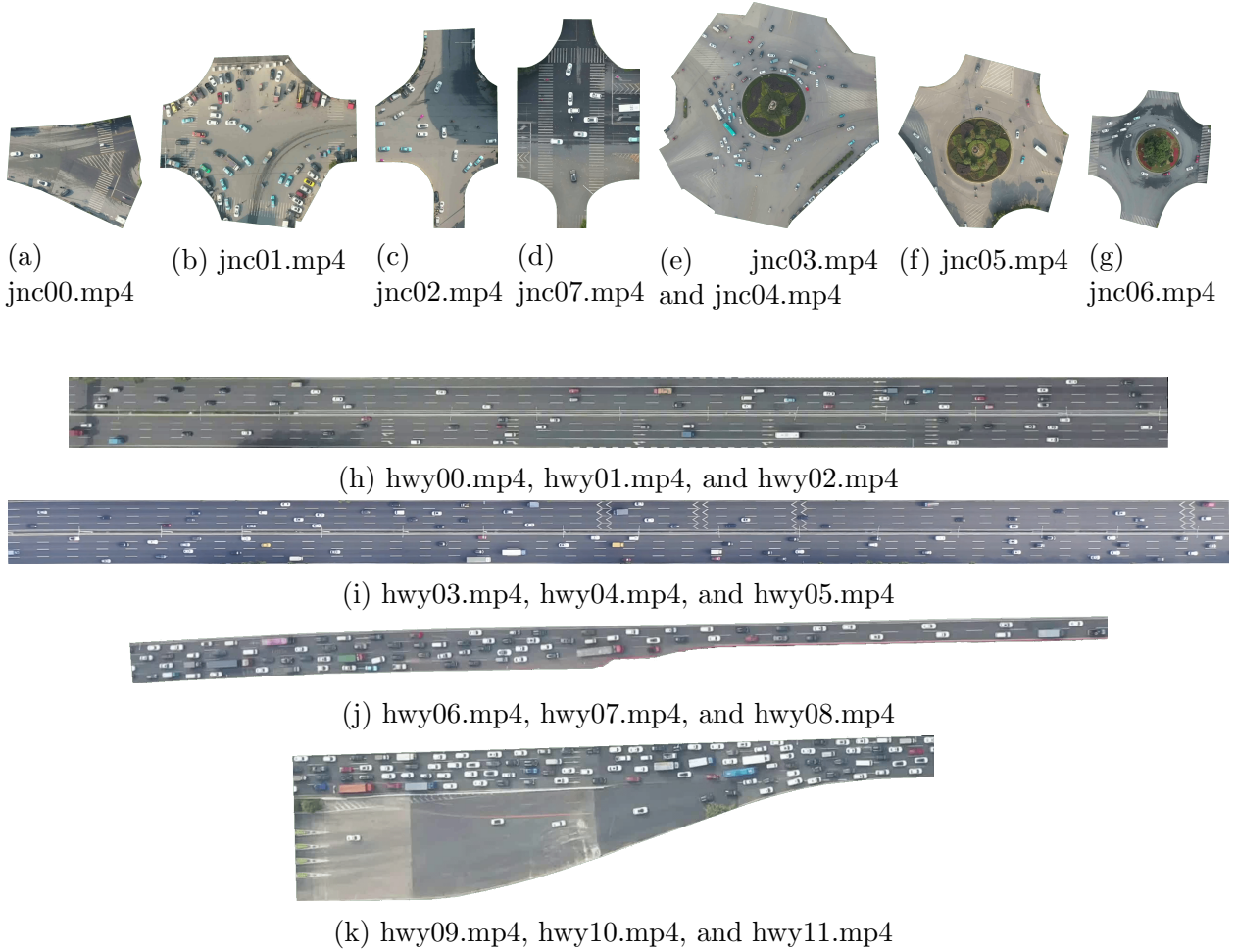


Figure 4.3: Typical Road Topology from the B3D dataset. In alphabetic order, we have (a)-(d) unsignalized intersections, (e)-(g) unsignalized roundabouts, (h)-(i) highways, and (i)-(k) merging bottlenecks. (a)-(d) are roughly on the same scale. (e)-(g) are on another identical scale. (h)-(i) are approximately on a third identical scale. Lastly, (j)-(k) are on a fourth distinct scale.

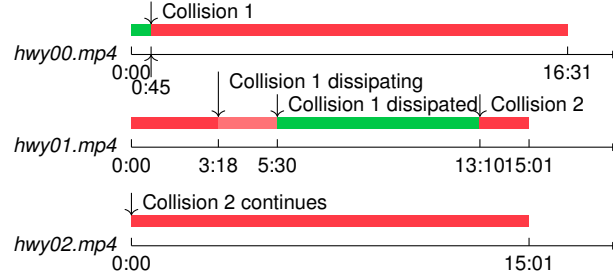


Figure 4.4: Timeline of the *tailgating accidents*. Green indicates regular traffic. Red indicates congestion caused by a collision. Light red indicates the induced congestion starts dissipating.

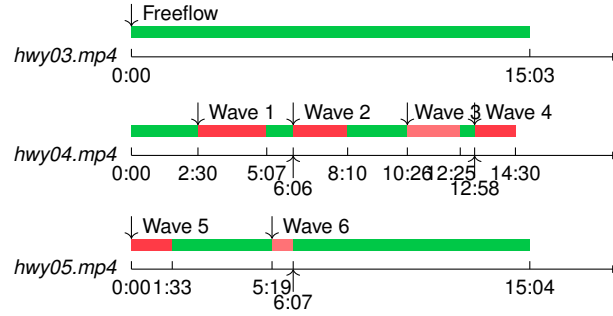


Figure 4.5: Timeline of the *stop-and-go waves*. Green indicates regular traffic. Red indicates congestion caused by a strong stop-and-go wave. Light red indicates congestion caused by a weak stop-and-go wave.

10% of which for validation, and 10% of which for testing. An example annotation is shown in Figure 4.6.

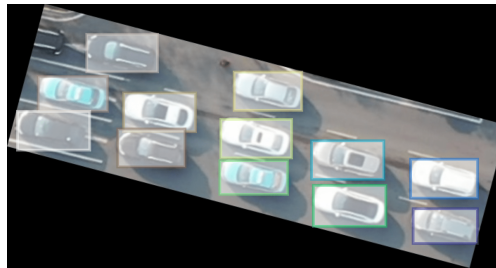


Figure 4.6: An example annotated image. Colored rectangles are axially aligned bounding boxes of the vehicles in the scene.

A total of 135303 bounding boxes are annotated for junction images and a total of 129939 bounding boxes created for highway images. Note that the annotations contain only

one object class, that is, the *vehicle* class. We do not distinguish large vehicles, such as buses and trucks, from small vehicles, such as sedans and SUVs.

To inspect or edit the provided annotations, we recommend using the open source annotation tool Computer Vision Annotation Tool (CVAT) [107].

4.5.3 Development Kit

In addition to the videos and the annotated images, we provide a development kit consisting of three example scripts: `train.py`, `test.py`, and `mask.py`.

Among them, the script `train.py` is used to show how the annotated image data may be used to train a neural network model for vehicle detection. In this script, we use the object detection library Detectron2 [135] to train a RetinaNet model [68] for detecting locations of vehicles in an input image.

The script `test.py` applies the trained model, trained via the `train.py` script, to an input image. This script is intended to serve as an example for post-training evaluation and inference. For convenience, we provide an optional pre-trained model, which can be used directly for inference. This way, users can directly use a working detection model without having to go through the computationally expensive training step locally. An instance of the detection results using the pre-trained model is shown in Figure 4.7.

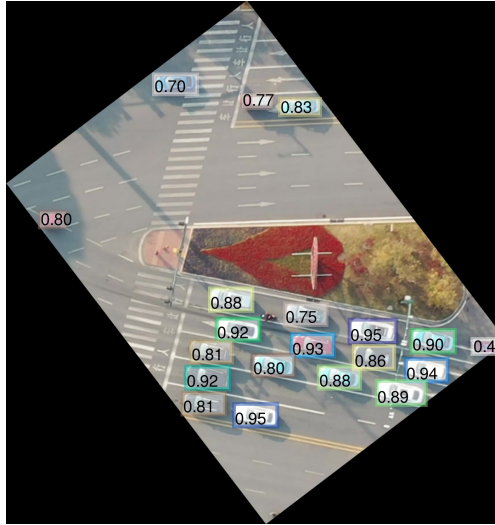


Figure 4.7: An example image with detection results. Colored rectangles are axially aligned bounding boxes. Black numbers are confidence scores of the estimated bounding boxes, with 1.0 being full certainty and 0.0 being full uncertainty.

Finally, we provide a masking script `mask.py` that crops an image according to a pre-defined polygonal mask. For example, Figure 4.8 shows how the script crops off irrelevant part of the roundabout image according to the red mask. The script intends to help users

to focus on the only relevant part of the scene, where relevance is defined by the user via CVAT. To crop a video, one only needs to define a polygonal mask for *one* frame of a video and then apply masking to *every* frame of the video.

For reproducibility, we encapsulate the development environment in a Docker image. The image can be directly built from the provided Dockerfile.



Figure 4.8: A demonstration of the masking process. The red polygonal mask is defined via CVAT. Pixels outside of the mask are considered irrelevant and are marked as black.

4.5.4 Areas of Applications

We believe that the dataset is useful in wide spectrum of applications in robotics. Specifically, we present two impactful use cases: one in motion planning and another in computer vision.

The primary area of applications is to model behaviors of human driving in understructured environments for decentralized motion planning. A typical modeling procedure may be as follows: 1) run the masking script to select relevant part of the video dataset; 2) apply the detection model to extract position of each vehicle in the masked scene; 3) inspect the results of the detection and, if necessary, manually correct any detection errors; 4) use a tracking algorithm such as SORT [12] to associate detected positions into trajectories; and 5) learn and validate a human driving model based on the estimated trajectories. An accurate human behavior model may be deployed on autonomous vehicles for navigation in understructured road environments. Additionally, the resulting model may serve as a baseline to develop decentralized coordination algorithms for other robotics applications such as warehouse robots.

The secondary use case is to develop or to evaluate computer vision algorithms for small road occupants in remote sensing settings. An accurate pedestrian and cyclist detection method enables us to study rich interactions among different modes of transportation without needing a labor-intensive process of human annotation. To our best knowledge, detecting pedestrians or cyclists from aerial videos remains a challenge for the state-of-the-art computer

vision. This is largely due to low resolution of those small objects, because low resolution renders the objects visually ambiguous, as illustrated in Figure 4.9. It is plausible that a detection model may find it challenging to distinguish a pedestrian from a watermark, as shown in Figure 4.9a and Figure 4.9b, or to discern two pedestrians from a cyclist, as shown in Figure 4.9c and Figure 4.9d. To this end, our dataset may serve as a data source to build training data and to validate candidate algorithms.

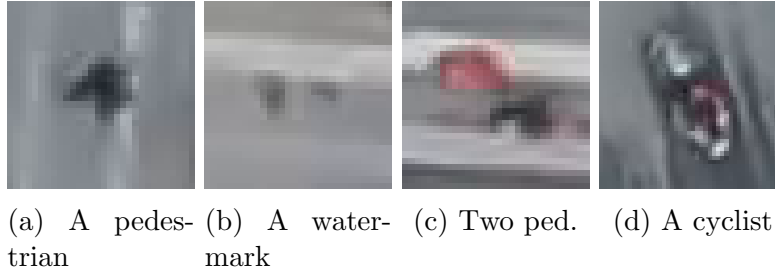


Figure 4.9: Visual ambiguity of low resolution objects. (a)(b) are visually alike. (c)(d) are also visually similar.

4.6 Numerical Simulation

In this section, we validate our work by conducting tests using 1) an sequence of two-agent scenarios with ground truth provided by the B3D dataset and 2) a constructed N -agent simulation scenario. Preliminary validation through the dataset reveals that the consensus model accurately predicts the priority order observed in cross-intersection settings. Further simulation demonstrates that the consensus control method effectively achieves decentralized conflict resolution, exhibiting qualitative batch queuing behavior similar to that observed in real-world scenarios.

4.6.1 Simulation Setup

The two-agent scenarios are extracted from a specific unsignalized intersection as recorded in video “jnc07.mp4” from the B3D dataset. The primary objective is to determine whether our least-action model accurately models the order of proceeding observed in the video. The road topology is illustrated in Figure 4.10. To test the model, we manually selected six relatively isolated two-agent interactions from this video to apply the least-action consensus model. Each of these interactions involves one agent traveling southbound and the other traveling westbound, with a single conflicting point at the intersection of the two paths. Three path pairs of the five scenarios with the corresponding initial velocity are shown in Figure 4.11.

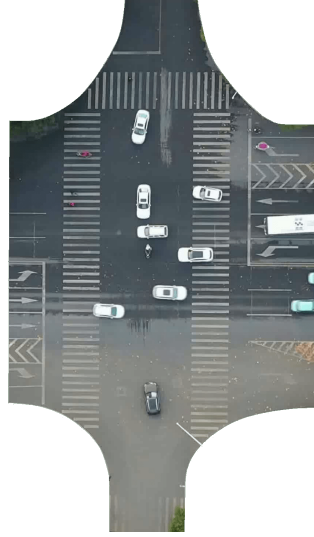


Figure 4.10: Road topology of the extracted two-agent scenario from video “jnc07.mp4.” The intersection is unsignalized, where southbound traffic and westbound traffic are in conflict at the crossing.

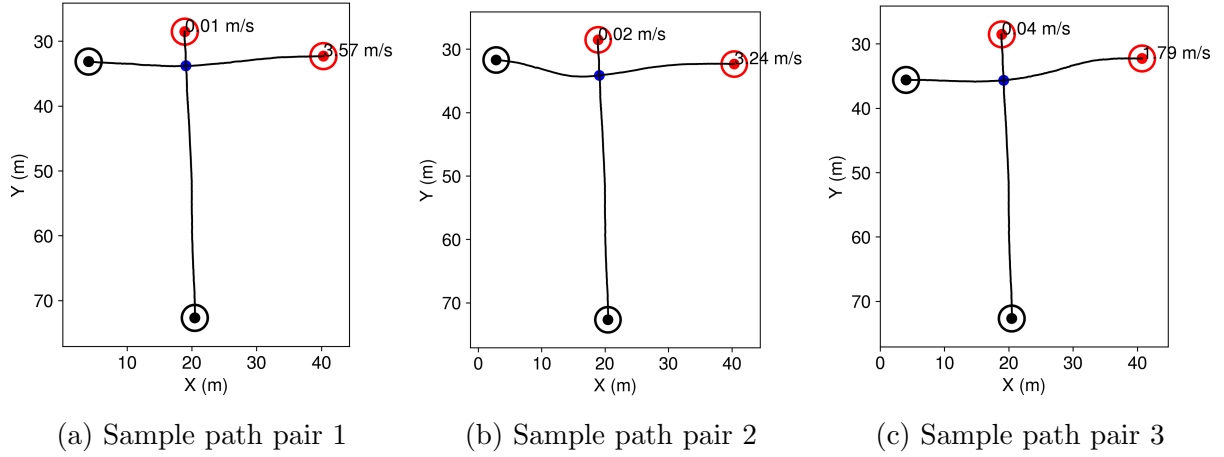


Figure 4.11: Example path pairs used for validating consensus model. Red circles indicate the origins of traffic while black circles denote the respective destinations. The blue node at the center marks the only conflicting point at the cross intersection. Green rectangles, labeled with numbers, represent the vehicles as they pass through the intersection. The initial speeds are displayed next to the respective origins.

To evaluate the integrative performance of the model, we constructed a full N -agent cross-intersection simulation involving southbound and westbound traffic flows. The goal of this evaluation is to check if our least-action model can be deployed in an integrative simulation without collision. The road topology of this simulation is depicted in Figure 4.12. Each arm of the cross-intersection is symmetrically configured to be 20 meters in length. The simulation is designed to run for a total duration of approximately 300 seconds, during which inbound traffic is randomly generated based on a Poisson distribution with an average arrival rate of 900 vehicles per hour. The initial position of each vehicle is set at the respective intersection entrances, and the initial velocity is sampled from a normal distribution with a mean of 3 m/s and a standard deviation of 1 m/s, truncated between 0 and 4 m/s. Should the initial condition of any inbound vehicle lead to an inevitable collision, that vehicle will be removed from the queue of incoming traffic.

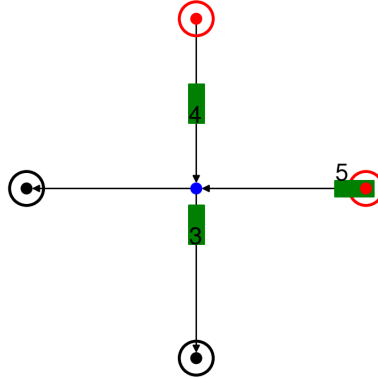


Figure 4.12: Road topology of the constructed N -agent simulation scenario. Red circles indicate the origins of traffic while black circles denote the respective destinations. The blue node at the center marks the only conflicting point at the cross intersection. Green rectangles, labeled with numbers, represent the vehicles as they pass through the intersection.

For both simulation scenarios, we use the least-action consensus model with quadratic cost on acceleration and IDM-based traversal policy. In particular, the IDM parameters we use here are $v_0 = 3.5$ m/s, $T = 0.8$ s, $a = 0.5$ m/s², $b = 1.0$ m/s², $\delta = 4$, and $s_0 = 2$ m.

4.6.2 Numerical Results

The simulation results confirm the effectiveness of the consensus model, demonstrating its capability 1) to accurately model the negotiation of the priority order performed by human drivers, and 2) to operate in an online setting for decentralized collision avoidance.

Results from the two-agent simulation tests are presented in Table 4.2. Each initial condition is characterized by entry speeds and distances to the conflict point. We represent each pair of entry speeds and distances as a tuple, with the data for southbound traffic

listed first, followed by westbound traffic. As indicated in the table, the model successfully simulates naturalistic conflict resolution under all initial conditions. It is important to note that in the scenario documented in the last row of the table, southbound traffic is prioritized over westbound traffic. This illustrates how perceived entry speeds influence the priority in the process of human conflict resolution.

Table 4.2: Configuration and Results of Two-Agent Simulation Tests

Entry Speeds (m/s)*	Distances to Conflict (m)*	True Priority Order	Modelled Priority Order
(0.01, 3.57)	(5.52, 21.25)	Westbound First	Westbound First
(0.01, 5.32)	(5.37, 20.04)	Westbound First	Westbound First
(0.02, 3.24)	(5.87, 21.14)	Westbound First	Westbound First
(0.16, 2.65)	(5.82, 21.71)	Westbound First	Westbound First
(0.04, 1.79)	(7.33, 21.97)	Westbound First	Westbound First
(0.41, 3.18)	(4.35, 21.19)	Southbound First	Southbound First

* Data on entry speeds and distance to conflict are arranged as (southbound, westbound)

The results of the N -agent simulation tests are depicted in Figure 4.13. The figure features two timelines: the top timeline visualizes the entry times of incoming vehicles, while the bottom timeline displays their crossing times. Red data points represent vehicles approaching from the southbound direction, and blue data points from the westbound. Although vehicles enter the intersection randomly, as per the simulation setup, they cross the conflict point in a distinct batch pattern. Each direction alternates in crossing the conflict point in batches of queues. This qualitative behavior aligns with observations from the B3D dataset.

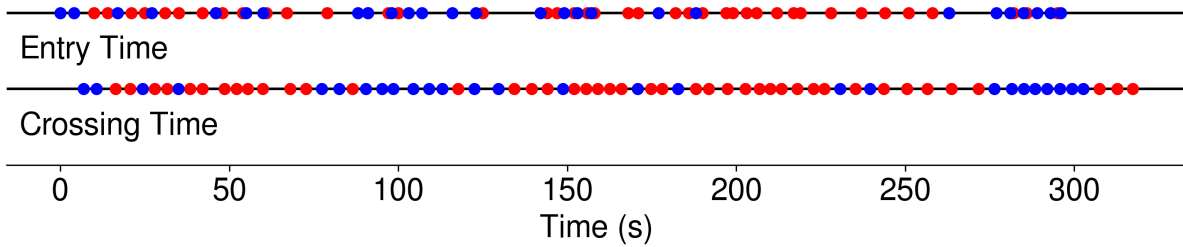


Figure 4.13: Visualization of the entry and crossing times of vehicles approaching from the southbound and westbound directions, respectively. Each red point represents a southbound vehicle, while each blue point represents a westbound vehicle. It is important to note that although vehicles enter the intersection randomly, they cross the intersection in organized batches, forming queues of alternating directions.

4.7 Discussion

The preliminary results from the numerical simulations suggest that the consensus-based approach is a viable framework for decentralized conflict resolution in unstructured road environments. Despite these promising findings, several critical aspects of this research remain unexplored. We identify and discuss four particularly salient open questions: 1) theoretical convergence analysis, 2) further empirical validation, 3) handshake modeling in complex environments, and 4) optimization of the model. Detailed discussions of these issues are presented below.

First and foremost, the primary theoretical challenge involves demonstrating the convergence of consensus in the presence of disturbances. Although we have empirically shown that this consensus approach to decentralized conflict resolution accurately describes how human drivers negotiate right-of-way in real-time, it has yet to be proven under which conditions the method will converge to a consistent priority order in a distributed setting. A critical aspect of this proof is to demonstrate that the cost rankings for each priority arrangement consistently converge when computed independently by different agents amid disturbances. These disturbances may include noisy observations, imperfect actuation, and asymmetrical coupling topology among agents.

Beyond theoretical analysis, it is crucial to evaluate the model's performance through more complex empirical testing. It is beneficial to compare the behavior of the consensus control model against the complex interactions observed in the dataset. For instance, rather than limiting tests to two-agent conflict resolution, the model could be evaluated against N -agent interactions involving diverse conflict types and demand profiles. Moreover, testing the model in real vehicles is equally important. Hardware tests assess whether sensor noise, actuation errors, and system delays impact the method's efficacy. Additionally, conducting tests of the consensus model alongside a human driver, in a Turing test-like manner, is intriguing. A successful model should operate in an unstructured environment in such a way that it is indistinguishable from neighboring human drivers.

Another open modeling question concerns the handshake phase, which determines how vehicles couple as they approach imminent collisions. The coupling behaviors of vehicles during this phase are influenced by several factors, including vehicle paths, proximity, speeds, and individual driver characteristics. Developing a model that succinctly captures the most critical features, while disregarding less relevant factors, remains a significant challenge. This issue becomes particularly critical in scenarios where multiple agents are densely packed within a small area. Human drivers appear to engage with only the minimal number of neighboring drivers necessary to avoid collisions, but the precise process of this minimum coupling are yet to be understood.

Last but not least, the consensus model can be optimized to surpass the capabilities of human drivers. In environments where all agents are automated, the handshake phase can be significantly enhanced. Automated agents have the capability to communicate faster, further, and more accurately than human drivers through advanced wireless communication technologies. This enhanced capability can be leveraged to significantly improve the safety

and performance of the entire system. Additionally, the consensus rules can be modified to include more diverse cost factors. Instead of solely basing costs on kinematic measurements, the cost metrics can be diversified by assigning different values to various vehicle classes, such as ambulances, school buses, and fire trucks, and mission types, including police vehicles on active duty or taxis on time-sensitive trips.

Chapter 5

Centrally Controlled Multi-Agent Systems

While partially meshed MASs are ubiquitous, arising naturally from uncoordinated multi-agent interactions, their theoretical performance upper bound is limited due to the lack of central coordination. For this reason, when performance is a primary concern, centrally controlled MASs are often preferred. In a centrally controlled MAS, all client agents can access the status of every other agent through a server agent. This server agent may also be authorized to exert central control over all client agents to optimize system-level performance. For instance, the quintessential topology in computer networking, including the Internet, is a centrally controlled communication network, also known as star-connected network. This approach is also prevalent in transportation systems, such as airport flight coordination and warehouse robot management. In this chapter, we explore a classical planning and control problem within centrally controlled MASs: What is the performance upper bound enabled by central coordination in the context of transportation? The chapter introduces a novel mixed-integer linear program (MILP) capable of addressing transportation problems across arbitrary network topologies. A reservation-based heuristic is proposed to scale the mixed-integer programming method to large network sizes effectively. Lastly, the chapter evaluates the empirical performance of the method across a diverse set of network topologies, including archetypal atomic networks and composite networks.

5.1 Introduction

With the advancement of automation and artificial intelligence, we are moving rapidly toward a future where conventional societal systems are being more and more optimized. Of these societal systems, a notable example is the transportation systems. In this field, we have witnessed the significant progress in autonomous driving.

Autonomous driving technology is poised to bring benefits that will fundamentally change the performance of our transportation networks. For instance, automation is able to eliminate

human errors, thus eliminating accidents caused by human drivers' oversight. It is able to prevent formation of stop-and-go waves caused by greedy driving behaviors and delay due to human reaction time. Moreover, t is also able to drive optimally to reduce driving discomfortable, increase energy efficiency, and mitigate consequently environmental impacts.

Beyond the aforementioned benefits, which are already partially achieved by assisted driving technology, perhaps the most fundamental advantage is significant enhancement in the density, speed, and therefore throughput of the entire transportation system. Consider the future where every vehicle on road is automated and centrally coordinated by a remote server. In this scenario, we have the ability to centrally optimize and control the entire transportation system, leading to performance improvement that is potentially beyond imagination. However, to date, a quantitative analysis on how much this idealized performance upper bound is has yet been explicitly explored.

To study the maximum potential of autonomous driving on our society, we propose a mixed-integer linear program that is able to find an optimal solution to the general transportation problem with any demand profile on any road networks. Moreover, just like any mixed-integer optimization, its scalability is poor when the problem size is large. To mitigate this scalability issue, we further present a numerical heuristic based on the idea of reservation. This reservation-based heuristic is able to turn the mixed-integer optimization model into a polynomial time method, at a cost of loss of global optimality.

To illustrate the effectiveness of our mixed-integer algorithm and efficiency of our reservation-based heuristic, we test our method on a variety of simulation scenarios ranging from atomic road networks such as cross intersection and composite road networks like intersection grid.

In summary, we state the main contributions of the work as follows.

- We propose a novel mixed-integer scheduling algorithm that works with arbitrary road network topologies.
- We introduce a reservation-based heuristic that enables the mixed-integer scheduling algorithm to scale to larger problem sizes.
- We evaluate the performance and efficiency of our method through a diverse set of numerical simulations.

5.2 Related Works

The problem of scheduling and navigation in multi-agent systems, particularly in the context of autonomous vehicles and robotic agents, has garnered significant attention in recent literature. This section provides an overview of several key studies that have substantially contributed to the field, emphasizing their methodologies, achievements, and limitations.

Research in this area began with the application of search algorithms such as A*, which laid the groundwork for more sophisticated approaches [40]. Silver, for instance, augmented the A* algorithm with a reservation grid to tackle the multi-agent version of pathfinding [110].

Although innovative, this method encounters scalability issues in large problems and suffers from limited temporal resolution due to discretization.

As interest in multi-agent systems expanded into engineering fields such as robotics, aviation, and transportation, the focus shifted towards collision avoidance. In robotics, multi-agent collision-free path planning has become a classical problem, with notable contributions such as those by Huo and Ioan [45, 47]. Aviation has similarly benefited from research in collision avoidance, with foundational works including those by Mao, Richards, and Bayen [73, 100, 8]. In the transportation sector, early influential studies by Austin [28] have spurred further developments, such as those seen in more recent works by Chandra and Antonio [19, 7].

Beyond traditional approaches utilizing control and optimization, innovative strategies employing machine learning have also emerged. For instance, Liu introduced a general multi-agent path planning algorithm using evolutionary reinforcement learning [69], and Huang applied reinforcement learning for multi-UAV collision avoidance [43]. These machine learning-based methodologies offer new perspectives and capabilities in handling the complex dynamics of multi-agent systems.

This array of prior works illustrates the diverse and evolving landscape of research in multi-agent system navigation and scheduling, highlighting both the progression and the challenges in optimizing these intricate systems.

5.3 Problem Statement

Consider a scenario involving N agents, each starting from a predefined origin in a parking lot, navigating a road network G , and aiming to reach a destination in a different parking location. The objective is to devise a waypoint schedule for each agent that directs them from their origin to their destinations as early as possible without slowdowns or collisions.

Let us denote the road network by $G = (V, E)$, where V represents the set of vertices and E the set of edges. A path p is defined as a finite sequence of homogeneously directed edges connecting a series of unique vertices. Provided that $v_0, v_1, \dots, v_f$ is the sequence of vertices along the path, with v_0 as the origin and v_f the destination, we define the path as $p = \{v_0, v_1, \dots, v_f\}$. The n -th agent is denoted as n , and their schedule T is defined as a set of arrival and departure times at each waypoint along the path p . Specifically, $\mathbf{t}_n = \{t_{a|n}^{v_i} \mid i = 1, 2, \dots, f\} \cap \{t_{d|n}^{v_i} \mid i = 0, 1, \dots, f-1\}$, where $t_{a|n}^{v_i}$ and $t_{d|n}^{v_i}$ represent the arrival and departure times of agent n at waypoint v_i , respectively.

With the notations defined above, the n -th agent travels from the origin to the destination as quickly as possible if and only if $t_{a|n}^{v_f}$ is minimized. Alternatively, if the goal is solely to minimize travel time, it suffices to only minimize $t_{a|n}^{v_f} - t_{d|n}^{v_0}$. Additionally, each agent may be assigned an earliest departure from its origin.

Assuming all agents are required to drive at an operational speed $\nu^* \in \mathcal{R}_{>0}$, the schedule for the n -th agent, following path p with agent length ℓ , must satisfy the following condition

to avoid slowdowns:

$$\frac{|v_{i+1} - v_i|}{t_{a|n}^{v_{i+1}} - t_{a|n}^{v_i}} = \frac{|v_{j+1} - v_j|}{t_{d|n}^{v_{j+1}} - t_{d|n}^{v_j}} = \frac{\ell}{t_{d|n}^{v_k} - t_{a|n}^{v_k}} = \nu^*,$$

for $i = 1, 2, \dots, f-1$, $j = 0, 1, \dots, f-2$, and $k = 1, 2, \dots, f-1$.

Two agents are free from collision if they do not occupy the same spatial location at the same time. Let p_1 and p_2 be the respective paths of the two agents. Define p_k as the overlapping segment(s) of paths p_1 and p_2 , i.e., $p_k := p_1 \cap p_2$. p_k could comprise any combination of isolated waypoints and isolated sub-paths. An agent is free from collision if and only if for every overlapped path segment, the arrival time of one agent at the entry of that segment is later than the departure time of the other agent at that entry.

Generalizing the collision avoidance problem to scenarios involving more than two agents, the necessary and sufficient condition is that no two distinct pairs of agents collide.

In summary, putting all discussions above together, we formally state the problem below:

Problem 6. (Multi-Agent Conflict Resolution Problem) Consider N agents of potentially conflicting paths $p_1, p_2, \dots, p_N$. Determine a schedule for each path such that 1) travel time is minimized, 2) travel speed is uniformly maintained at ν^* , and 3) travel trajectory is free from collisions.

5.4 Mixed-Integer Scheduling Method

In this section, we address Problem 6 using a Mixed Integer Linear Programming (MILP) approach. The overall structure of our MILP formulation is as follows:

$$\min_{\mathbf{t}_1, \dots, \mathbf{t}_n, \mathbf{b}} \sum_{i=1}^n J_i(\mathbf{t}_i) \quad (5.1a)$$

$$\text{s.t.} \quad h_0(\mathbf{t}_1, \mathbf{t}_2, \dots, \mathbf{t}_n) \geq \mathbf{0}, \quad (5.1b)$$

$$h(\mathbf{t}_1, \mathbf{t}_2, \dots, \mathbf{t}_n) = \mathbf{0}, \quad (5.1c)$$

$$f(\mathbf{t}_1, \mathbf{t}_2, \dots, \mathbf{t}_n, \mathbf{b}) \geq 0, \quad (5.1d)$$

where $J_i(\cdot)$ represents the cost functions associated with travel time, $h_0(\cdot)$ defines the initial conditions, $h(\cdot)$ denotes the equality constraints that encapsulate kinematic relationships, and $f(\cdot)$ describes the inequality constraints ensuring non-collision.

5.4.1 Optimization Model Design

The cost functions $J_i(\cdot)$ and equality constraints $h_i(\cdot)$ are linear, while the non-collision constraint $f(\cdot)$ is mixed-integer linear. Specifically, for the j -th agent following the path $p_j = \{v_0, v_1, \dots, v_f\}$, the decision variables are defined as $\mathbf{t}_j = \{t_{a|j}^{v_i} \mid i = 1, 2, \dots, f\} \cup \{t_{d|j}^{v_i} \mid i = 0, 1, \dots, f-1\}$. The cost function $J_j(\cdot)$ is then expressed by:

$$J_j(\mathbf{t}_j) = t_{a|j}^{v_f} - t_{d|j}^{v_0}.$$

The initial condition function $h_0(\cdot)$ specifies the earliest possible departure time for each agent from their respective origins, formulated as:

$$t_{d|i}^{v_0} > t_{0|i} \quad \forall i = 1, 2, \dots, n,$$

where $t_{0|i}$ denotes the earliest departure time for agent i . Additionally, the kinematic constraints $h(\cdot)$ can be represented as:

$$\frac{|v_{i+1} - v_i|}{t_{a|j}^{v_{i+1}} - t_{a|j}^{v_i}} = \frac{|v_{j+1} - v_j|}{t_{d|j}^{v_{j+1}} - t_{d|j}^{v_j}} = \frac{\ell}{t_{d|j}^{v_k} - t_{a|j}^{v_k}} = \nu^*,$$

for $i = 1, 2, \dots, f-1$, $j = 0, 1, \dots, f-2$, and $k = 1, 2, \dots, f-1$. To illustrate the formulation of non-collision constraints, consider, without loss of generality, that the j -th agent conflicts with the k -th agent at waypoint $v_\times$. The non-collision constraint is then formulated as:

$$t_{a|j}^{v_\times} > t_{d|k}^{v_\times} \text{ or } t_{a|k}^{v_\times} > t_{d|j}^{v_\times}.$$

This “either or” condition is implemented numerically using the Big M method, involving a binary decision variable and a sufficiently large constant, as shown below:

$$\begin{aligned} (t_{a|k}^{v_\times} - t_{d|j}^{v_\times}) + M \cdot b &\geq 0, \\ (t_{a|j}^{v_\times} - t_{d|k}^{v_\times}) + M \cdot (1 - b) &\geq 0, \end{aligned}$$

where b is the binary decision variable, and M is a large constant. If multiple conflict points exist within a problem, each point should introduce a similar “either or” constraint.

5.4.2 A Two-Agent Example

To concretely demonstrate the application of the MILP formulation, we consider a simple two-agent cross-intersection scenario, depicted in Figure 5.1. In this scenario, two agents navigate intersecting paths: the first agent, covering a distance of ℓ_1 , follows path $p_1 = \{A, B, C, D\}$, while the second agent, traveling a distance of ℓ_2 , takes path $p_2 = \{E, F, B, G\}$. These paths intersect at point B , thereby creating a conflict point. The continuous decision variables, which represent the respective arrival and departure times of the two agents along their paths, are defined as:

$$\begin{aligned} \mathbf{t}_1 &= \{t_{d|1}^A, t_{a|1}^B, t_{d|1}^B, t_{a|1}^C, t_{d|1}^C, t_{a|1}^D\}, \\ \mathbf{t}_2 &= \{t_{d|2}^E, t_{a|2}^F, t_{d|2}^F, t_{a|2}^B, t_{d|2}^B, t_{a|2}^G\}. \end{aligned}$$

The binary decision variable, denoted by b , determines which agent has the priority to clear the conflict point at B .

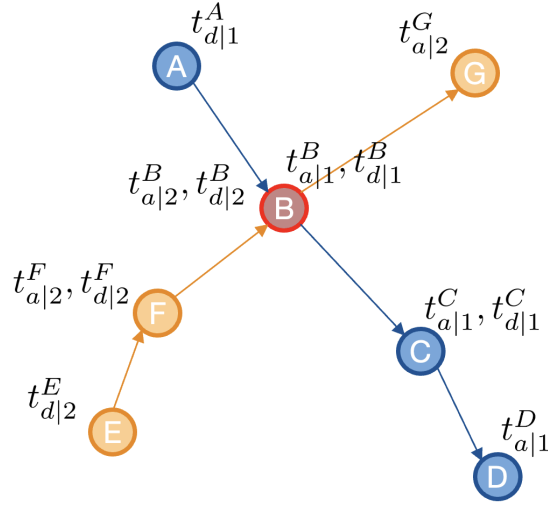


Figure 5.1: The two-agent setup at a crossing intersection. Each path is discretized into waypoints, at which arrival and departure times are computed through optimization.

Utilizing the above setup and the MILP formulation outlined in (5.1), we derive the following specific optimization formulation to resolve the problem:

$$\begin{aligned}
 \min_{\mathbf{t}_1, \mathbf{t}_2, b} \quad & t_{a|1}^D + t_{a|2}^G \\
 \text{s.t.} \quad & t_{d|1}^A \geq t_{0|1}, \\
 & t_{d|2}^E \geq t_{0|2}, \\
 & \frac{\ell_1}{t_{d|1}^B - t_{a|1}^B}, \frac{\ell_1}{t_{d|1}^C - t_{a|1}^C}, \frac{\ell_2}{t_{d|2}^F - t_{a|2}^F}, \frac{\ell_2}{t_{d|2}^B - t_{a|2}^B} = \nu^*, \\
 & \frac{|B-A|}{t_{d|1}^B - t_{d|1}^A}, \frac{|C-B|}{t_{d|1}^C - t_{d|1}^B}, \frac{|F-E|}{t_{d|2}^F - t_{d|2}^E}, \frac{|B-F|}{t_{d|2}^B - t_{d|2}^F} = \nu^*, \\
 & \frac{|C-B|}{t_{a|1}^C - t_{a|1}^B}, \frac{|D-C|}{t_{a|1}^D - t_{a|1}^C}, \frac{|B-F|}{t_{a|2}^B - t_{a|2}^F}, \frac{|G-B|}{t_{a|2}^G - t_{a|2}^B} = \nu^*, \\
 & (t_{a|2}^B - t_{d|1}^B) + M \cdot b \geq 0, \\
 & (t_{a|1}^B - t_{d|2}^B) + M \cdot (1-b) \geq 0,
 \end{aligned}$$

where $t_{0|1}$ and $t_{0|2}$ are the respective initial entry times of the agents, and M is chosen to be sufficiently large. When $b = 0$, the constraint $(t_{a|1}^B - t_{d|2}^B) + M \cdot (1-b) \geq 0$ is deactivated, thus prioritizing agent 1 over agent 2. Conversely, when $b = 1$, agent 2 is prioritized over agent 1.

Regarding the choice of M , we propose $M = \frac{|D-A|+|G-E|}{\nu^*}$ as a feasible selection. This value represents the time required for one agent to completely traverse its path before the

other agent begins its movement.

5.5 A Reservation-Based Heuristic

While the MILP model presented in (5.1) effectively encapsulates all aspects of the problem requirements, it inherits a critical challenge from the formulation of mixed-integer constraints. The computational complexity of the MILP formulation is primarily influenced by the mixed-integer component, rather than by the linear programming aspects. The number of binary decision variables correlates directly with the number of conflicting locations within the network. In a network containing n conflicting locations, there are 2^n possible conflict resolution scenarios. As a result, the problem may only be solvable in exponential time under the worst-case conditions. This exponential increase in the number of binary variables and constraints poses significant practical difficulties for the deployment of the MILP in real-world settings.

5.5.1 Formulation of the Heuristic

To address the practical challenges of the MILP model described in (5.1), due to the inclusion of mixed-integer constraints, we introduce a reservation-based heuristic. This heuristic limits the growth of binary decision variables and constraints, thereby rendering the algorithm suitable for scalable, real-time deployment. The fundamental idea of this heuristic is to decompose a large problem into several smaller, manageable problems, each solved sequentially. Consider a scenario with N conflicting locations. The total number of possible conflict resolution scenarios for this large problem is $O(2^N)$. By dividing the problem into smaller segments, each containing $O(M)$ conflicting locations, the number of conflict resolution scenarios for all these smaller problems collectively becomes $O(\frac{N}{M}2^M) = O(N)$. Thus, this heuristic substantially simplifies the mixed-integer component of the optimization from exponential to linear complexity, albeit at the expense of some sub-optimality.

To ensure that the solutions to the smaller problems remain viable for the original, larger problem, a reservation system is employed. The non-collision constraint (5.1d) is the primary barrier preventing the problem from being resolved through distributed approaches. Therefore, to effectively manage a large optimization by splitting it into several smaller ones, it is sufficient to selectively break the constraint (5.1d).

In this work, we adopt a reservation-based approach, utilizing a system with limited memory depth. Let $\mathbf{t}_1, \dots, \mathbf{t}_n, \mathbf{b}$ denote the decision variables of the original large problem, which we divide into m subproblems as follows:

$$\begin{aligned} \text{Subproblem 1: } & \mathbf{t}_{1|1}, \dots, \mathbf{t}_{n_1|1}, \mathbf{b}_1 \\ \text{Subproblem 2: } & \mathbf{t}_{1|2}, \dots, \mathbf{t}_{n_2|2}, \mathbf{b}_2 \\ & \vdots \\ \text{Subproblem } m: & \mathbf{t}_{1|m}, \dots, \mathbf{t}_{n_m|m}, \mathbf{b}_m, \end{aligned}$$

with the condition that $\sum_{i=1}^m n_i = n$. After solving the first subproblem, the resolved time schedule, $t_{1|1}^*, \dots, t_{n_1|1}^*$, is stored in the reservation system, denoted $\mathcal{R}$. Subsequent subproblems are then solved using the MILP (5.1), along with an additional constraint:

$$\mathbf{t}_{1|2}, \dots, \mathbf{t}_{n_2|2} \notin \mathcal{R},$$

where $\notin$ requires that the new schedules must not collide with or precede those already in $\mathcal{R}$. A time buffer may be added to enhance temporal separation. This process is repeated for each subproblem in sequence. If the number of stored solutions at any waypoint exceeds a memory threshold γ , one entry is pruned by merging two closely timed solutions.

The following figure illustrates this sequential optimization approach.

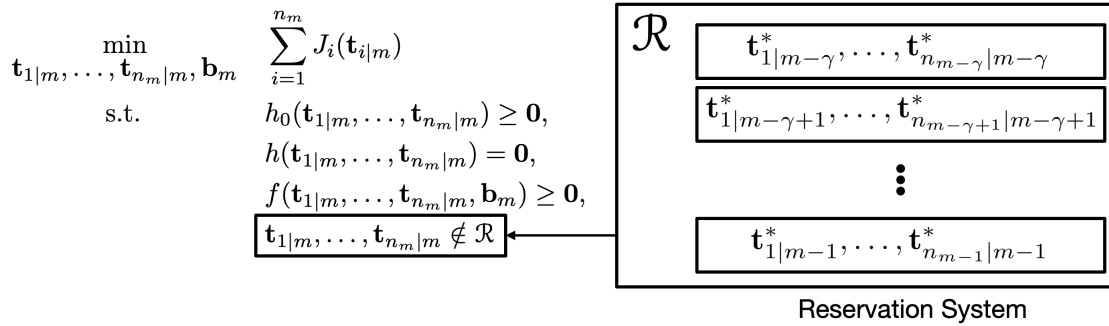


Figure 5.2: The reservation-based heuristic applied to the two-agent setup. The reservation table $\mathcal{R}$ caches existing solutions, which are then used as constraints in subsequent optimizations. This approach allows the mixed-integer method to be scalable in real-time applications.

5.5.2 A Four-Agent Example

To demonstrate the practical application of the heuristic, we illustrate its usage through a four-agent scenario. For simplicity, we employ the same road topology as depicted in Figure 5.1. Agent 1 and Agent 3 travel in sequence along the path $p_1 = \{A, B, C, D\}$, while Agent 2 and Agent 4 follow the path $p_2 = \{E, F, B, G\}$. The continuous decision variables for the four agents are defined as:

$$\begin{aligned} \mathbf{t}_1 &= \{t_{d1}^A, t_{a1}^B, t_{d1}^B, t_{a1}^C, t_{d1}^C, t_{a1}^D\}, \\ \mathbf{t}_2 &= \{t_{d2}^E, t_{a2}^F, t_{d2}^F, t_{a2}^B, t_{d2}^B, t_{a2}^G\}, \\ \mathbf{t}_3 &= \{t_{d3}^A, t_{a3}^B, t_{d3}^B, t_{a3}^C, t_{d3}^C, t_{a3}^D\}, \\ \mathbf{t}_4 &= \{t_{d4}^E, t_{a4}^F, t_{d4}^F, t_{a4}^B, t_{d4}^B, t_{a4}^G\}. \end{aligned}$$

To reduce the computational complexity of solving a simultaneous problem involving all four agents, the problem is divided into two two-agent subproblems. The first subproblem,

involving Agent 1 and Agent 2, is solved as detailed in Section 5.4.2, and the solution is cached in the reservation system $\mathcal{R}$. Subsequently, the second subproblem involving Agent 3 and Agent 4 is addressed, referencing the reserved solution for Agent 1 and Agent 2.

The MILP formulation for Agent 3 and Agent 4 largely mirrors that of Agent 1 and Agent 2, with the critical addition of the new constraint:

$$\mathbf{t}_3, \mathbf{t}_4 \notin \mathcal{R},$$

expanded below in terms of individual constraints.

Considering the previously optimized solutions for Agent 1 and Agent 2:

$$\begin{aligned}\mathbf{t}_1^* &= \{t_{d|1}^{A*}, t_{a|1}^{B*}, t_{d|1}^{B*}, t_{a|1}^{C*}, t_{d|1}^{C*}, t_{a|1}^{D*}\}, \\ \mathbf{t}_2^* &= \{t_{d|2}^{E*}, t_{a|2}^{F*}, t_{d|2}^{F*}, t_{a|2}^{B*}, t_{d|2}^{B*}, t_{a|2}^{G*}\},\end{aligned}$$

the following constraints enforce that the new solutions for Agent 3 and Agent 4 respect the reserved solution:

$$\begin{aligned}\text{Agent 3 at waypoint A: } & t_{d|3}^A > t_{d|1}^{A*}, \\ \text{Agent 3 at waypoint B: } & (t_{d|3}^B < t_{a|1}^{B*} \text{ or } t_{a|3}^B > t_{d|1}^{B*}) \text{ and } (t_{d|3}^B < t_{a|2}^{B*} \text{ or } t_{a|3}^B > t_{d|2}^{B*}), \\ \text{Agent 3 at waypoint C: } & t_{d|3}^C < t_{a|1}^{C*} \text{ or } t_{a|3}^C > t_{d|1}^{C*}, \\ \text{Agent 3 at waypoint D: } & t_{a|3}^D > t_{d|1}^{D*}, \\ \text{Agent 4 at waypoint E: } & t_{d|4}^E > t_{d|2}^{E*}, \\ \text{Agent 4 at waypoint F: } & t_{d|4}^F < t_{a|2}^{F*} \text{ or } t_{a|4}^F > t_{d|2}^{F*}, \\ \text{Agent 4 at waypoint B: } & (t_{d|4}^B < t_{a|1}^{B*} \text{ or } t_{a|4}^B > t_{d|1}^{B*}) \text{ and } (t_{d|4}^B < t_{a|2}^{B*} \text{ or } t_{a|4}^B > t_{d|2}^{B*}), \\ \text{Agent 4 at waypoint G: } & t_{d|4}^G > t_{a|2}^{G*}.\end{aligned}$$

Each set of “or” inequalities is replaced by a pair of numerically equivalent mixed-integer constraints through the Big M method, each introducing a binary integer decision variable. Because the memory depth of the reservation system is intentionally limited, the introduction of new binary variables has bounded impacts on running time.

5.6 Numerical Simulation

To evaluate the MILP formulation along with the reservation-based heuristic, we applied the algorithm to a variety of simulation scenarios. These scenarios encompass simple atomic networks, including cross intersections, roundabouts, and thruways with onramps and offramps. Additionally, composite networks composed of elements from these atomic networks were tested, such as a 2×2 intersection grid, three-way intersection mesh, concentric hexagonal rings, and a large 8×8 intersection grid. The simulation results confirm that the algorithm is capable of solving networks with arbitrary topology, can scale to very large problems, and is suitable for real-time deployment.

5.6.1 Simulation Setup

To test the performance of the MILP scheduling algorithm, we present seven distinct simulation scenarios: 1) a cross intersection, 2) a roundabout intersection, 3) a thruway with onramps and offramps, 4) a 2×2 intersection grid, 5) a three-way intersection mesh, 6) concentric hexagonal rings, and 7) an 8×8 intersection grid. The first three scenarios represent archetypal atomic road networks, while the remaining four are composite networks, representing common real-world road network patterns. We detail the specifications of these seven scenarios as follows.

The *cross intersection* scenario, illustrated in Figure 5.3, features a four-way, single-lane cross intersection connected to four parking lots at each end. Each parking lot accommodates 32 vehicles, with 16 randomly selected as vehicle origins. Vehicles are assigned random destinations at different, unoccupied parking lots. Paths for each origin-destination pair are computed using a standard shortest-path algorithm. The initial departure time for all vehicles is uniformly set to zero, which marks the beginning of the simulation. This scenario serves as a fundamental unit test for evaluating multi-agent conflict resolution strategies.

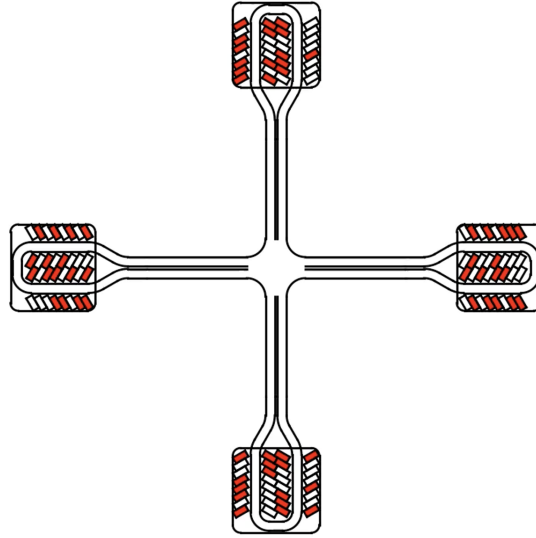


Figure 5.3: Visualization of the four-way cross intersection scenario. The red rectangles represent the origins of vehicles in the respective parking lots. Path conflicts primarily happen at the intersection.

The *roundabout intersection* scenario, illustrated in Figure 5.4, presents a four-way, single-lane roundabout. This setup bears resemblance to the cross intersection scenario, featuring four parking lots connected at each endpoint. The configuration of the parking lots, trip generation, path assignment, and initial departure times are consistent with those employed in the cross intersection scenario. A notable alteration in this scenario is the introduction of

a roundabout, which test the algorithm’s ability to manage continuous merging and forking at the intersection.

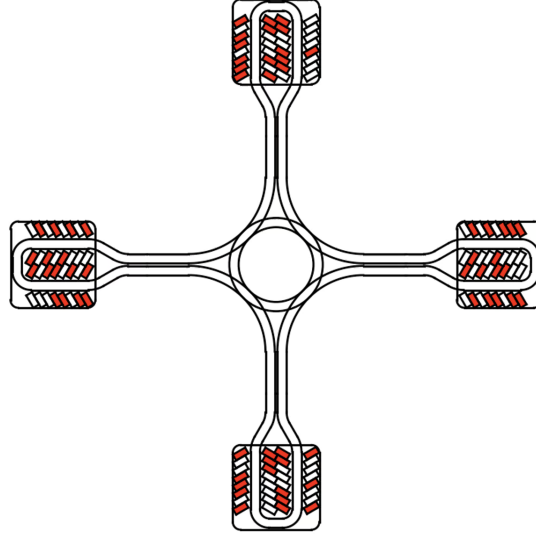


Figure 5.4: Visualization of the four-way roundabout intersection scenario. The red rectangles represent the origins of vehicles in the respective parking lots. Path conflicts primarily happen at roundabout merge points.

The *thruway with onramps and offramps* scenario, depicted in Figure 5.5, features a single-lane thruway with two onramps and two offramps. As in previous scenarios, this setting includes four parking lots: two are linked to the onramps, and the remaining two are connected to the offramps. Trips are generated by randomly connecting a parking spot at an onramp lot to a spot at an offramp lot. The configuration of the parking lots, path assignment, and initial departure times remain identical to those in the cross intersection scenario. This scenario extends the complexity of the cross intersection scenario by modeling arterial and highway driving constraints.



Figure 5.5: Visualization of the single-lane thruway with onramps and offramps scenario. The red rectangles represent the origins of vehicles in the respective parking lots. Path conflicts primarily happen at on-ramp merge points.

The 2×2 *intersection grid* scenario, depicted in Figure 5.6, comprises a medium-sized composite intersection network with two cross intersections and two roundabout intersec-

tions. Each parking lot can accommodate 64 vehicles, 32 of which are randomly selected as the origins for the vehicles. For path assignment, the shortest path algorithm is once again employed. In cases where multiple shortest paths exist, one is randomly selected from the viable options. As with previous scenarios, the initial departure time for each vehicle is uniformly set to zero. This scenario is designed to evaluate the scalability and generalizability of the algorithm on medium-sized problems.

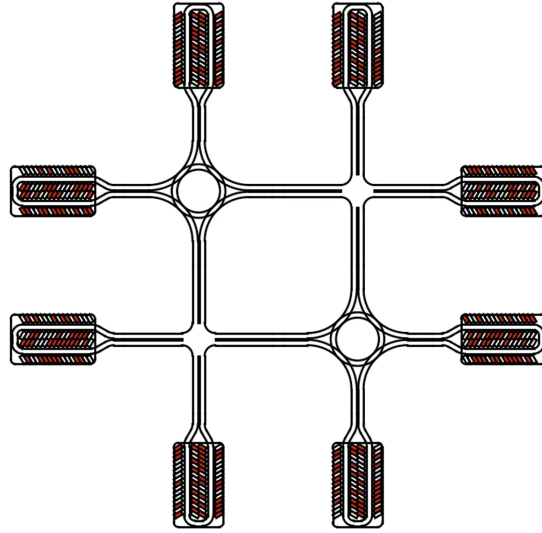


Figure 5.6: Visualization of the 2×2 intersection grid scenario. The red rectangles represent the origins of vehicles in the respective parking lots. Path conflicts primarily happen at 4-way cross intersections and roundabout merge points.

The *three-way intersection mesh* scenario, illustrated in Figure 5.7, features a medium-sized composite network with six three-way cross intersections. The configuration of the parking lots, trip generation, path assignment, and initial departure times are consistent with the 2×2 intersection grid scenario. This scenario represents a variation on the 2×2 intersection grid, substituting the conventional four-way intersection design with a non-conventional three-way design. This design seeks to emulate the organically formed road networks commonly found in moderately sized historical towns.

The *concentric hexagonal rings* scenario, depicted in Figure 5.8, features a medium-sized composite network consisting of two concentric hexagonal ring roads. The parking lot configuration, trip generation, path assignment, and initial departure times mirror those used in the 2×2 intersection grid scenario. This scenario is another variation of the 2×2 intersection grid scenario, incorporating a blend of three-way and four-way cross intersections. This design is inspired by historical metropolitan urban planning, where the road network expands from a central point within a city.

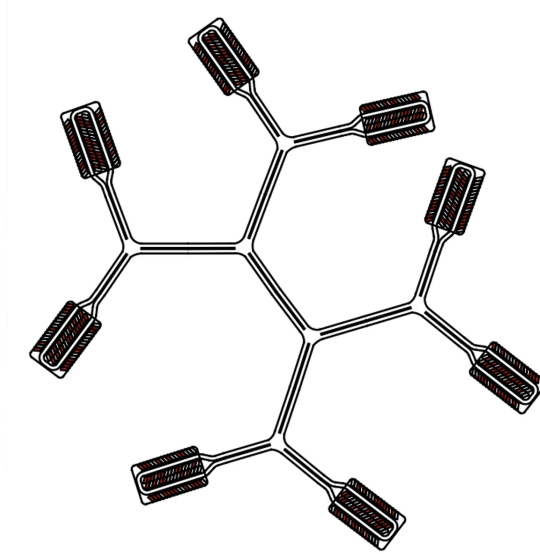


Figure 5.7: Visualization of the three-way cross intersection mesh scenario. The red rectangles represent the origins of vehicles in the respective parking lots. Path conflicts primarily happen at 3-way cross intersections.

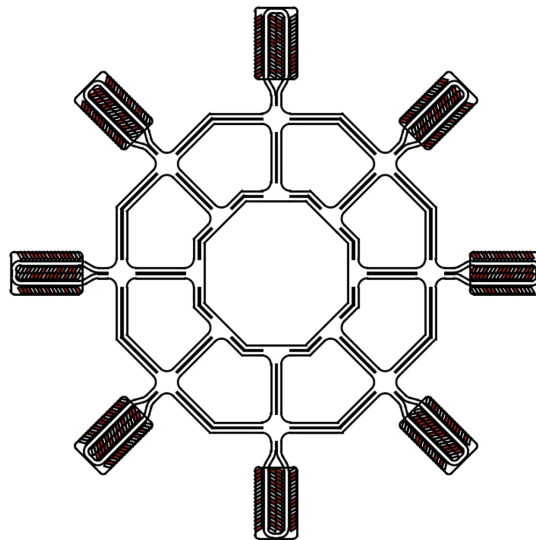


Figure 5.8: Visualization of the concentric hexagonal rings scenario. The red rectangles represent the origins of vehicles in the respective parking lots. Path conflicts primarily happen at 3-way and 4-way cross intersections.

The 8×8 intersection grid scenario, depicted in Figure 5.9, encompasses a large-sized composite network with 64 standard cross intersections. The configuration of the parking lots mirrors that used in the cross intersection scenario. Trip generation, path assignment, and initial departure times are consistent with the 2×2 intersection grid scenario. This scenario represents the ultimate variation of the 2×2 intersection grid scenario, featuring a significantly larger problem footprint. This design emulates modern metropolitan urban planning, where road networks are systematically divided into rectangular districts. This scenario serves as the final test of the algorithm’s numerical scalability.

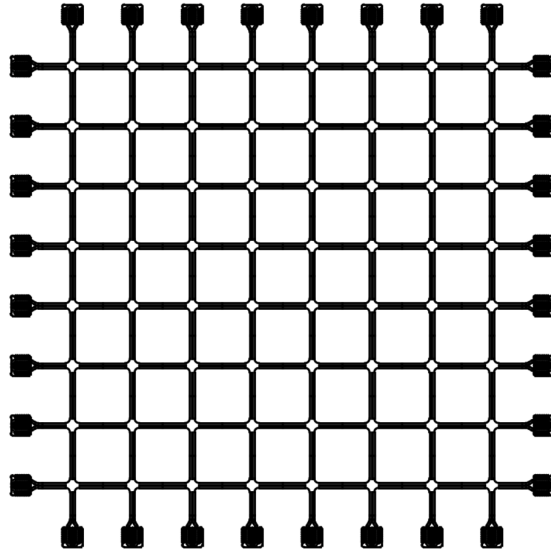


Figure 5.9: Visualization of the 8×8 intersection grid scenario. The red rectangles represent the origins of vehicles in the respective parking lots. Path conflicts primarily happen at 4-way cross intersections.

To speed up the numerics, for all scenarios, the optimization problem is segmented into sequences of subproblems, each involving four vehicles, with a maximum reservation memory depth of five solutions. All vehicles are standardized to be 4.6 m in length and 1.85 m in width. Additionally, a time buffer of approximately 0.26 s is added to each collision avoidance constraint to ensure a minimum time headway. The operational speed is uniformly set at 18 m/s. Each optimization problem is solved using Google OR Tools [93], with the SCIP solver serving as the backend [10, 11].

5.6.2 Numerical Results

These seven scenarios, varying in size, complexity, and topology, while being constructed in an idealized setting, are designed to have sufficient complexity to assess the effectiveness

of the algorithm in solving centralized scheduling problems in real-world settings. In the following paragraphs, we provide a high-level summary of the simulation results.

To demonstrate the qualitative behavior of the MILP algorithm in action, we present three snapshots of solutions for the cross intersection scenario, the roundabout intersection scenario, and the thruway with onramps and offramps scenario in Figure 5.10. These snapshots illustrate the algorithm’s capability to efficiently coordinate vehicles along potentially conflicting paths. As shown in the figure, the MILP manages to tightly pack vehicles in both space and time, minimizing the gaps between them, thereby maximizing throughput and ensuring collision-free movement.

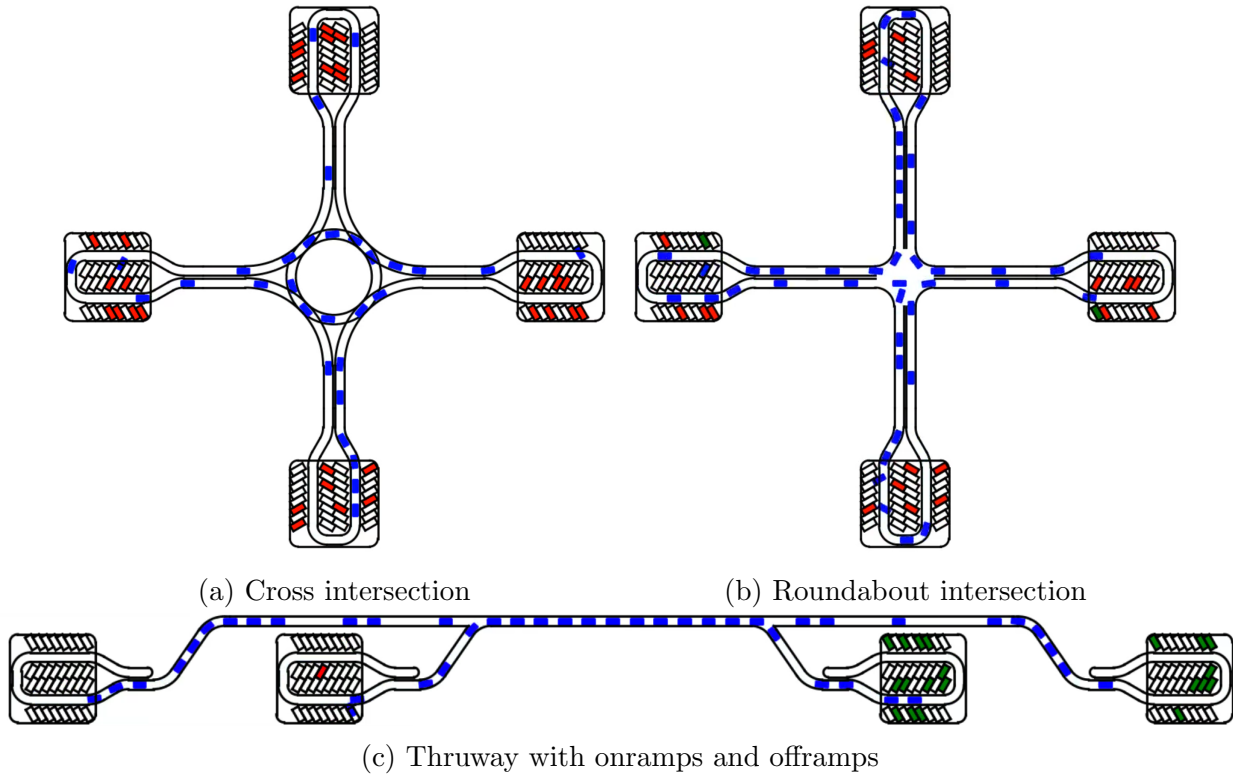


Figure 5.10: Example snapshots of running simulation of the cross intersection, roundabout intersection, and thruway with onramps and offramps scenarios, optimized by the MILP algorithm. Red rectangles represent vehicles parking at the original parking lots. Blue rectangles represent vehicles travelling between respective origins and destinations. Green rectangles represent vehicles parking at the destination parking lots.

The total number of trips and the overall fulfillment time for the seven simulated scenarios are presented in Table 5.1. The problem size, characterized by the number of variables and constraints, as well as the corresponding running times, are detailed in Table 5.2. From the data summarized in these tables, it is evident that the algorithm consistently finds effec-

tive solutions across all scenarios, while maintaining running times within a practical time frame. Specifically, the reservation heuristic effectively prevents an exponential increase in computation time as the problem size expands, particularly with respect to binary variables and constraints.

5.7 Discussion

In this section, we explore the qualitative behavior of the MILP algorithm and discuss future work. Firstly, we observe the absence of local inter-agent delays, a result of centralized control over all agents. Next, we outline two promising and practically relevant future research directions: 1) enhancing the MILP algorithm for robust real-time deployment, and 2) expanding the algorithm to include lane-changing capabilities.

5.7.1 Nonlocality of Multi-Agent Interactions

The coordination of all agents via the MILP algorithm leads to fundamentally nonlocal interactions among agents. This contrasts with the local interactions common in sequentially chained MASs and partially meshed MASs, where interactions inherently introduce delays into the inter-agent dynamics. Such delays can significantly affect the performance of multi-agent systems, potentially causing issues such as the formation of stop-and-go waves, discussed in detail in Chapters 2 and 3.

Another potential issue is the establishment of a safe speed limit, which can be derived using basic Newtonian physics. For a human driver with a reaction time of t_{delay} and a maximum braking capability of d_{max} , and assuming a leading vehicle at a distance Δs brakes hard at d_{max} , the theoretical maximum speed v_{max} is given by

$$v_{\text{max}} = \frac{\Delta s}{t_{\text{delay}}},$$

where exceeding v_{max} would make avoiding a collision impossible. In nonlocally synchronized systems, where information exchange occurs via fast wireless communication, t_{delay} is significantly reduced compared to locally synchronized systems, thereby increasing t_{max} .

5.7.2 Robust Real-Time Deployment

A limitation of the current MILP algorithm is its lack of robustness in real-time deployment. The optimization model, even with a reservation-based heuristic, links each agent to every other, making the entire system vulnerable if a single agent fails to adhere to its assigned schedule, potentially necessitating a global replan. This sensitivity is clearly impractical for real-world applications.

One temporary mitigation strategy is to incorporate a robustness margin into each collision constraint to reduce the likelihood of schedule deviations. While this can prevent

Table 5.1: Trip Count and Fulfillment Time of the Simulated Scenarios

Scenario	Trip Count	Fulfillment Time (s)
Cross	64	37.7
Roundabout	64	37.3
Thruway	64	56.1
2×2 Grid	256	130.8
Triangles	256	151.6
Hexagons	256	122.1
8×8 Grid	512	204.8

Table 5.2: Problem Sizes and Running Times of the Simulated Scenarios

Scenario	All Variables	Binary Variables	All Constraints	Binary Constraints	Running Time (s)	Branch and Bound
Cross	14022	10666	21518	11642	0.640	32
Roundabout	13805	10801	19488	10668	0.493	18
Thruway	19228	16156	19902	10878	0.703	422
2×2 Grid	194645	171765	164694	96822	3.359	64
Triangles	222689	198541	175074	103398	3.466	64
Hexagons	209153	180165	205052	118856	3.802	64
8×8 Grid	279443	217483	414150	229806	7.460	128

unnecessary replans, it reduces the overall system efficiency by decreasing road network capacity. Ultimately, this approach does not address the high coupling inherent in the model.

Another strategy is to reduce the degree of schedule coupling by planning within a fixed spatial horizon and replanning as this horizon is approached. Nevertheless, defining appropriate terminal costs and constraints to maintain safety and performance remains an open question.

5.7.3 Extension for Lane-Changing

Currently, the MILP algorithm does not accommodate lane changes, which presents a significant limitation. Integrating lane changes within a mixed-integer optimization framework is challenging due to the spontaneous nature of such maneuvers. One potential workaround is to restrict lane changes to predetermined locations along a multi-lane path, effectively simplifying the continuous problem into a discrete, albeit suboptimal, one. Alternatively, optimizing waypoint schedules prior to optimizing lane changes could facilitate a two-stage optimization process.

Chapter 6

Learning-Augmented Planning and Control

This chapter offers an alternative perspective on the planning and control of MASs through the lens of machine learning. While classical approaches have proven effective across various applications, the renaissance of machine learning since the late 2000s has introduced numerous powerful methods. Some of these methods offer promising generalizations to the issues considered in this dissertation. Specifically, this chapter explores a hybrid control approach that employs neural networks to augment MPC. We demonstrate that neural networks can be integrated with an MPC method without compromising any pre-existing stability guarantees. Furthermore, we empirically show that a well-trained neural network can significantly enhance the amortized efficiency of the overall hybrid control scheme.

6.1 Introduction

Model predictive control is a powerful method for controlling dynamical systems with constraints [96, 15]. It has been widely used in robotics, with applications ranging from ground [16] and aerial [3] vehicle maneuvers to humanoid [63] and quadruped [14] robot control.

When it comes to *implement* MPC, one has two conventional approaches: 1) implicit MPC, which solves an optimization problem in real time with a preferably efficient problem formulation and numerical scheme, 2) explicit MPC, which computes every solution of the optimization problem via parametric optimization *offline* and looks up the computed solutions *online*.

Explicit MPC often has faster running time than implicit MPC but requires expensive pre-computation. For problems of many variables and of intricate constraints, such pre-computation tends to demand a prohibitively large amount of time and memory.

Alternatively, one can approximate an implicit MPC controller with a neural network (NN) through supervised learning. Neural network models are more flexible when it comes

to pre-training and generally scales better to higher-dimensional problems. Nevertheless, to our best knowledge, the NN function approximation approach has yet to fully address the following open problems: 1) it is difficult, if not impossible, to guarantee that the approximation will always converge to a solution of bounded approximation errors; 2) it is challenging to establish closed-loop stability without any knowledge of the accuracy of the approximation.

Motivated by the above shortcomings of explicit MPC and NN function approximation methods and inspired by [78], we propose a triple-mode hybrid control scheme, named *Memory-Augmented MPC* (MAMPC). The core idea is to mix a costly MPC controller with an efficient linear quadratic regulator (LQR) controller and an efficient NN controller, whenever stability permits.

The simplicity of our method makes it possible for it to be analyzed theoretically for stability. Such stability guarantee is not a common quality among works that involve neural networks but is a necessity for safety-critical applications. Moreover, unlike conventional explicit MPC or function approximation methods, our method does not require any form of pre-computing. Instead, it is operational on day one, in spite of being inefficient, and learns to be efficient over time.

We summarize the main contributions of our work below.

- We present a novel controller design, i.e., Memory-Augmented MPC, in its standard form and two modified forms.
- We prove stability of Memory-Augmented MPC without imposing any condition on the approximation errors of the neural network.
- We demonstrate amortized computational efficiency of Memory-Augmented MPC via four numerical experiments.

The rest of the article is organized as follows. A brief literature review is provided in Section 6.2 to situate the work in context. Standard and modified versions of MAMPC are presented in Section 6.3. Theoretical properties of MAMPC, including stability, are proven in Section 6.4. Four numerical experiments are presented in Section 6.5 and discussed in Section 6.6.

6.2 Related Works

A major limitation of MPC has been a lack of computational efficiency as evident in applications such as the Atlas humanoid robot [63] and the MIT Cheetah robot [14]. To overcome this shortcoming, one typically has three approaches: 1) developing an efficient implicit MPC policy, 2) pre-compiling an explicit MPC policy, and 3) approximating a slow implicit MPC policy by an efficient surrogate policy such as a NN.

Implicit MPC approach implements the MPC policy by devising an efficient numerical algorithm for solving the optimization program. This is perhaps the most common approach

of implementing an MPC policy. Efficient implicit MPC controller often depends on custom optimization solvers that leverage structures specific to the problems they solve, such as [26, 75]. Implementation of such implicit MPC is usually done in a fast low-level programming language such as C. For example, see [42, 76]. See also [52] for practical considerations of MPC deployment in embedded systems. Moreover, recent advances in software and hardware enable implicit MPC controllers to be deployed in traditionally challenging problems, including [102, 31, 61].

When existing implicit MPC methods do not provide satisfactory latency, one sometimes resorts to explicit MPC, whenever such alternative is applicable. Explicit MPC approach implements the MPC policy by pre-computing the solution of the optimization problem and looking up the solution in real time. At its core, explicit methods achieve reduced running time at the cost of expensive pre-computation and increased memory footprint. To reduce memory footprint, numerous works have been developed [115, 64]. For survey of explicit MPC can be found in [2]. Nevertheless, due to poor scalability of the pre-computation step, explicit MPC has more limited use cases than implicit MPC.

When the explicit MPC method fails to apply to a problem, one may consider another alternative, namely, function approximation. Function approximation approach implements an MPC policy by approximating the costly MPC control law with an efficient surrogate model, such as a NN, and uses that surrogate model for fast online deployment. Early works in this direction include [90, 89], which has described how to design a surrogate NN controller and has proposed conditions to guarantee its closed-loop stability. Later development has focused on guaranteeing different theoretical properties of the NN-controlled closed-loop system. For example, [137] proposes a NN method with *probabilistic* optimality bounds; [92] has designed a projection operator that can modify a NN controller to a stabilizing one for linear systems; and [54] proposes a mixed-integer linear program to certify stability and feasibility of ReLU-activated NN controllers. Besides, rather than replacing the implicit MPC controller, [21] uses a NN to warm start an optimization solver, which may then speed up convergence of the implicit MPC solver. For a comprehensive study, readers may consult [138]. Applications of function approximation methods have found many successes in practice, such as [84].

It is worth noting that closed-loop stability of the function approximation methods generally requires reasonable convergence of NN model to the original implicit MPC control law. Nevertheless, training a NN is commonly done through a black-box solver, such as [59], the convergence of which is generally not guaranteed. In contrary, we propose a novel function approximation approach named Memory-Augmented MPC, which is always stable and always guarantees constraint satisfaction, without imposing *any* condition on the convergence and optimality of the NN. The accuracy of the NN only affects running time: the more accurate the NN is, the more efficient the overall control scheme is. As a result, it can be deployed without any training, with the expectation that it will attain computational efficiency over time via learning.

As a final remark, a quick comparison of MAMPC with a few key existing works is in order. We acknowledge that our method is inspired by the seminal early work in dual-mode

MPC [78], which combines an MPC with a LQR. An advantage of our approach over [78] is that it could potentially provide more reduction in latency with the help of a fast intermediate NN mode. As detailed in Section 6.6, with parallelization, MAMPC is *at worst* equivalent to [78]. Moreover, compared to the techniques in [92, 21], which only apply to linear plants with convex quadratic objectives, our method works for any nonlinear problems.

6.3 Composing MPC, LQR, and NN

We begin the section by first introducing some basic notations, terminologies, and a key stability theorem of MPC. Next, we present the standard construction in MAMPC, followed by two variations of such construction, namely, alternating-authority MAMPC and way-point MAMPC.

6.3.1 Basics of MPC

Consider the discrete-time time-invariant constrained dynamical systems of the form

$$\mathbf{x}[i+1] = f(\mathbf{x}[i], \mathbf{u}[i]), \quad i \in \mathbb{Z}_{\geq 0} \quad (6.1)$$

where

$$\begin{aligned} \mathbf{x}[0] &= \mathbf{x}_0 \quad \text{for some } \mathbf{x}_0 \in \mathbb{R}^n, \\ (\mathbf{x}[i], \mathbf{u}[i]) &\in \mathbb{A}, \quad \forall i > 0, \end{aligned}$$

where $\mathbb{A} \subset \mathbb{R}^n \times \mathbb{R}^m$ is a closed set defining the system constraints, $\mathbf{x}[i]$ is the state at time index i , $\mathbf{u}[i]$ is the input at time index i , $f : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$ is continuously differentiable in both its components, with an equilibrium point at $(\mathbf{0}_n, \mathbf{0}_m)$, i.e., $f(\mathbf{0}_n, \mathbf{0}_m) = \mathbf{0}_n$, and the linearized time-invariant system of $f(\cdot, \cdot)$ around the equilibrium is stabilizable.

Let $\mathbb{X}$ be state constraint set and $\mathbb{U}$ be input constraint set. We consider system constraints of the following form $\mathbb{A} := \mathbb{X} \times \mathbb{U}$. The system dynamics is a map $f : \mathbb{A} \rightarrow \mathbb{X}$ and any control law is a map $u : \mathbb{X} \rightarrow \mathbb{U}$.

Consider a finite planning and control horizon N . Let $\mathcal{X}^{1:N} := (\mathbf{x}[1], \dots, \mathbf{x}[N])$ be a trajectory of the system and $\mathcal{U}^{0:N-1} := (\mathbf{u}[0], \dots, \mathbf{u}[N-1])$ be the corresponding ordered control sequence.

Provided with the dynamical system (6.1), an MPC is an optimal control law implicitly

defined through the following set of optimization problems

$$\begin{aligned}
\min_{\mathcal{X}_i^{1:N}, \mathcal{U}_i^{0:N-1}} \quad & \sum_{k=0}^{N-1} c(\mathbf{x}[k|i], \mathbf{u}[k|i]) + c_f(\mathbf{x}[N|i]) \\
\text{s.t.} \quad & \mathbf{x}[0|i] = \mathbf{x}[i], \\
& \mathbf{x}[k+1|i] = f(\mathbf{x}[k|i], \mathbf{u}[k|i]), \\
& (\mathbf{x}[k|i], \mathbf{u}[k|i]) \in \mathbb{A}, \\
& \mathbf{x}[N|i] \in \mathbb{X}_f, \\
& k = 0, \dots, N-1,
\end{aligned} \tag{6.2}$$

where N is planning and control horizon, $c : \mathbb{A} \rightarrow \mathbb{R}_+$ is a continuous stage cost function, $c_f : \mathbb{X}_f \rightarrow \mathbb{R}_+$ is a continuous terminal cost function, $\mathbf{x}[k|i]$ is the predicted state k steps ahead of *present* state $\mathbf{x}[i]$, $\mathbf{u}[k|i]$ is the anticipated input that generates $\mathbf{x}[k+1|i]$ from $\mathbf{x}[k|i]$, $\mathcal{X}_i^{1:N} := (\mathbf{x}[1|i], \dots, \mathbf{x}[N|i])$ and $\mathcal{U}_i^{0:N-1} := (\mathbf{u}[0|i], \dots, \mathbf{u}[N-1|i])$ are two sets of optimization variables corresponding to predicted states and anticipated inputs, $\mathbb{X}_f$ is a terminal constraint set containing the origin usually designed to guarantee asymptotic stability. If optimization problem (6.2) is feasible and admits an optimal solution $(\mathcal{X}_i^{1:N*}, \mathcal{U}_i^{0:N-1*})$, then the MPC control law selects the first element of $\mathcal{U}_i^{0:N-1*}$ as the input at time index i , that is,

$$u_{\text{MPC}}(\mathbf{x}[i]) := \mathbf{u}^*[0|i] = \mathcal{U}_i^{0:N-1*}[0]. \tag{6.3}$$

This process is repeated at the next time index $i+1$, until some termination criterion is met. As a result, we obtain a closed-loop system

$$\mathbf{x}[i+1] = f(\mathbf{x}[i], u_{\text{MPC}}(\mathbf{x}[i])), \quad i \in \mathbb{Z}_{\geq 0}, \tag{6.4}$$

with initial condition $\mathbf{x}[0] = \mathbf{x}_0 \in \mathbb{X}_0 := \{\mathbf{x}_0 \in \mathbb{R}^n \mid \text{problem (6.2) is feasible}\}$ is the set of all admissible initial states. For well-posedness of MPC, we require $\mathbb{X} \subseteq \mathbb{X}_0$.

Let $J(\mathbf{x}[i]) := \sum_{k=0}^{N-1} c(\mathbf{x}[k|i], \mathbf{u}[k|i]) + c_f(\mathbf{x}[N|i])$ be the cumulative cost and $J^*(\mathbf{x}[i]) := \sum_{k=0}^{N-1} c(\mathbf{x}^*[k|i], \mathbf{u}^*[k|i]) + c_f(\mathbf{x}^*[N|i])$ be the optimal cumulative cost. By Lyapunov stability theorem, we can derive the following sufficient condition for asymptotic stability of MPC as in Theorem 12.2 in [15].

Theorem 2 (Local Asymptotic Stability of MPC). Assume that

- The stage cost and terminal cost are continuous and positive definite.
- $\mathbb{X}_f$ is control invariant.
- For all $\mathbf{x} \in \mathbb{X}_f$,

$$\min_{\mathbf{u} : \mathbf{u} \in \mathbb{U}, f(\mathbf{x}, \mathbf{u}) \in \mathbb{X}_f} c(\mathbf{x}, \mathbf{u}) - c_f(\mathbf{x}) + c_f(f(\mathbf{x}, \mathbf{u})) \leq 0.$$

then the MPC problem (6.2) is persistently feasible and the closed-loop system (6.4) is locally asymptotically stable with respect to the origin. In addition, $J^*(\cdot)$ is a Lyapunov function and $\mathbb{X}_0$ is a positively invariant region of attraction.

With the above definitions and theorem, we are ready to describe the composition rules of MPC, LQR, and NN.

6.3.2 Composition Rules

Provided with a computationally costly implicit MPC u_{MPC} that satisfies Theorem (2), we propose a hybrid control scheme, namely, *Memory-Augmented Model Predictive Control* (MAMPC), $u_{\text{MAMPC}} : \mathbb{X} \rightarrow \mathbb{U}$, by augmenting u_{MPC} with a LQR controller u_{LQR} and a NN controller u_{NN} , where the LQR controller and the NN controller are independently developed from the implicit MPC. We first describe how the LQR and NN controllers are derived from the implicit MPC and then present ways to combine these controllers.

6.3.2.1 LQR from MPC

Provided with an MPC, we derive an infinite-time LQR controller by 1) linearizing the system dynamics around the equilibrium, 2) removing stage constraints, terminal constraint, and terminal cost, 3) taking planning and control horizon N to ∞ , and 4) replacing stage cost $c(\cdot)$ with a positive definite quadratic cost, if it is not already so in the original MPC.

Formally, the MPC-induced LQR problem is defined as follows

$$\begin{aligned} \min_{\mathbf{x}_i^{1:\infty}, \mathbf{u}_i^{0:\infty}} \quad & \sum_{k=0}^{\infty} \mathbf{x}^\top[k|i] \mathbf{Q} \mathbf{x}[k|i] + \mathbf{u}^\top[k|i] \mathbf{R} \mathbf{u}[k|i] \\ \text{s.t.:} \quad & \mathbf{x}[0|i] = \mathbf{x}[i], \\ & \mathbf{x}[k+1|i] = \mathbf{A} \mathbf{x}[k|i] + \mathbf{B} \mathbf{u}[k|i], \\ & k \in \mathbb{Z}_+, \end{aligned} \tag{6.5}$$

where $\mathbf{A} := \frac{\partial f}{\partial \mathbf{x}}|_{(\mathbf{0}, \mathbf{0})} \in \mathbb{R}^{n \times n}$, $\mathbf{B} := \frac{\partial f}{\partial \mathbf{u}}|_{(\mathbf{0}, \mathbf{0})} \in \mathbb{R}^{n \times m}$ are linearized system dynamics around the equilibrium, and $\mathbf{Q} \in \{\mathbf{N} \in \mathbb{R}^{n \times n} \mid \mathbf{N} \succeq 0\}$, $\mathbf{R} \in \{\mathbf{M} \in \mathbb{R}^{m \times m} \mid \mathbf{M} \succeq 0\}$ are weighting matrices that measure significance of state deviations and control costs, respectively.

It can be shown that the optimal LQR control law for an initial state $\mathbf{x}$ is a linear map

$$u_{\text{LQR}}(\mathbf{x}) = -\mathbf{K} \mathbf{x} \tag{6.6}$$

for some $\mathbf{K} \in \mathbb{R}^{m \times n}$ [6]. Furthermore, if the linearized system $(\mathbf{A}, \mathbf{B})$ is *stabilizable* and if $\mathbf{A} - \mathbf{B} \mathbf{K}$ is Hurwitz, then the closed-loop system

$$\mathbf{x}[i+1] = f(\mathbf{x}[i], u_{\text{LQR}}(\mathbf{x}[i])), \tag{6.7}$$

is *locally* asymptotically stable near the equilibrium with a *positively invariant* region of attraction $\mathcal{R}_{\text{LQR}}$, as shown in Theorem 4.7 in [57]. Specifically, $\mathcal{R}_{\text{LQR}}$ is a set that if $\mathbf{x}_0 \in \mathcal{R}_{\text{LQR}}$ and $\mathbf{x}[0] = \mathbf{x}_0$, then $\mathbf{x}[i] \in \mathcal{R}_{\text{LQR}}, \forall i \in \mathbb{Z}_{>0}$ and $\lim_{i \rightarrow \infty} \mathbf{x}[i] = \mathbf{0}$.

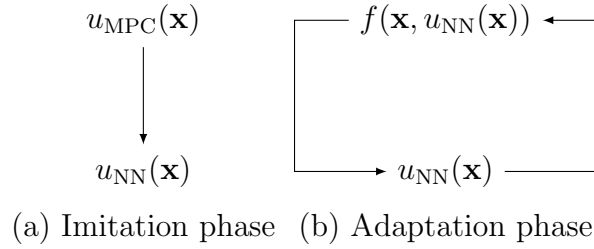


Figure 6.1: Schematics of the learning processes. (a) Imitation phase: $(\mathbf{x}, u_{\text{MPC}}(\mathbf{x}))$ provides data to train the NN controller u_{NN} in a supervised learning setting. (b) Adaptation phase (optional): u_{NN} interacts with the environment $f(\cdot, \cdot)$ to enhance control effectiveness in a reinforcement learning setting.

As implied by local asymptotic stability, the largest possible $\mathcal{R}_{\text{LQR}}$ is nonempty in theory. In practice, however, estimating $\mathcal{R}_{\text{LQR}}$ is a challenging problem. To this end, several methods have been proposed, such as [22, 122, 25].

Because the LQR controller only requires one matrix multiplication within a single controller step, it is usually significantly more efficient than the implicit MPC in terms of per-step computation. Hence, it is usually economical to switch to the LQR controller when the state is in the region of attraction of the LQR, that is, $\mathbf{x}[i] \in \mathcal{R}_{\text{LQR}}$.

6.3.2.2 NN from MPC

Provided with an MPC, we derive a NN controller 1) by imitating the MPC policy $u_{\text{MPC}}(\mathbf{x})$ through supervised learning and 2) *optionally* by interacting with the environment through reinforcement learning, as illustrated in Figure 6.1.

In the *imitation* phase, the NN training is a supervised learning problem usually of the following mean squared form

$$\min_{\mathcal{W}} \quad \frac{1}{M} \sum_{\mathbf{x} \in \mathcal{D}} \|u_{\text{MPC}}(\mathbf{x}) - \phi(\mathbf{x} \mid \mathcal{W})\|_2^2, \quad (6.8)$$

where $\phi : \mathbb{R}^n \rightarrow \mathbb{R}^m$ is a NN model with weights $\mathcal{W}$ and $\mathcal{D}$ is a set of M states randomly sampled from $\mathbb{X}_0$.

Therefore, the resulting NN control law is

$$u_{\text{NN}}(\mathbf{x}) := \phi(\mathbf{x} \mid \overline{\mathcal{W}}), \quad (6.9)$$

where $\overline{\mathcal{W}}$ is a suboptimal solution to problem (6.8), generally estimated via a gradient method such as Adam [59].

In the *adaptation* phase, it is possible to fine tune the NN further through a reinforcement learning (RL) algorithm. A reasonable choice for environment may be the dynamic model

used by the MPC. A good choice of the reward function may be the cost function of the MPC. The training setup varies based on the specific choice of the RL algorithm. For a comprehensive review of reinforcement learning methods, please refer to [118]. We note that the adaptation phase may be skipped if the imitation phase is sufficiently effective or if the adaptation phase does not bring any improvement.

In practice, people find that implicit MPC controllers can be well approximated by a more efficient NN controller [90]. However, it is not easy to certify stability of the closed-loop system without bounding approximation errors between the NN controller and the original MPC controller.

$$\mathbf{x}[i+1] = f(\mathbf{x}[i], u_{\text{NN}}(\mathbf{x}[i])). \quad (6.10)$$

To solve the problem, we show that by composing the NN controller with the MPC and the LQR in a hybrid control scheme, we can prove the local asymptotic stability of the closed-loop system, even if the NN is *random*. Next, we present the basic form of our method, i.e., the standard MAMPC.

6.3.2.3 Standard Memory-Augmented MPC

The hybrid control scheme combines the LQR controller and the NN controller with the original MPC controller. At every control step, if the state is in the region of attraction of the LQR, apply the LQR controller; else, we simulate the closed-loop system (6.10) for up to N_{LQR} steps: if there *exists* a step $j \leq N_{\text{LQR}}$ such that the state reaches within the region of attraction of LQR and that up until the j^{th} step the simulated system does not violate any stage constraint of the MPC, apply the NN controller; otherwise, if the state is within the admissible initial states of the MPC, apply the implicit MPC.

Formally, the hybrid controller is defined as follows

$$u_{\text{MAMPC}}(\mathbf{x}) := \begin{cases} u_{\text{LQR}}(\mathbf{x}), & \text{if } \mathbf{x} \in \mathcal{R}_{\text{LQR}}, \\ \hline u_{\text{NN}}(\mathbf{x}), & \text{if } \mathbf{x} \in \mathbb{X}_0 \setminus \mathcal{R}_{\text{LQR}} \text{ and} \\ & \exists i = 1, \dots, N_{\text{LQR}}, \quad \mathbf{y}[i] \in \mathcal{R}_{\text{LQR}} \text{ and} \\ & \forall j = 0, \dots, i, \quad (\mathbf{y}[j], u_{\text{NN}}(\mathbf{y}[j])) \in \mathbb{A}, \\ \hline u_{\text{MPC}}(\mathbf{x}), & \text{otherwise,} \end{cases}$$

where $\mathbf{y}$ is the *simulated* state by numerically stepping: $\mathbf{y}[i] = f(\mathbf{y}[i-1], u_{\text{NN}}(\mathbf{y}[i-1]))$ with $\mathbf{y}[0] = \mathbf{x}$, $\mathcal{R}_{\text{LQR}}$ is designed to be a subset of $\mathbb{X}_0$, and $N_{\text{LQR}} \in \mathbb{Z}_{>0}$ is the verification horizon of $\mathcal{R}_{\text{LQR}}$. Note that compliance with state and input constraints are satisfied by construction: we explicit check constraint satisfaction via forward simulation.

As we show in Section 6.4, the stability of NN relies on verifying whether the system reaches within $\mathcal{R}_{\text{LQR}}$ through forward simulation. However, this approach is ineffective for systems that are sensitive to initial conditions and systems that require a large number of steps to converge to the origin. To address these two challenges, we introduce the following two variants of MAMPC.

6.3.2.4 Alternating-Authority Memory-Augmented MPC

The first variant of MAMPC is designed specifically for chaotic systems. A *chaotic system* is loosely defined as a system that is very sensitive to initial condition and control input. Qualitatively speaking, the same chaotic system that begins with two slightly different initial conditions and control sequences will arrive at significantly different terminal states. As a result, it is very challenging to stabilize a chaotic system with any NN controller because the function approximator will inevitably produce random control errors which can easily deter the system from its stabilizing trajectory. In this case, it is useful to modify the hybrid control scheme to bound error accumulation induced by the NN by periodically alternating between the NN controller and the MPC controller. We name this modified version of MAMPC as *alternating-authority* MAMPC, which is formally described below.

$$u_{\text{MAMPC}}^{\text{AA}}(\mathbf{x}) := \begin{cases} u_{\text{LQR}}(\mathbf{x}), & \text{if } \mathbf{x} \in \mathcal{R}_{\text{LQR}}, \\ \hline u_{\text{NN}}(\mathbf{x}), & \text{if } \mathbf{x} \in \mathbb{X}_0 \setminus \mathcal{R}_{\text{LQR}} \text{ and} \\ & u_{\text{NN}}(\mathbf{x}) \in \mathbb{U} \wedge \mathbf{y}[1] \in \mathbb{X} \text{ and} \\ & i \pmod{i_d} \neq 0, \\ \hline u_{\text{MPC}}(\mathbf{x}), & \text{otherwise,} \end{cases}$$

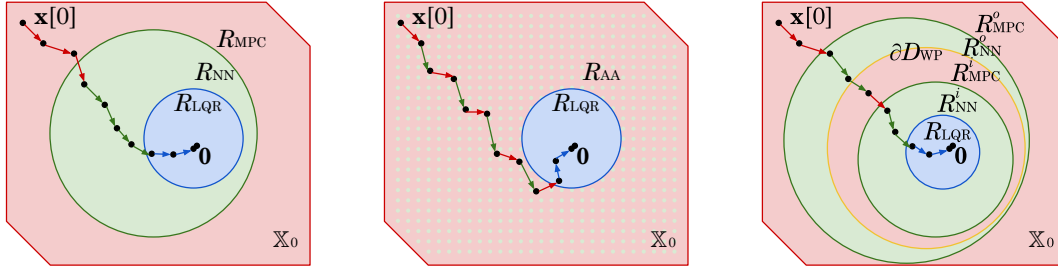
where $i_d \in \mathbb{Z}_{\geq 1}$ is the period of MPC defaulting and $\mathcal{R}_{\text{LQR}} \subset \mathbb{X}_0$.

The main variation from the standard MAMPC is that we do not verify if the system falls within $\mathcal{R}_{\text{LQR}}$ in N_{LQR} steps, as forward simulation is not as reliable for chaotic systems. This implies that i_d should be chosen as a small positive number. Otherwise, prediction produced by forward simulation will be a reliable estimate of the future. However, note that reducing i_d will prolong running time. Therefore, a rule of thumb of designing i_d should be to choose it as large as the stability permits.

In addition, we remark that the closed-loop system has a smaller region of attraction compared to standard MAMPC, since we do not verify whether the system enters $\mathcal{R}_{\text{LQR}}$ through forward simulation.

6.3.2.5 Way-Point Memory-Augmented MPC

Another variant of MAMPC is designed specifically for slow systems. A *slow system* is loosely defined as a system that requires a substantial number of steps to steer close to the origin. For slow systems, it is difficult to choose an effective verification horizon N_{LQR} : a small N_{LQR} may be too short to predict convergence of the NN controller; a large N_{LQR} may incur computational overhead and thus render the hybrid control inefficient. To address this problem, we propose a second variant of MAMPC by introducing a “way-point” set, $\mathcal{D}_{\text{WP}}$, and enabling the NN controller to be applied as long as the anticipated trajectory falls with that way-point set. We name this variant of MAMPC as *way-point* MAMPC, which is



(a) Standard MAMPC (b) Alternating-Authority MAMPC (c) Way-Point MAMPC

Figure 6.2: An illustrative example of how different variants of MAMPC *could* steer a system toward equilibrium. (a) Standard MAMPC: a triple-mode hybrid control composed of an MPC, a NN, and a LQR. (b) Alternating-authority MAMPC: a variant of the standard MAMPC, modified for chaotic systems. (c) Way-point MAMPC: a variant of the standard MAMPC, modified for slow systems. Solid red regions represent $\mathbb{X}_0$; solid green regions represent $\mathcal{R}_{\text{NN}}$; *green dots* indicate periodic controller switching region in alternating-authority MAMPC; and solid blue regions represent $\mathcal{R}_{\text{LQR}}$. A black dot represents a state and an arrow represents a controlled state transition. The color of arrows indicates which sub-controller is invoked during that control step: red is MPC; green is NN; and blue is LQR. The yellow line represents $\partial\mathcal{D}_{\text{WP}}$. Note that the above examples by no means exhaustive. For instance, it is possible that $\mathcal{R}_{\text{NN}} = \emptyset$.

formulated as follows.

$$u_{\text{MAMPC}}^{\text{WP}}(\mathbf{x}) := \begin{cases} u_{\text{LQR}}(\mathbf{x}), & \text{if } \mathbf{x} \in \mathcal{R}_{\text{LQR}}, \\ \hline u_{\text{NN}}(\mathbf{x}), & \begin{array}{l} \text{if } \mathbf{x} \in \mathcal{D}_{\text{WP}} \setminus \mathcal{R}_{\text{LQR}} \text{ and} \\ \exists i = 1, \dots, N_{\text{LQR}}, \quad \mathbf{y}[i] \in \mathcal{R}_{\text{LQR}} \text{ and} \\ \forall j = 0, \dots, i, \quad (\mathbf{y}[j], u_{\text{NN}}(\mathbf{y}[j])) \in \mathbb{A}_{\text{WP}}, \end{array} \\ \hline u_{\text{NN}}(\mathbf{x}), & \begin{array}{l} \text{if } \mathbf{x} \in \mathbb{X}_0 \setminus \mathcal{D}_{\text{WP}} \text{ and} \\ \exists i \in 1, \dots, N_{\text{WP}}, \quad \mathbf{y}[i] \in \mathcal{D}_{\text{WP}} \text{ and} \\ \forall j \in 0, \dots, i, \quad (\mathbf{y}[j], u_{\text{NN}}(\mathbf{y}[j])) \in \mathbb{A}, \end{array} \\ \hline u_{\text{MPC}}(\mathbf{x}), & \text{otherwise,} \end{cases}$$

where $\mathcal{D}_{\text{WP}}$ is a compact set that contains $\mathcal{R}_{\text{LQR}}$, $\mathbb{A}_{\text{WP}} := \mathcal{D}_{\text{WP}} \times \mathbb{U}$, and $N_{\text{WP}} \in \mathbb{Z}_{>0}$ is the verification horizon of $\mathcal{D}_{\text{WP}}$.

At minimum, we require that the way-point set contains the region of attraction of the LQR, that is, $\mathcal{R}_{\text{LQR}} \subset \mathcal{D}_{\text{WP}}$. Additional improvement of the way-point set design can be performed through search. For example, one can parameterize $\mathcal{D}_{\text{WP}}$ with one or more

parameters and search for a set of parameters that produces the most efficient closed-loop performance.

Besides, note that introduction of a way-point set may reduce the area of the region of attraction of the closed-loop system.

6.4 Theoretical Analysis

In this section, we prove the stability of the three MAMPC methods proposed above. We also remark on the robustness of the methods and the necessity of the fail-safe MPC mode.

6.4.1 Stability of MAMPC

We show that the three MAMPC methods are locally asymptotically stable with slightly different stability properties. We first prove that standard MAMPC is locally asymptotically stable in $\mathbb{X}_0$, as stated in the theorem below.

Theorem 3 (Stability of Standard MAMPC). For every MPC problem of the form (6.2), there always exists a u_{MAMPC} of the form (6.11). Furthermore, the closed-loop system

$$\mathbf{x}[i+1] = f(\mathbf{x}[i], u_{\text{MAMPC}}(\mathbf{x}[i])), \quad i \in \mathbb{Z}_{\geq 0}, \quad (6.11)$$

with $\mathbf{x}[0] = \mathbf{x}_0 \in \mathbb{X}_0$ is locally asymptotically stable in $\mathbb{X}_0$.

Proof. Existence of MAMPC depends on existence of LQR and NN. A LQR controller of the form (6.5) can always be derived from (6.2) because 1) f is continuously differentiable, so it can always be linearized to produce the $\mathbf{A}, \mathbf{B}$ matrices in (6.5); 2) the linearized system $(\mathbf{A}, \mathbf{B})$ is by assumption *stabilizable*, so region of attraction of LQR is nonempty, i.e., $\mathcal{R}_{\text{LQR}} \neq \emptyset$; 3) as for $\mathbf{Q}, \mathbf{R}$ we only require $\mathbf{Q} \succeq 0, \mathbf{R} \succ 0$, which is always possible; 4) lastly, optimization problem (6.5) is just a relaxation of problem (6.2), which can always be solved.

Meanwhile, a NN controller trivially exists because we do not require NN to satisfy any properties other than the basic definition of a NN. This completes the first part of the proof on existence.

To prove local asymptotic stability of standard MAMPC in $\mathbb{X}_0$, we partition $\mathbb{X}_0$ into three regions as follows

$$\mathbb{X}_0 = \mathcal{R}_{\text{LQR}} \cup \mathcal{R}_{\text{NN}} \cup \mathcal{R}_{\text{MPC}},$$

where $\mathcal{R}_{\text{NN}}$ is the set of states where the NN is invoked and $\mathcal{R}_{\text{MPC}}$ is the rest of states in $\mathbb{X}_0$, i.e., $\mathcal{R}_{\text{MPC}} := \mathbb{X}_0 \setminus (\mathcal{R}_{\text{LQR}} \cup \mathcal{R}_{\text{NN}})$. By construction, $\mathcal{R}_{\text{LQR}}$, $\mathcal{R}_{\text{NN}}$, and $\mathcal{R}_{\text{MPC}}$ are mutually disjoint.

For every state $\mathbf{x}[0] \in \mathcal{R}_{\text{LQR}}$, u_{LQR} is invoked and the closed-loop system (6.11) will be equivalent to (6.7), which is locally asymptotically stable.

For every state $\mathbf{x}[0] \in \mathcal{R}_{\text{NN}}$, u_{NN} is invoked and the closed-loop system (6.11) will be equivalent to (6.10) until the state enters $\mathcal{R}_{\text{LQR}}$ in no more than N_{LQR} steps. The system

will stay in $\mathcal{R}_{\text{LQR}}$ and will be taken to the origin by u_{LQR} as $i \rightarrow \infty$. Consequently, the closed-loop system (6.11) is locally asymptotically stable in $\mathcal{R}_{\text{MPC}} \cup \mathcal{R}_{\text{NN}}$.

For every state $\mathbf{x}[0] \in \mathcal{R}_{\text{MPC}}$, u_{MPC} is invoked and the closed-loop system (6.11) will be equivalent to (6.4) for at least one time step. In the next time step, if the state enters $\mathcal{R}_{\text{LQR}} \cup \mathcal{R}_{\text{NN}}$, then the closed-loop system will converge to the origin as it is locally asymptotically stable in $\mathcal{R}_{\text{LQR}} \cup \mathcal{R}_{\text{NN}}$; otherwise, $\mathcal{R}_{\text{MPC}}$ will be invoked again. Because the closed-loop system (6.4) is locally asymptotically stable, even if MPC never brings the state into $\mathcal{R}_{\text{NN}}$, it will eventually steer the system into $\mathcal{R}_{\text{LQR}}$ in finite time, which will then be taken to the origin by u_{LQR} . Consequently, the closed-loop system (6.11) is locally asymptotically stable in $\mathbb{X}_0$. $\square$

To provide geometric intuition on how a MAMPC achieves local asymptotic stability, we illustrate schematically in Figure 6.2a how a MAMPC *could* steer a system from an initial state to the equilibrium at zero. A state in $\mathcal{R}_{\text{MPC}}$ is first steered by u_{MPC} into $\mathcal{R}_{\text{NN}}$, which is then steered by u_{NN} into $\mathcal{R}_{\text{LQR}}$, after which is steered to zero by u_{LQR} . Note that the example demonstrated in Figure 6.2a is only a hypothetical particular case of MAMPC closed-loop behaviors.

Next, for alternating-authority MAMPC, the closed-loop system no longer has guaranteed local asymptotic stability in $\mathbb{X}_0$, because it does not check whether NN shoots inside $\mathcal{R}_{\text{LQR}}$. Rather, we can only show that the system is locally asymptotically stable in $\mathcal{R}_{\text{LQR}}$ and will never escape $\mathbb{X}_0$, as stated the theorem below.

Theorem 4 (Stability of Alternating-Authority MAMPC). For every MPC problem of the form (6.2), there always exists a $u_{\text{MAMPC}}^{\text{AA}}$ of the form (6.11). Furthermore, the closed-loop system

$$\mathbf{x}[i+1] = f(\mathbf{x}[i], u_{\text{MAMPC}}^{\text{AA}}(\mathbf{x}[i])), \quad i \in \mathbb{Z}_{\geq 0}, \quad (6.12)$$

with $\mathbf{x}[0] = \mathbf{x}_0 \in \mathbb{X}_0$ is locally asymptotically stable in $\mathcal{R}_{\text{LQR}}$. Besides, for every initial condition $\mathbf{x}[0] \in \mathbb{X}_0$, $\mathbf{x}[i] \in \mathbb{X}_0$ for all $i \in \mathbb{Z}_+$.

Proof. The first part of the proof on existence is identical to the one in Theorem 3.

To prove stability of alternating-authority MAMPC, we partition $\mathbb{X}_0$ into two disjoint regions $\mathbb{X}_0 = \mathcal{R}_{\text{LQR}} \cup \mathcal{R}_{\text{AA}}$, where $\mathcal{R}_{\text{AA}} := \mathbb{X}_0 \setminus \mathcal{R}_{\text{LQR}}$.

For every state $\mathbf{x}[0] \in \mathcal{R}_{\text{LQR}}$, the closed-loop system (6.12) is identical to closed-loop system (6.11), which is locally asymptotically stable, as shown in Theorem (3).

Next, we prove by contradiction that the system will never escape $\mathbb{X}_0$. Suppose there exists an escaping control $\bar{\mathbf{u}}$ such that $\bar{\mathbf{x}} \in \mathbb{X}_0$ but $f(\bar{\mathbf{x}}, \bar{\mathbf{u}}) \notin \mathbb{X}_0$. This escaping control $\bar{\mathbf{u}}$ must be produced by either u_{MPC} , u_{NN} , or u_{LQR} , that is, $\bar{\mathbf{u}} = u_{\text{MPC}}(\bar{\mathbf{x}})$, $\bar{\mathbf{u}} = u_{\text{NN}}(\bar{\mathbf{x}})$, or $\bar{\mathbf{u}} = u_{\text{LQR}}(\bar{\mathbf{x}})$. If $\bar{\mathbf{u}} = u_{\text{MPC}}(\bar{\mathbf{x}})$, then u_{MPC} is not persistently feasible in $\mathbb{X}_0$, which is not possible because we assume that u_{MPC} satisfies Theorem (2). If $\bar{\mathbf{u}} = u_{\text{NN}}(\bar{\mathbf{x}})$, then the following switch condition must hold: $f(\bar{\mathbf{x}}, \bar{\mathbf{u}}) \in \mathbb{X} \subseteq \mathbb{X}_0$. However, this is not possible since $\bar{\mathbf{u}}$ is assumed to be an escaping control, that is, $f(\bar{\mathbf{x}}, \bar{\mathbf{u}}) \notin \mathbb{X}_0$. If $\bar{\mathbf{u}} = u_{\text{LQR}}(\bar{\mathbf{x}})$, then there must be $\bar{\mathbf{x}} \in \mathcal{R}_{\text{LQR}}$. Since $\mathcal{R}_{\text{LQR}}$ is positively invariant, we must have $f(\bar{\mathbf{x}}, \bar{\mathbf{u}}) \in \mathcal{R}_{\text{LQR}} \subset \mathbb{X}_0$, which is not possible since $f(\bar{\mathbf{x}}, \bar{\mathbf{u}}) \notin \mathbb{X}_0$. Therefore, the system will never escape $\mathbb{X}_0$. $\square$

An illustration of how the alternating-authority MAMPC *could* steer a system to the equilibrium is provided in Figure 6.2b. Note that marginal stability may be an understatement in practice: a well-trained NN, combined with a modest alternation period i_d , could very well result in a closed-loop system that is asymptotically stable.

If analytical guarantee of asymptotic stability in the *entire* $\mathbb{X}_0$ is absolutely required, one can replace the admissible control set $\mathbb{X}_0$ with a time-varying set $\mathbb{S}[i]$, where

$$\mathcal{R}_{\text{LQR}} \subset \mathbb{S}[i+1] \subset \mathbb{S}[i] \subset \mathbb{X}_0, \quad \forall i = 1, \dots, n-1, \quad (6.13)$$

for some $n \in \mathbb{Z}_{>1}$ with $\mathbb{S}[0] = \mathbb{X}_0$ and $\mathbb{S}[n] = \mathcal{R}_{\text{LQR}}$. Geometrically, $\mathbb{S}$ can be viewed as a set that gets successively shrunk from $\mathbb{X}_0$ toward $\mathcal{R}_{\text{LQR}}$. Such design will invoke the fail-safe MPC if the systems is “stuck” outside of $\mathcal{R}_{\text{LQR}}$.

Lastly, similar to alternating-authority MAMPC, the closed-loop system of a way-point MAMPC is locally asymptotically stable in $\mathcal{R}_{\text{LQR}}$ and will eventually enter and stay within a level set that *describes* $\mathcal{D}_{\text{WP}}$, as stated in the following theorem.

Theorem 5 (Stability of Way-Point MAMPC). For every MPC problem of the form (6.2), there always exists a $u_{\text{MAMPC}}^{\text{WP}}$ of the form (6.11). Furthermore, the closed-loop system

$$\mathbf{x}[i+1] = f(\mathbf{x}[i], u_{\text{MAMPC}}^{\text{WP}}(\mathbf{x}[i])), \quad i \in \mathbb{Z}_{\geq 0}, \quad (6.14)$$

with $\mathbf{x}[0] = \mathbf{x}_0 \in \mathbb{X}_0$ is locally asymptotically stable in $\mathcal{R}_{\text{LQR}}$. Besides, there exists a time index $j \in \mathbb{Z}_{\geq 0}$ such that for all $i \geq j$

$$\mathbf{x}[i] \in \mathcal{D}_{\text{WP}}^+ := \{\mathbf{x} \in \mathbb{R}^n \mid V(\mathbf{x}) < \max_{q \in \partial \mathcal{D}_{\text{WP}}} V(\mathbf{x})\},$$

where $V : \mathbb{R}^n \rightarrow \mathbb{R}_+$ is a Lyapunov function of the closed-loop system (6.4) and $\partial \mathcal{D}_{\text{WP}}$ is the boundary of the compact set $\mathcal{D}_{\text{WP}}$ that by construction contains $\mathcal{R}_{\text{LQR}}$.

Proof. The first part of the proof on existence is identical to the one in Theorem 3.

To prove stability of way-point MAMPC, we partition $\mathbb{X}_0$ into five disjoint regions as follows

$$\mathbb{X}_0 = \mathcal{R}_{\text{LQR}} \cup \mathcal{R}_{\text{NN}}^i \cup \mathcal{R}_{\text{MPC}}^i \cup \mathcal{R}_{\text{NN}}^o \cup \mathcal{R}_{\text{MPC}}^o,$$

where $\mathcal{R}_{\text{NN}}^i$ is the set of states in or on the boundary of $\mathcal{D}_{\text{WP}}$ where the NN is invoked; $\mathcal{R}_{\text{NN}}^o$ is the set of states in $\mathbb{X}_0 \setminus \mathcal{D}_{\text{WP}}$ where the NN is invoked; $\mathcal{R}_{\text{MPC}}^i := \mathcal{D}_{\text{WP}} \setminus (\mathcal{R}_{\text{LQR}} \cup \mathcal{R}_{\text{NN}}^i)$; and $\mathcal{R}_{\text{MPC}}^o := (\mathbb{X}_0 \setminus \mathcal{D}_{\text{WP}}) \setminus \mathcal{R}_{\text{NN}}^o$.

For every state $\mathbf{x} \in \mathcal{R}_{\text{LQR}}$, the closed-loop system (6.14) is identical to closed-loop system (6.11), which is locally asymptotically stable, as shown in Theorem (3).

Next, we prove that the system will eventually enter and stay within $\mathcal{D}_{\text{WP}}^+$.

For every state $\mathbf{x}[0] \in \mathbb{X}_0 \setminus \mathcal{D}_{\text{WP}}$, either NN or MPC will steer the system into $\mathcal{D}_{\text{WP}}^+$ in finite time. If NN is ever invoked, the system (6.14) will be taken into $\mathcal{D}_{\text{WP}}$ in no more than N_{WP} steps. Otherwise, MPC will bring the system (6.14) into $\mathcal{D}_{\text{WP}}$ in finite time since the MPC is locally asymptotically stable in $\mathbb{X}_0$.

For every state $\mathbf{x}[0] \in \mathcal{D}_{\text{WP}}$, we prove by contradiction that there exists a $j \geq 0$ such that for all $i \geq j$, $\mathbf{x}[i] \in \mathcal{D}_{\text{WP}}^+$. Suppose there exists a $\mathbf{x}[0] = \bar{\mathbf{x}} \in \mathcal{D}_{\text{WP}}$ such that for every $k \geq 0$, there exists an $i \geq k$ such that $\mathbf{x}[i] \notin \mathcal{D}_{\text{WP}}^+$. Without loss of generality, define a control $\bar{\mathbf{u}}$ such that $\mathbf{x}[i-1] \in \mathcal{D}_{\text{WP}}^+$ but $\mathbf{x}[i] = f(\mathbf{x}[i-1], \bar{\mathbf{u}}) \notin \mathcal{D}_{\text{WP}}^+$.

This escaping control $\bar{\mathbf{u}}$ must be produced by either u_{MPC} , u_{NN} , or u_{LQR} , that is, $\bar{\mathbf{u}} = u_{\text{MPC}}(\mathbf{x}[i-1])$, $\bar{\mathbf{u}} = u_{\text{NN}}(\mathbf{x}[i-1])$, or $\bar{\mathbf{u}} = u_{\text{LQR}}(\mathbf{x}[i-1])$. If $\bar{\mathbf{u}} = u_{\text{MPC}}(\mathbf{x}[i-1])$, then the close-loop system (6.14) is equivalent to system (6.4). Because $\mathbf{x}[i] \notin \mathcal{D}_{\text{WP}}^+$, we have

$$V(\mathbf{x}[i]) = V(f(\mathbf{x}[i-1], u_{\text{MPC}}(\mathbf{x}[i-1]))) > V(\mathbf{x}[i-1]),$$

but this is not possible because V is a Lyapunov function of the system (6.4), i.e.,

$$V(f(\mathbf{x}[i-1], u_{\text{MPC}}(\mathbf{x}[i-1]))) \leq V(\mathbf{x}[i-1]).$$

If $\bar{\mathbf{u}} = u_{\text{NN}}(\mathbf{x}[i-1])$, then $f(\mathbf{x}[i-1], u_{\text{NN}}(\mathbf{x}[i-1])) \notin \mathcal{D}_{\text{WP}}$, which is not possible because invocation of NN implies that $f(\mathbf{x}[i-1], u_{\text{NN}}(\mathbf{x}[i-1])) \in \mathcal{D}_{\text{WP}}$. If $\bar{\mathbf{u}} = u_{\text{LQR}}(\mathbf{x}[i-1])$, then by assumption $\lim_{i \rightarrow \infty} \mathbf{x}[i]$ does not exist. However, this is impossible because $\lim_{i \rightarrow \infty} \mathbf{x}[i] = \mathbf{0}$ by the local asymptotic stability of the system (6.14) in $\mathcal{R}_{\text{LQR}}$. Consequently, the assumption must be false. $\square$

An illustration of how the way-point MAMPC *could* steer a system to the equilibrium is provided in Figure 6.2c. As in the alternating-authority MAMPC, to always achieve asymptotic stability, one can successively shrink $\mathcal{D}_{\text{WP}}$ until $\mathcal{D}_{\text{WP}} = \mathcal{R}_{\text{LQR}}$ in the same way as presented in equation (6.13).

6.4.2 Remark on Robustness

MAMPC may be *robustified* to account for model uncertainties using the idea of bounding error margin. We demonstrate such robustification procedure by applying it to the standard MAMPC.

Let the *true* system dynamics be f . We distinguish $\bar{f}$ as the *model* of the dynamical system. Define the model uncertainties of a controller u with initial condition $\mathbf{x}[i]$ at time step $i+k$ as

$$\Delta \mathbf{x}_u[k|i] := \mathbf{y}[k] - \bar{\mathbf{y}}[k],$$

where

$$\begin{aligned} \mathbf{y}[0] &= \mathbf{x}[i], & \mathbf{y}[k] &= f(\mathbf{y}[k-1], u(\mathbf{y}[k-1])), \\ \bar{\mathbf{y}}[0] &= \mathbf{x}[i], & \bar{\mathbf{y}}[k] &= \bar{f}(\bar{\mathbf{y}}[k-1], u(\bar{\mathbf{y}}[k-1])). \end{aligned}$$

When $\|\Delta \mathbf{x}_u[k|i]\| = \alpha > 0$, it is possible that $\mathbf{y}[k] \notin \mathcal{R}_{\text{LQR}}$ but $\bar{\mathbf{y}}[k] \in \mathcal{R}_{\text{LQR}}$, potentially leading to a failure of MAMPC. To address this problem, we define a set operator Ero_δ as follows

$$\text{Ero}_\delta(\mathcal{X}) := \{\mathbf{x} \in \mathcal{X} \mid \|\mathbf{x} - \mathbf{q}\| \geq \delta, \forall \mathbf{q} \in \partial \mathcal{X}\} \text{ for some } \delta > 0,$$

where $\mathcal{X}$ is an arbitrary set and $\partial\mathcal{X}$ denotes the boundary of the set $\mathcal{X}$. We claim that for $\|\Delta\mathbf{x}_u[k|i]\| = \alpha > 0$, if $\bar{\mathbf{y}}[k] \in \text{Ero}_\delta(\mathcal{R}_{\text{LQR}})$ with $\delta \geq \alpha$, then there must be $\mathbf{y}[k] \in \mathcal{R}_{\text{LQR}}$. Proof of the claim is a direct application of the triangle inequality of norm. We can therefore incorporate the set operator Ero_δ to enhance the robustness of MAMPC with respect to model uncertainties.

We can apply the above procedure to robustify the standard MAMPC as follows

$$u_{\text{MAMPC}}^\delta(\mathbf{x}) := \begin{cases} u_{\text{LQR}}(\mathbf{x}), & \mathbf{x} \in \mathcal{R}_{\text{LQR}}, \\ \hline u_{\text{NN}}(\mathbf{x}), & \begin{array}{l} \exists i = 1, \dots, N_{\text{LQR}}, \quad \bar{\mathbf{y}}[i] \in \mathcal{R}_{\text{LQR}}^\delta, \\ \forall j = 0, \dots, j, \quad (\bar{\mathbf{y}}[j], u_{\text{NN}}(\bar{\mathbf{y}}[j])) \in \mathbb{A}^\delta \end{array} \\ \hline u_{\text{MPC}}(\mathbf{x}), & \text{otherwise,} \end{cases}$$

where $\mathcal{R}_{\text{LQR}}^\delta := \text{Ero}_\delta(\mathcal{R}_{\text{LQR}})$ and $\mathbb{A}^\delta := \text{Ero}_\delta(\mathbb{X}) \times \mathbb{U}$, for some $\delta > 0$. Here, δ can be viewed geometrically as a margin of robustness.

The stability of the above robustified MAMPC is stated in the following theorem.

Theorem 6 (Robustification of Standard MAMPC). Suppose the model uncertainties are upper bounded by some $\alpha > 0$, that is,

$$\|\Delta\mathbf{x}_u[k|i]\| \leq \alpha, \quad \forall i \in \mathbb{Z}_{\geq 0}, \forall k = 0, \dots, N_{\text{LQR}}.$$

Then the following closed-loop system

$$\mathbf{x}[i+1] = f(\mathbf{x}[i], u_{\text{MAMPC}}^\alpha(\mathbf{x}[i])), \quad i \in \mathbb{Z}_{\geq 0},$$

with $\mathbf{x}[0] = \mathbf{x}_0 \in \mathbb{X}_0$ is locally asymptotically stable in $\mathbb{X}_0$.

Proof. The proof is identical to that of Theorem 3. Note that it possible that $\mathcal{R}_{\text{LQR}}^\delta = \emptyset$. In this case, the hybrid control is virtually equivalent to a dual-mode MPC-LQR controller. $\square$

Similar to the above example, the robustification procedure can also be applied to enhance the alternating-authority MAMPC and way-point MAMPC.

6.4.3 Remark on Necessity of Fail-Safe MPC

The hybrid control scheme can be clearly more efficient if we can skip the forward verification and remove the fail-safe MPC mode. In light of this observation, we describe a condition, where the fail-safe MPC mode will *never* be invoked and thus can be safely removed from the hybrid control scheme.

Define the following “fail-free” MAMPC

$$u_{\text{MAMPC}}^+(\mathbf{x}) := \begin{cases} u_{\text{LQR}}(\mathbf{x}), & \mathbf{x} \in \mathcal{R}_{\text{LQR}}, \\ u_{\text{NN}}(\mathbf{x}) & \mathbf{x} \in \mathbb{X}_0 \setminus \mathcal{R}_{\text{LQR}}. \end{cases}$$

Let $L : \mathbb{X} \rightarrow \mathbb{R}_+$ be an augmented cumulative cost function of NN, that is,

$$L(\mathbf{x}) := \sum_{k=0}^{N-1} c(\mathbf{y}[k], u_{\text{NN}}(\mathbf{y}[k])) + c_f(\mathbf{y}[N]),$$

with

$$\mathbf{y}[0] = \mathbf{x}, \quad \mathbf{y}[k] = f(\mathbf{y}[k-1], u_{\text{NN}}(\mathbf{y}[k-1])).$$

From Theorem 13.1 in [15], we are ready to state the condition in the following theorem.

Theorem 7 (Fail-Free MAMPC). If there exists some nonnegative function $\gamma : \mathbb{X}_0 \rightarrow \mathbb{R}_+$ such that

$$\begin{aligned} (\mathbf{x}, u_{\text{NN}}(\mathbf{x})) &\in \mathbb{A}, \quad \forall \mathbf{x} \in \mathbb{X}_0, \\ L(\mathbf{x}) &\leq J^*(\mathbf{x}) + \gamma(\mathbf{x}), \quad \forall \mathbf{x} \in \mathbb{X}_0, \\ \gamma(\mathbf{x}) - c(\mathbf{x}, \mathbf{0}) &< 0, \quad \forall \mathbf{x} \in \mathbb{X}_0 \setminus \mathbf{0}, \end{aligned} \tag{6.15}$$

then the closed-loop system

$$\mathbf{x}[i+1] = f(\mathbf{x}[i], u_{\text{MAMPC}}^+(\mathbf{x}[i])), \quad i \in \mathbb{Z}_{\geq 0},$$

with $\mathbf{x}[0] = \mathbf{x}_0 \in \mathbb{X}_0$ is locally asymptotically stable in $\mathbb{X}_0$ without violating any constraints in the original MPC problem (6.2).

Proof. The proof of the theorem is a direct application of Theorem 13.1 in [15]. $\square$

Note that the inequality conditions in (6.15) are typically verified through sampling and interpolation [15], which makes this approach only suitable for relatively simple, moderately sized problems. Besides, the above theorem is only a sufficient condition and other sufficient conditions are possible too. For example, see [101, 20].

Last but not least, we emphasize that *all* the analysis above does not impose any condition on the approximation error of the NN. So long as the NN has the right input and output dimensions, the corresponding MAMPC will be closed-loop stable, even if the weights of the NN are *random*. In the worst case, MAMPC reduces to a hybrid controller consisting of just the MPC and the LQR, where the NN mode will never be invoked. This reduced hybrid controller is virtually identical to the one proposed in [78], despite of having slightly different running time performance.

6.5 Numerical Experiments

To evaluate the running time performance of MAMPC, we conduct four numerical experiments on controlling a pendulum, a triple pendulum, a bicopter, and a quadcopter. The models are selected for their relevance in industrial robotic applications: Pendulum models are the building blocks for robot arm manipulation, while copter models are important model classes in unmanned aerial vehicle control. Performance and efficiency of MAMPC are

evaluated through comparison with the corresponding baseline implicit MPC. In addition to the four experiments, a short numerical simulation is appended to the end of the section to demonstrate the notion of robustness margin.

All computations were conducted on an Intel Core i7-1065G7 CPU machine. Implementation is done via MATLAB. The following paragraphs highlight the key parameters and results of the four experiments. For implementation details, please refer to the source code at [131].

6.5.1 Pendulum

The first model is a single-arm pendulum as shown in Figure 6.3. The goal is to maintain the pendulum at the position of the highest potential energy as marked by the dashed line.

The state is defined as $\mathbf{x} := [\theta \ \dot{\theta}]^\top \in [-\pi, \pi) \times \mathbb{R}$, where θ is the angular displacement from the inverted position. The input is a scalar $u \in \mathbb{R}$, which is a bounded torque applied at the joint. The control objective is to steer the pendulum to the origin through the limited torque actuator $u \in [-0.05, 0.05]$ N · m at the joint.

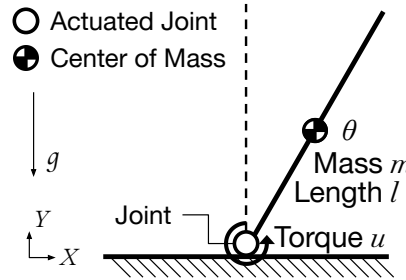


Figure 6.3: Diagram of a pendulum model. A torque between $[-0.05, 0.05]$ N · m is being applied to the hinge to keep the pendulum at its inverted position.

The MPC design highlights a sampling interval of 0.1 s, a planning horizon $N = 5$, and three optimization variables per step, resulting in a linearly-constrained quadratic programming (LCQP) of 0.5 s prediction horizon and 15 variables. Plant nonlinearity is handled through linearizing around the equilibrium, leading to a linear time-invariant system.

We apply a standard MAMPC to control the pendulum with

$$\mathcal{R}_{\text{LQR}} = \{\mathbf{x} \in \mathbb{R}^2 \mid \|\mathbf{x}\|_2 \leq 0.5\}, \quad N_{\text{LQR}} = 5.$$

The NN is a two-layer, 20-neuron multilayer perceptron (MLP) trained through supervised learning with data uniformly randomly sampled from $\theta \in [\pi, \pi], \dot{\theta} \in [-1, 1]$. Details of the NN architecture can be found in Table 6.1 in the Appendix.

Performance and efficiency of MAMPC, at various stages of training, is illustrated in Figure 6.4. The total and per-step running times of MAMPC are summarized in the first columns of Table 6.2 and Table 6.3, respectively.

Please note that if even though the running times of implicit MPC and NN appear constant in Figure 6.4, they are in fact time-varying if we zoom in. This applies to the other three systems below.

6.5.2 Triple Pendulum

The second model is a triple pendulum as shown in Figure 6.5, which extends the above pendulum model to a more complex use case. The goal is to maintain the triple pendulum at the position of the highest potential energy.

The state is defined as $\mathbf{x} := [\theta_1 \ \dot{\theta}_1 \ \theta_2 \ \dot{\theta}_2 \ \theta_3 \ \dot{\theta}_3]^\top \in ([-\pi, \pi) \times \mathbb{R})^3$, where $\theta_1, \theta_2, \theta_3$ are the angular displacements of the three links respectively. The input is defined as $\mathbf{u} := [u_1 \ u_2 \ u_3]^\top \in \mathbb{R}^3$, which is the applied torques at the three joints respectively. The control objective is to steer the triple pendulum to the origin by applying three limited torque inputs $u_1, u_2, u_3 \in [-1, 1]$ N · m at the three joints, respectively.

The MPC design highlights a sampling interval of 0.1 s, a planning horizon $N = 5$, and nine optimization variables per step, resulting in a LCQP of 0.5 s prediction horizon and 45 variables. Linearization is applied as before to handle nonlinearity.

Because triple pendulum is a chaotic system, we apply an alternating-authority MAMPC instead of standard MAMPC with

$$\mathcal{R}_{\text{LQR}} = \{\mathbf{x} \in \mathbb{R}^6 \mid \|\mathbf{x}\|_2 \leq 0.4\}, \quad N_{\text{LQR}} = 5, \quad i_d = 2.$$

The NN features a three-layer, 50-neuron MLP trained through supervised learning with data uniformly randomly sampled from

$$\theta_1, \theta_2, \theta_3 \in [-\pi/6, \pi/6], \quad \dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3 \in [-1, 1].$$

Details of the NN architecture can be found in Table 6.1 in the Appendix.

Performance and efficiency of MAMPC, at various stages of training, is illustrated in Figure 6.6. The total and per-step running times of MAMPC are summarized in the second columns of Table 6.2 and Table 6.3, respectively.

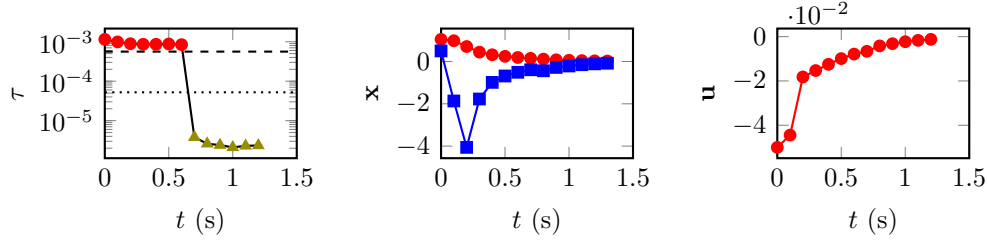
Lastly, note that running times of implicit MPC is represented as constant lines when in reality it is decreasing with time. For readability, we choose to use the median of the varying running times to represent it as they are orders of magnitude larger than NN and MAMPC and look like constant in log scale.

6.5.3 Bicopter

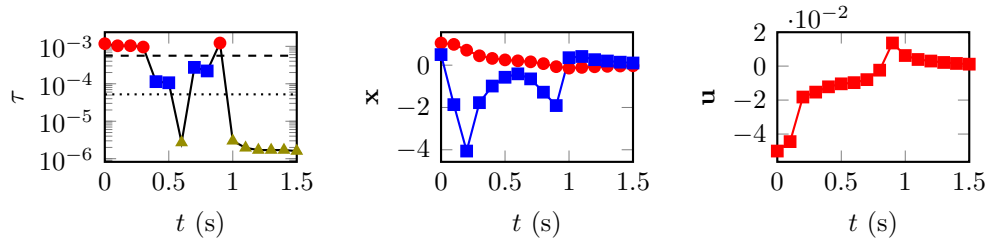
The third model is a bicopter as shown in Figure 6.7. The goal is to hover the bicopter in air.

The state is defined as $\mathbf{x} := [x \ \dot{x} \ y \ \dot{y} \ \theta \ \dot{\theta}]^\top \in \mathbb{R}^4 \times ([-\pi, \pi) \times \mathbb{R})$, where x, y are horizontal and vertical translations and θ is the angle of tilting. The input is defined as $\mathbf{u} := [u_1 \ u_2]^\top \in \mathbb{R}^2$, where u_1, u_2 are the thrust exerted by the left and right propellers,

Initialized: $|\mathcal{D}| = 0$



Being Trained: $|\mathcal{D}| = 150$



Converged: $|\mathcal{D}| = 500$

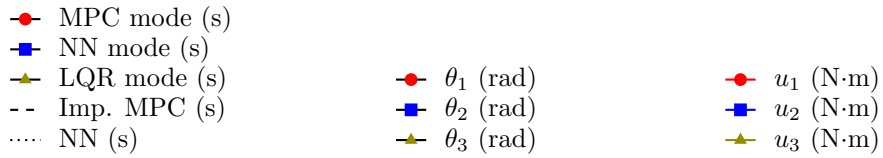
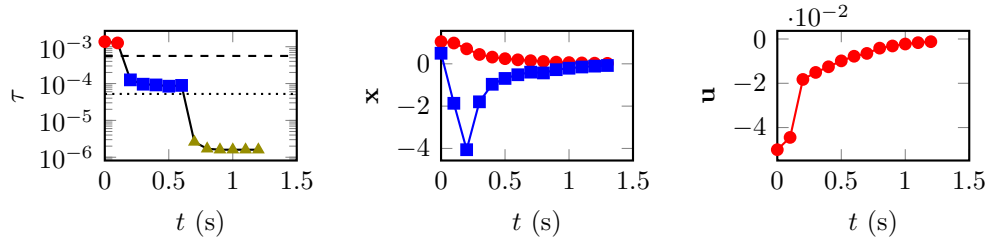


Figure 6.4: Application of standard MAMPC to the pendulum problem. τ is running time. $\mathbf{x}$ is selected states and $\mathbf{u}$ is selected inputs. t is simulation time. At initialization, the NN mode fails forward verification and thus never get invoked. As the NN is being trained, its likelihood of passing forward verification increases. At last, a well trained NN takes over most of MPC steps, resulting in a more efficient operation.

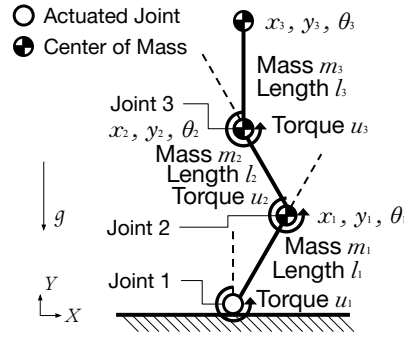


Figure 6.5: Diagram of a triple pendulum model. Note that the centers of mass of the first and second torques coincide with the second and third joints. A torque between $[-1, 1]$ N·m is being applied to each joint to keep the triple pendulum at its inverted position.

respectively. The control objective is to steer the bicopter to the origin by applying limited thrusts $u_1, u_2 \in [0.1, 9.1572]$ N at the two propellers.

The MPC design highlights a sampling interval of 0.1 s, a planning horizon $N = 20$, and eight optimization variables per step, resulting in a LCQP of 2.0 s prediction horizon and 160 variables. To handle plant nonlinearity, we linearize the system around the equilibrium.

We apply a standard MAMPC to control the bicopter with

$$\mathcal{R}_{\text{LQR}} = \{\mathbf{x} \in \mathbb{R}^6 \mid \|\mathbf{x}\|_2 \leq 0.5\}, \quad N_{\text{LQR}} = 10.$$

The NN is a three-layer, 50-neuron MLP trained through supervised learning with data uniformly randomly sampled from

$$x, y, \theta \in [-\pi/2, \pi/2], \quad \dot{x}, \dot{y}, \dot{\theta} \in [-1, 1].$$

Details of the NN architecture can be found in Table 6.1 in the Appendix.

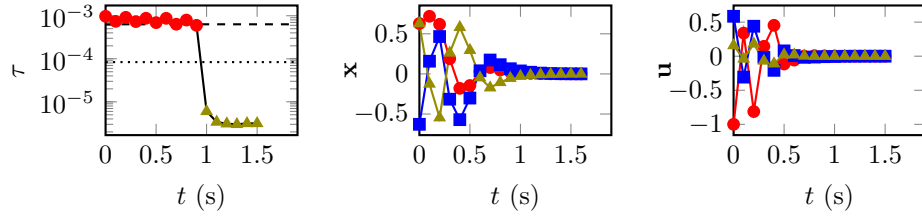
Performance and efficiency of MAMPC, at various stages of training, is illustrated in Figure 6.8. The total and per-step running times of MAMPC are summarized in the third columns of Table 6.2 and Table 6.3, respectively.

6.5.4 Quadcopter

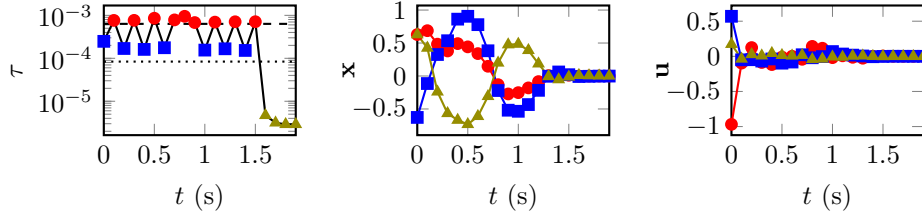
As an extension to the ideal bicopter model, the last model is a quadcopter as shown in Figure 6.9 [3]. The goal is also to hover the copter in air but with more realistic system dynamics.

The state is defined as $\mathbf{x} := [x \ \dot{x} \ y \ \dot{y} \ z \ \dot{z} \ \phi \ \dot{\phi} \ \theta \ \dot{\theta} \ \psi \ \dot{\psi}]^\top \in \mathbb{R}^6 \times ([-\pi, \pi) \times \mathbb{R})^3$, where x, y, z represent translations and ϕ, θ, ψ represent rotations corresponding to the three ZYX Euler angles roll, pitch, and yaw. The input is $\mathbf{u} = [u_1 \ u_2 \ u_3 \ u_4]^\top \in \mathbb{R}^4$, where u_i is the rotational speeds of the i^{th} propeller for $i = 1, 2, 3, 4$. The control objective is to steer the

Initialized: $|\mathcal{D}| = 0$



Being Trained: $|\mathcal{D}| = 20000$



Converged: $|\mathcal{D}| = 30000$

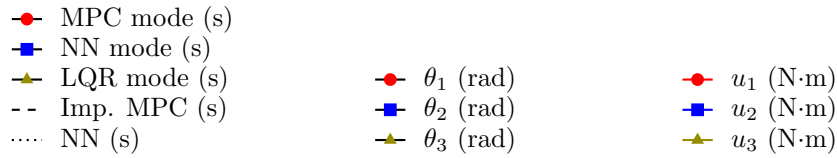
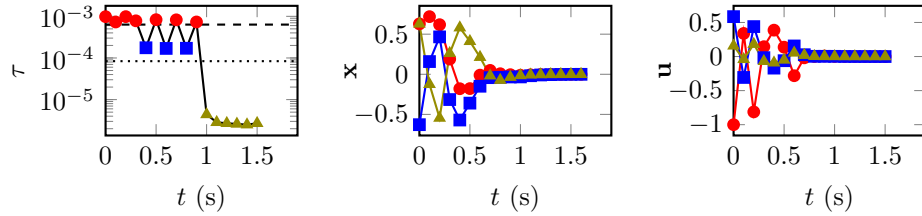


Figure 6.6: Application of alternating-authority MAMPC to the triple pendulum problem. τ is running time. $\mathbf{x}$ is selected states and $\mathbf{u}$ is selected inputs. t is simulation time. At initialization, the NN mode fails forward verification and thus never get invoked. As the NN is being trained, its likelihood of passing forward verification increases. At last, a well trained NN takes over most of MPC steps, resulting in a more efficient operation.

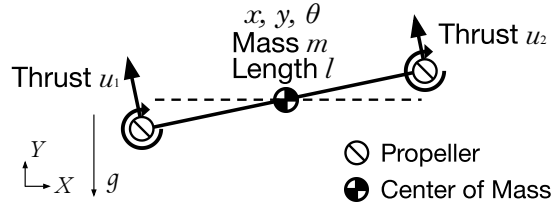


Figure 6.7: Diagram of a bicopter model. Each of the two propellers is capable of generating a thrust between 0.1 N and 9.1572 N to keep the copter hover in air. The thrust limits are chosen such that hovering thrusts are approximately in the center of the thrust limit.

quadcopter to the origin by applying bounded rotations $u_1, u_2, u_3, u_4 \in [0, 313.96]$ in $\text{rad} \cdot \text{s}^{-1}$ at the four propellers.

The MPC design highlights a sampling interval of 0.1 s, a planning horizon $N = 20$, and 16 optimization variables per step, resulting in a LCQP of 2.0 s prediction horizon and 320 variables. Nonlinearity is handled by linearization around the equilibrium.

Because the closed-loop quadcopter dynamics with MPC takes many steps to converge, we choose a way-point MAMPC with

$$\begin{aligned} \mathcal{D}_{\text{WP}} &= \{\mathbf{x} \in \mathbb{R}^{12} \mid \|\mathbf{x}\|_2 \leq 2\}, \quad N_{\text{WP}} = 10, \\ \mathcal{R}_{\text{LQR}} &= \{\mathbf{x} \in \mathbb{R}^{12} \mid \|\mathbf{x}\|_2 \leq 0.5\}, \quad N_{\text{LQR}} = 10. \end{aligned}$$

The NN features a four-layer, 60-neuron MLP trained through supervised learning with data uniformly randomly sampled from

$$\begin{aligned} x, y, z &\in [-0.5, 0.5], \quad \dot{x}, \dot{y}, \dot{z} \in [-0.1, 0.1], \\ \phi, \theta &\in [-\pi/6, \pi/6], \psi \in [-\pi/4, \pi/4], \dot{\phi}, \dot{\theta}, \dot{\psi} \in [-0.1, 0.1]. \end{aligned}$$

Details of the NN architecture can be found in Table 6.1 in the Appendix.

Performance and efficiency of MAMPC, at various stages of training, is illustrated in Figure 6.10. The total and per-step running times of MAMPC are summarized in the last columns of Table 6.2 and Table 6.3, respectively.

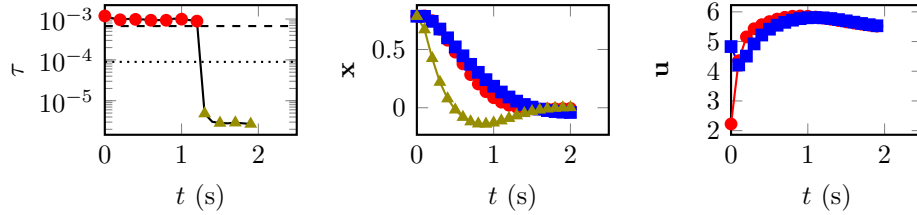
6.5.5 Robustness Margin

Lastly, we present how $\mathcal{R}_{\text{LQR}}$ can be shrunk to enhance the controller robustness with respect to model uncertainty.

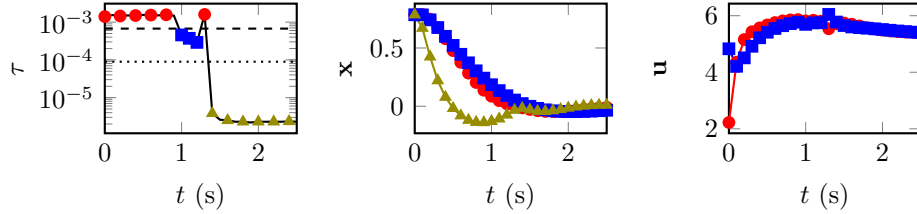
Consider the pendulum model presented above. Denote the pendulum system model as $\dot{\mathbf{x}} = \bar{f}(\mathbf{x})$. Suppose there is an external disturbance which alters the system into $\dot{\mathbf{x}} = f(\mathbf{x}) := \bar{f}(\mathbf{x}) + \mathbf{1} \cdot b$ with $b > 0$ being a constant bias term. Let the true system trajectory be $\mathbf{x}(t)$ and the trajectory predicted by $\bar{f}(\cdot)$ be $\bar{\mathbf{x}}(t)$.

Define $\mathcal{R}_{\text{LQR}} = \{\mathbf{x} \in \mathbb{R}^2 \mid \|\mathbf{x}\|_2 \leq 0.35\}$. Simulating both systems from the same initial condition $\mathbf{x}_0 = [0.65 \quad -4]^\top$ and $b = 0.2$, we obtain the result shown in Figure 6.11.

Initialized: $|\mathcal{D}| = 0$



Being Trained: $|\mathcal{D}| = 50$



Converged: $|\mathcal{D}| = 350$

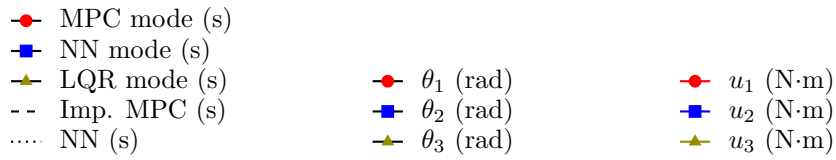
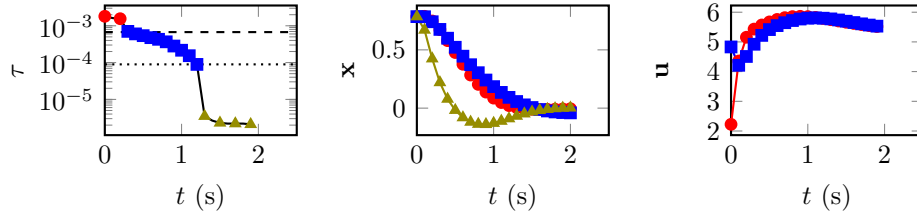


Figure 6.8: Application of standard MAMPC to the bicopter problem. τ is running time. $\mathbf{x}$ is selected states and $\mathbf{u}$ is selected inputs. t is simulation time. At initialization, the NN mode fails forward verification and thus never get invoked. As the NN is being trained, its likelihood of passing forward verification increases. At last, a well trained NN takes over most of MPC steps, resulting in a more efficient operation.

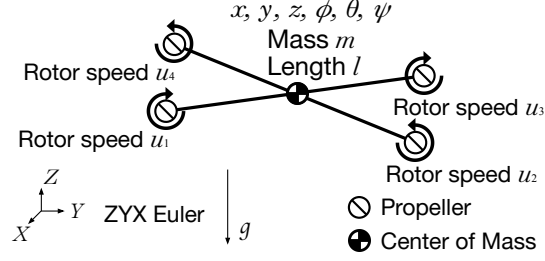
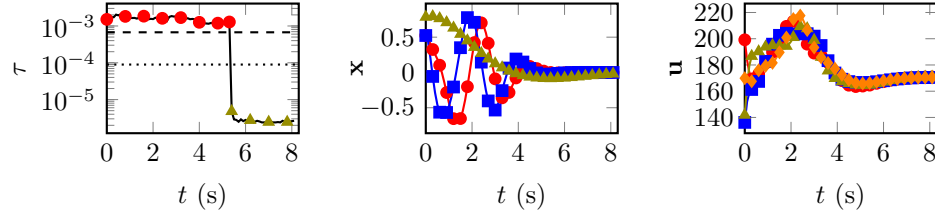


Figure 6.9: Diagram of a quadcopter model. Each of the four propellers is capable of rotating at speed between $0 \text{ rad} \cdot \text{s}^{-1}$ and $313.96 \text{ rad} \cdot \text{s}^{-1}$ to keep the copter hover in air. The rotor limits are chosen such that hovering rotor rotations are approximately in the center of the rotation range.

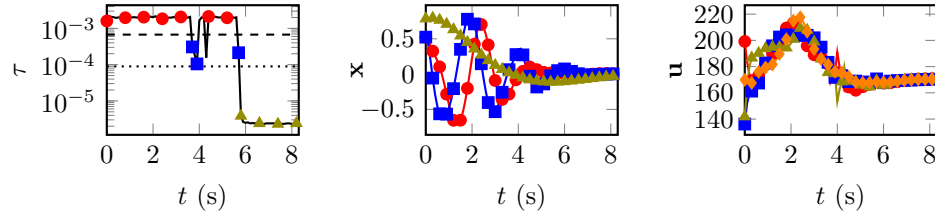
Table 6.1: Neural Networks Architecture

	Layer	Dimension	Activation
Pendulum	Input	$\mathbf{x} \in \mathbb{R}^2$	
	Hidden layer 1	10 neurons + bias	ReLU
	Hidden layer 2	10 neurons + bias	ReLU
	Output	$u \in \mathbb{R}$	Linear
Tri. Pend.	Input	$\mathbf{x} \in \mathbb{R}^6$	
	Hidden layer 1	20 neurons + bias	ReLU
	Hidden layer 2	10 neurons + bias	ReLU
	Hidden layer 3	20 neurons + bias	ReLU
	Output	$\mathbf{u} \in \mathbb{R}^3$	Linear
Bicopter	Input	$\mathbf{x} \in \mathbb{R}^6$	
	Hidden layer 1	20 neurons + bias	ReLU
	Hidden layer 2	10 neurons + bias	ReLU
	Hidden layer 3	20 neurons + bias	ReLU
	Output	$\mathbf{u} \in \mathbb{R}^2$	Linear
Quadcopter	Input	$\mathbf{x} \in \mathbb{R}^{12}$	
	Hidden layer 1	20 neurons + bias	ReLU
	Hidden layer 2	10 neurons + bias	ReLU
	Hidden layer 3	10 neurons + bias	ReLU
	Hidden layer 4	20 neurons + bias	ReLU
	Output	$\mathbf{u} \in \mathbb{R}^4$	Linear

Initialized: $|\mathcal{D}| = 0$



Being Trained: $|\mathcal{D}| = 400$



Converged: $|\mathcal{D}| = 800$

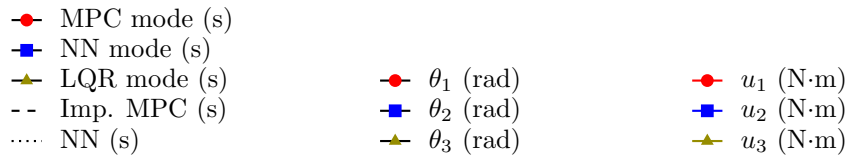
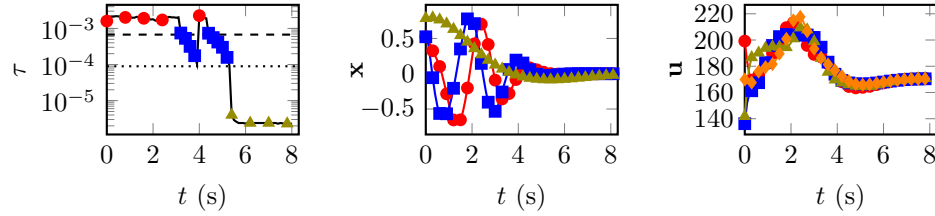


Figure 6.10: Application of way-point MAMPC to the quadcopter problem. τ is running time. $\mathbf{x}$ is selected states and $\mathbf{u}$ is selected inputs. t is simulation time. At initialization, the NN mode fails forward verification and thus never get invoked. As the NN is being trained, its likelihood of passing forward verification increases. At last, a well trained NN takes over most of MPC steps, resulting in a more efficient operation.

From the simulation result, we learn that the true system does not enter $\mathcal{R}_{\text{LQR}}$ until the modeled system enters $\mathcal{R}_{\text{LQR}}^\delta$ with $\delta = 0.2$. Hence, δ can be considered as a margin of robustness. That is, if we verify $\bar{\mathbf{x}} \in \mathcal{R}_{\text{LQR}}^\delta$, we can ensure that $\mathbf{x} \in \mathcal{R}_{\text{LQR}}$ despite the lack of the bias term $\mathbf{1} \cdot b$ in $\bar{f}(\cdot)$.

Table 6.2: Summary of Total Running Time in Millisecond

	Pendulum	Tri. Pend.	Bicopter	Quadcopter
Imp. MPC	10.17 ± 3.41	13.03 ± 3.44	15.58 ± 3.12	92.66 ± 11.15
MAMPC (initialized)	7.37 ± 2.40	8.54 ± 1.36	14.69 ± 3.97	93.55 ± 11.67
MAMPC (being train.)	7.21 ± 2.11	9.00 ± 1.69	20.04 ± 4.74	117.64 ± 13.74
MAMPC (converged)	3.63 ± 0.94	7.11 ± 1.63	9.66 ± 1.85	87.84 ± 16.69

Table 6.3: Summary of Per-Step Running Time in Mircosecond

	Pendulum	Tri. Pend.	Bicopter	Quadcopter
Imp. MPC	686.39 ± 248.03	695.79 ± 178.03	743.84 ± 152.67	1032.76 ± 89.52
MAMPC-MPC*	1529.29 ± 411.57	897.02 ± 226.43	1777.19 ± 404.51	2100.90 ± 401.96
MAMPC-NN*	105.23 ± 33.60	187.53 ± 51.62	418.31 ± 90.21	420.92 ± 95.02
MAMPC-LQR*	1.86 ± 0.46	3.11 ± 0.92	2.42 ± 0.55	2.65 ± 0.67
NN [†]	59.85 ± 21.86	79.86 ± 24.20	98.60 ± 24.88	56.25 ± 23.89

* MAMPC-MPC, MAMPC-NN, and MAMPC-LQR are the respective modes of the MAMPC.

† NN is the NN mode of MAMPC *without* forward verification.

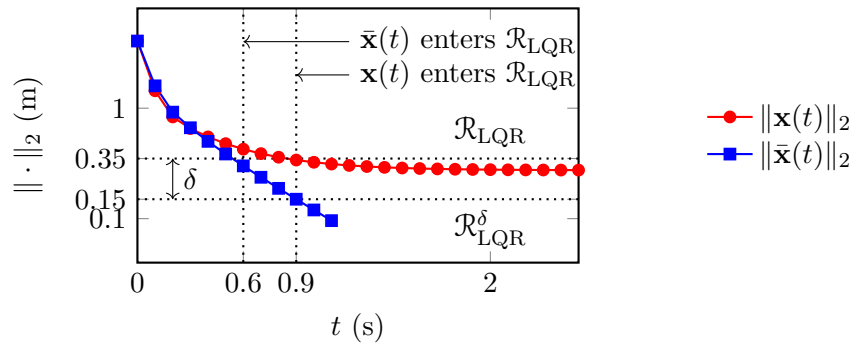


Figure 6.11: Illustration of robustness margin. The *true* system $f(\cdot)$ enters $\mathcal{R}_{\text{LQR}}$ when the *modeled* system $\bar{f}(\cdot)$ enters $\mathcal{R}_{\text{LQR}}^\delta$. Here, δ maybe viewed as a notion of robustness margin.

6.6 Discussion

As shown in Figure 6.4, Figure 6.6, Figure 6.8, and Figure 6.10 and in Table 6.2, the MAMPC gains efficiency over time through learning on the trajectory data generated by the implicit MPC policy. Upon initialization, the MAMPC policy, despite of not being the most efficient, is immediately functional. After learning on a few samples, the NN mode behaves closer to the MPC mode but still lacks in accuracy. This period of learning sometimes results an temporary degradation in control efficacy and an temporary increase in running time. In particular, see Figure 6.4 and Figure 6.6. After training on sufficient number of samples, the NN mode eventually converges to and takes over the implicit MPC. Consequently, the overall control efficacy is recovered and computational efficiency improved.

After the NN mode converges, the resulting MAMPC is shown to have superior amortized running time but an inferior worst-case running time. As the shown in the first four rows of Table 6.3, the LQR mode and NN mode of MAMPC is faster than implicit MPC, while the MPC mode of MAMPC is slower than the implicit method due to overhead from computing the switch conditions. Therefore, in the worst case, the MAMPC will take more time to stabilize the plant than the implicit MPC. However, if the MAMPC spends sufficient number of steps in the NN and LQR modes, it will still has faster average running time, which is the case observed in the four numerical experiments.

If the target system is capable of parallel computation, it is possible the reduce the worst-time running time down to the baseline running time of the implicit MPC. For example, a possible parallelization strategy for standard MAMPC is as follows: 1) compute the three control modes of MAMPC in three *parallel* threads, 2) among those that pass their corresponding switch conditions, execute whichever that finishes first, and 3) reset and repeat for next control step. Ignoring communication overhead, one can show that the worst-case running time is roughly identical to that of the implicit MPC. Note that under this setup the MAMPC scheme is *at worst* equivalent to the dual-mode MPC in [78].

Of course, it should be noted that there is no guarantee that the resulting MAMPC will be more efficient than the implicit MPC, because there is no guarantee that the NN will converge to an accurate surrogate policy. Comparing to methods like [137], where the implicit MPC controller is completely replaced by a NN controller, we trade a lack of deterministic guarantee on stability for a lack of deterministic guarantee on running time improvement. We believe that such a trade-off is often desirable in practice: in industrial applications, safety almost always precedes efficiency.

Beyond the direct implications of the experiments, we make following remarks on set design, forward verification, additional running time optimizations, and limitations.

6.6.0.1 Set Design

Identification and design of feasibility set $\mathbb{X}_0$ and attraction set $\mathcal{R}_{\text{LQR}}$ is very challenging. One one hand, it is easier to just approximate them by conservative set estimates for robustness measures. For example, one can replace $\mathbb{X}_0$ and $\mathcal{R}_{\text{LQR}}$ with $\mathbb{X}_0^-$ and $\mathcal{R}_{\text{LQR}}^-$, where $\mathbb{X}_0^- \subset \mathbb{X}_0$

and $\mathcal{R}_{\text{LQR}}^- \subset \mathcal{R}_{\text{LQR}}$. On the other hand, however, to guarantee computational efficiency, one should make sets $\mathcal{R}_{\text{LQR}}, \mathcal{R}_{\text{NN}}$ as large as possible. This way, faster modes of MAMPC get invoked more frequently than the slow, default mode of MAMPC, leading to a more efficient amortized running time performance. Conservative set design makes the system more robust to model uncertainties but also slower in running time. Balancing the trade-off between latency and robustness is therefore a design process.

6.6.0.2 Forward verification

The horizon used in forward verification immediately affects the maximum possible running time of the MAMPC. A valid verification horizon should not make the running time of the MPC mode longer than the maximum allowable computational latency. Moreover, an additional rule of thumb is to choose a verification horizon such that the maximum possible running time of NN mode is on par with the running time of the implicit MPC.

The simulation method used for forward verification also directly impacts the worst-case latency of the MAMPC. For continuous plants, we recommend to use forward Euler method with a sampling time that is as large as possible and to use a conservative set design to absorb integration errors. Like designing sets, choosing the right sampling time is a balancing act between latency and robustness.

6.6.0.3 Additional Optimizations

For small to moderately sized problems, we may consider developing a fail-free MAMPC as specified in Theorem 7, since it will further speed up computation. Fail-free MAMPC prevents controller from ever having to default to fail-safe mode and therefore skips the routine computation of forward verification in NN mode. It significantly improves the amortized and worst-case running time.

Another possibility for reducing running time is to pre-compute the results of forward verification for the entire state space *offline* and look up the results *online*. This look-up table approach is in spirit similar to explicit MPC. The complexity of this approach also grows exponentially with state dimension, so it suffers the similar scalability issue like explicit MPC does.

Of course, a third possibility for reducing running time is to speed up computation of the trained neural network through techniques such as quantization, pruning, or model distillation.

It is sometimes useful to further improve the performance of the NN mode via unsupervised reinforcement learning. This is usually the case when the default implicit MPC solver is sub-optimal, typically due to lack of convexity in the objective or constraints or due to large problem dimension induced by a long prediction horizon.

Finally, one could leverage warm-started MPC with our method for enhanced performance. Warm-started MPC methods like [21] aim to use a NN to speed up the *solving* of an implicit MPC, while MAMPC seeks to *replace* the implicit MPC with a NN or a LQR.

Consequently, one could directly combine the two techniques together by substituting the standard implicit MPC in a MAMPC with a warm-started implicit MPC.

Chapter 7

Devices and Experiments for Multi-Agent Research

An integral part of MARS research is empirical validation and hardware demonstration. Given that the ultimate aim of any multi-agent planning and control methodology is deployment on physical devices, both hardware devices and experimental designs are essential validation steps to truly assess the performance of any new techniques. Within the field of MARS research, several robotic devices are commonly utilized, including ground mobile robots, quadcopters, automated vehicles, biped and quadruped robots, and manipulator arms. Typical experimental setups involve navigation in controlled 2D or 3D environments, driving on real roads, traversing uneven natural terrains, and operations within factory settings. In this chapter, we focus our discussion on the devices and experiments implemented in this dissertation. We categorize our discussion into two parts: actuation and estimation. In the first half of the chapter, we detail the design of two actuation devices and relevant experiment setups: unicycle robots and automated vehicles. In the second half of the chapter, we explain the design of two estimation devices and relevant deployment scenarios: indoor localization systems and aerial video collection.

7.1 Actuation Devices and Experiments

In this section, we discuss the two primary types of actuation devices utilized in this thesis study, along with the corresponding experimental considerations developed. The first device is a unicycle robot, which is well-suited for preliminary indoor testing, while the second is an automated vehicle, appropriate for conducting real-world road tests. We present the basic mathematical models for each actuation device, the hardware aspects necessary for construction, and the typical experimental designs associated with these devices.

7.1.1 Unicycle Robot

A unicycle robot is a type of ground mobile robot primarily driven by two wheels at differential speeds. It typically features a three-wheeled design, as shown in Figure 7.1, with two active rear wheels providing differential drive and one passive front wheel for stability. In some designs, the passive front wheel is omitted, which complicates control as the remaining two wheels must manage both propulsion and balance. In our discussion, we focus on the simpler three-wheel design.

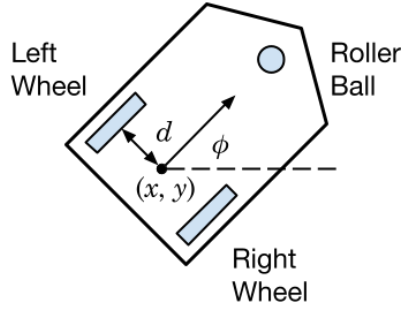


Figure 7.1: A schematic diagram of a typical unicycle robot.

The dynamics model of a unicycle robot is given by:

$$\mathbf{x} := \begin{bmatrix} \phi \\ x \\ y \\ \theta_L \\ \theta_R \end{bmatrix}, \quad \mathbf{u} := \begin{bmatrix} u_L \\ u_R \end{bmatrix}, \quad (7.1)$$

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) := \begin{bmatrix} -\frac{r}{2d} & \frac{r}{2d} \\ \frac{r}{2} \cos \phi & \frac{r}{2} \cos \phi \\ \frac{r}{2} \sin \phi & \frac{r}{2} \sin \phi \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_L \\ u_R \end{bmatrix},$$

where r is the radius of the robot's wheels, $2d$ is the wheelbase, ϕ is the robot's orientation, x and y are the robot's 2D coordinates, θ_L and θ_R are the rolling angles of the left and right wheels, respectively, and u_L and u_R are the rotation speeds of the left and right wheels.

Controlling a unicycle robot, given its simple dynamics in (7.1), is straightforward. Rotating both wheels at the same speed produces linear acceleration along the robot's direction, while a differential in speed creates a moment along the z -axis, altering the robot's orientation.

Due to their ease of control, unicycle robots are popular in robotics research for prototyping and validating trajectory planning algorithms. There are many commercially available educational unicycle robots, ranging from the inexpensive, under \$100 per unit, Makeblock mBot to the more costly, under \$500 per unit, Pololu 3pi series and TurtleBot series, which are above \$1000 per unit. Common features of these robots include an onboard computer, programmable microcontroller, camera, distance sensor, and an interface for wheel control.

Given that multi-agent hardware experiments often involve numerous robots, the cost of each unit is crucial, especially when budget constraints are tight. To democratize robotics research, we developed a cost-effective open-source unicycle robot platform by modifying the entry-level Makeblock mBot, as depicted in Figure 7.2.

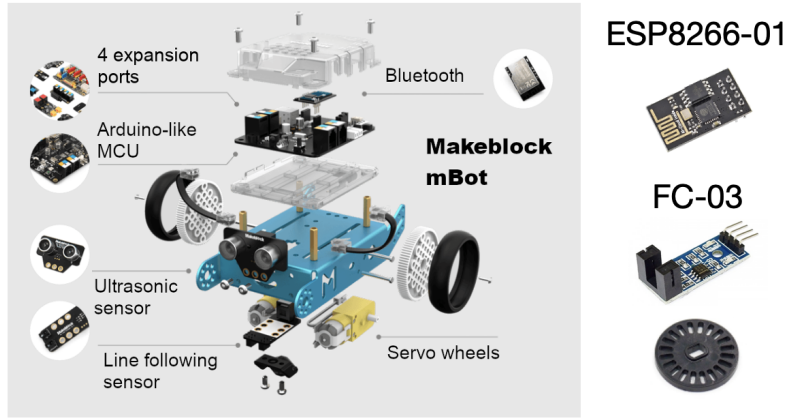


Figure 7.2: Prototype unicycle robot customized from mBot.

For scientific experiments, it is essential to implement two core features that are missing: wireless communication capability and a reliable control interface. The mBot’s onboard Bluetooth module is unsuitable for indoor experiments due to interference issues with multiple devices. A more reliable method involves using a wireless local-area network with the Robot Operating System (ROS), facilitated by the ESP8266-01 module, which supports ROS. Additionally, we develop a control interface based on rotational speed, as the original control based on operating voltage percentage is imprecise due to battery discharge. We incorporate optical wheel encoders to measure the rotational speed of each wheel, using this feedback to more accurately control wheel rotation. With this modified mBot unicycle platform, researchers can conduct various indoor multi-agent experiments.

In Section 7.2.1, we discuss the estimation aspects of indoor experiments, including common methods for locating each agent in a scene.

7.1.2 Automated Vehicle

An automated vehicle is defined as a vehicle that can have its mechanical control interfaces, such as longitudinal acceleration and lateral steering, operated by a third-party electronic device. Through this device, a user can issue valid commands to control the vehicle’s movement. In this dissertation, we specifically examine the aspect of longitudinal control, focusing on the acceleration dynamics.

Unlike unicycle robots, the precise dynamical model for longitudinal acceleration—which involves mechanical models of throttle and brake—is highly complex and often impractically detailed for many practical applications. Instead, a simpler surrogate model is typically used by approximating the longitudinal dynamics with a reduced-order model.

To this end, we present a reduced-order model that describes the input-output dynamics from commanded acceleration to actuated acceleration, utilizing the system identification toolbox in MATLAB.

Let $x(t)$, $v(t)$, and $a(t)$ represent the position, velocity, and acceleration of the ego vehicle at time t , respectively. At any given moment t , one can control the vehicle’s acceleration $a(t)$ through a bounded acceleration command signal $a_{\text{cmd}}(t)$. For simplicity, we parametrize the acceleration dynamics using the following first-order system:

$$\frac{A(s)}{A_{\text{cmd}}(s)} = \frac{k_1}{s + k_2}, \quad (7.2)$$

where $A(s) = \mathcal{L}[a(t)]$ and $A_{\text{cmd}}(s) = \mathcal{L}[a_{\text{cmd}}(t)]$ for parameters k_1 and k_2 . This transfer function (7.2) corresponds to the following state-space model in the time domain:

$$\dot{a}(t) = -k_2 a(t) + k_1 a_{\text{cmd}}(t). \quad (7.3)$$

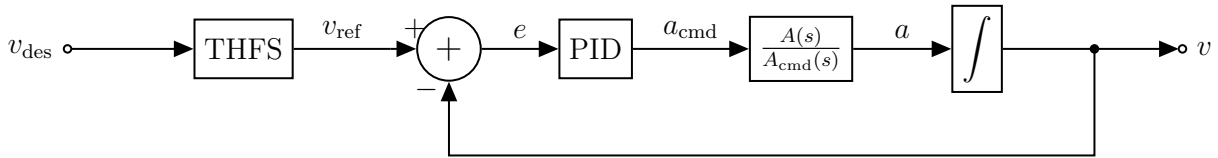


Figure 7.3: Block diagram demonstrating our recent hardware design, which features a time headway follower stopper (THFS) as the safety filter and a PID controller with output saturation as the velocity controller.

However, to ensure safety, we do not expose the interface of a_{cmd} to users. Instead, one can only indirectly access it by commanding a desired velocity signal $v_{\text{des}}(t)$. The desired velocity is passed through a safety filter, producing a reference velocity signal $v_{\text{ref}}(t)$, which is then tracked by a velocity controller. An example of our recent hardware design can be seen in Fig. 7.3.

To illuminate the underlying structure of the system in Figure 7.3, we assume that v_{des} is safe, i.e., $v_{\text{ref}} = v_{\text{des}}$, and that the PID is never saturated. Let the PID takes the following form,

$$\frac{A_{\text{cmd}}(s)}{E(s)} = k_p + \frac{k_i}{s} + \frac{k_d \cdot k_n}{1 + k_n/s} \quad (7.4)$$

$$= \frac{(k_d k_n + k_p)s^2 + (k_p k_n + k_i)s + k_i k_n}{s^2 + k_n s}, \quad (7.5)$$

where $A_{\text{cmd}}(s) = \mathcal{L}[a_{\text{cmd}}(t)]$, $E(s) = \mathcal{L}[e(t)]$, and $e(t) := v_{\text{ref}}(t) - v(t)$ for some parameters k_p, k_i, k_d and k_n . The controllable canonical form of (7.5) is

$$\begin{aligned} \begin{bmatrix} \dot{p}(t) \\ \dot{q}(t) \end{bmatrix} &= \begin{bmatrix} 0 & 1 \\ 0 & -k_n \end{bmatrix} \begin{bmatrix} p(t) \\ q(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} (v_{\text{des}}(t) - v(t)), \\ a_{\text{cmd}}(t) &= \begin{bmatrix} k_i k_n & k_i - k_n^2 k_d \end{bmatrix} \begin{bmatrix} p(t) \\ q(t) \end{bmatrix} \\ &\quad + (k_d k_n + k_p)(v_{\text{des}}(t) - v(t)), \end{aligned} \quad (7.6)$$

where p, q are some internal states of the PID controller.

Combining (7.3) and (7.6), we obtain the vehicle dynamics as the following linear time-invariant (LTI) system

$$\begin{aligned} \begin{bmatrix} \dot{p}(t) \\ \dot{q}(t) \\ \dot{v}(t) \\ \dot{a}(t) \end{bmatrix} &= \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -k_n & -1 & 0 \\ 0 & 0 & 0 & 1 \\ \kappa_1 & \kappa_2 & \kappa_3 & \kappa_4 \end{bmatrix} \begin{bmatrix} p(t) \\ q(t) \\ v(t) \\ a(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 0 \\ -\kappa_3 \end{bmatrix} v_{\text{des}}(t), \\ y(t) &= \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} p(t) & q(t) & v(t) & a(t) \end{bmatrix}^\top, \end{aligned} \quad (7.7)$$

where y is the output of the system and

$$\begin{bmatrix} \kappa_1 \\ \kappa_2 \\ \kappa_3 \\ \kappa_4 \end{bmatrix} = \begin{bmatrix} k_1 k_i k_n \\ k_1 k_i - k_1 k_n^2 k_d \\ -(k_1 k_d k_n + k_1 k_p) \\ -k_2 \end{bmatrix}. \quad (7.8)$$

The values of system parameters are provided in Table 7.1, where k_1 and k_2 are determined via system identification, whereas the rest are selected during controller design experiments. Specifically, system identification is performed on data collected by recording the inputs and outputs of system (7.2) under different driving velocities. The PID is first designed in Simulink and then tested on the actual hardware. It can be verified that system (7.7) with parameters in Table 7.1 is fully controllable and observable.

After identifying a precise reduced-order model, the next challenge is automating the vehicle for scientific research. This challenge is particularly acute for researchers working with

Table 7.1: Parameters of vehicle system dynamics.

k_1	k_2	k_p	k_i	k_d	k_n
1.745	1.566	2.720	0.0656	1.340	7.549

proprietary systems where parts of the vehicle system are encrypted by the original equipment manufacturer. For example, the most common method to interface with a commercial vehicle is through its Control Area Network (CAN) buses. Developed by the automotive industry in the 1980s, CAN buses are ubiquitous in most modern commercial vehicles. These vehicles rely on their internal CAN buses to exchange sensor measurements and control commands in real-time. Therefore, the ability to encode and decode the signals in a vehicle’s CAN buses is essential for automation. However, there is no standard method for encoding and decoding CAN bus signals, and each manufacturer typically has their own proprietary protocols, commonly stored as DBC files, which are generally kept confidential.

Unfortunately, reverse engineering these protocols is a highly challenging task. To address this, the open-source community has achieved some success, notably with projects by Comma.ai, which releases its reverse-engineered DBC files publicly on GitHub.

Assuming one has access to the encoding and decoding protocols for a specific commercial vehicle, it is possible to automate the vehicle by replacing the original ACC module onboard with a custom, ACC-like control module. For instance, this approach was utilized to automate the Cadillac XT5, as documented in Chapter 2. We demonstrate this ACC replacement strategy in Figure 7.4. From the perspective of other electronic modules connected to the CAN buses, the custom control module mimics the signal behavior of the original ACC module. This simulation ensures that other modules continue to transmit sensor measurements or execute control commands from the custom module.

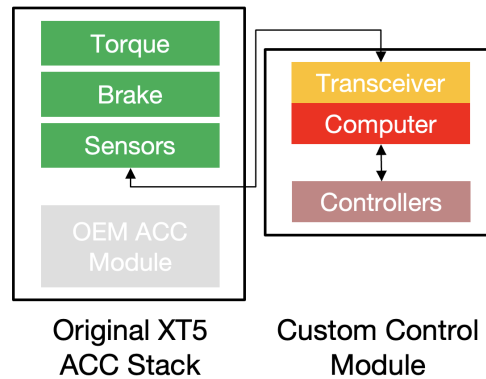


Figure 7.4: Illustration of the ACC bypass system. The OEM ACC module is disabled and replaced with the custom control module at each startup.

After automating and modeling a vehicle, it can be deployed for real-world road tests. In the following discussion, we present an example of an automated vehicle experiment, codenamed MegaVanderTest (MVT). This test aims to mitigate instabilities in traffic flow, which are natural phenomena due to human driving behaviors. Also known as “phantom jams” or stop-and-go waves, these instabilities are a significant source of energy waste. To this end, the MVT project designed a control system, dubbed the MegaController by the research team, for deployment in actual traffic scenarios.

Our field experiment utilized a heterogeneous fleet of approximately 100 longitudinally-controlled vehicles as Lagrangian traffic actuators. The MegaController employs a hierarchical control architecture consisting of two main layers: the Speed Planner and the ACC Controller. The upper layer, the planning layer, is a centralized optimal control algorithm that assigns speed targets to the vehicles, communicated through the LTE cellular network. The lower layer, the control layer, operates on each vehicle, overriding the stock adaptive cruise controller using the onboard sensors. The interplay between these two layers is depicted in a high-level block diagram in Figure 7.5.

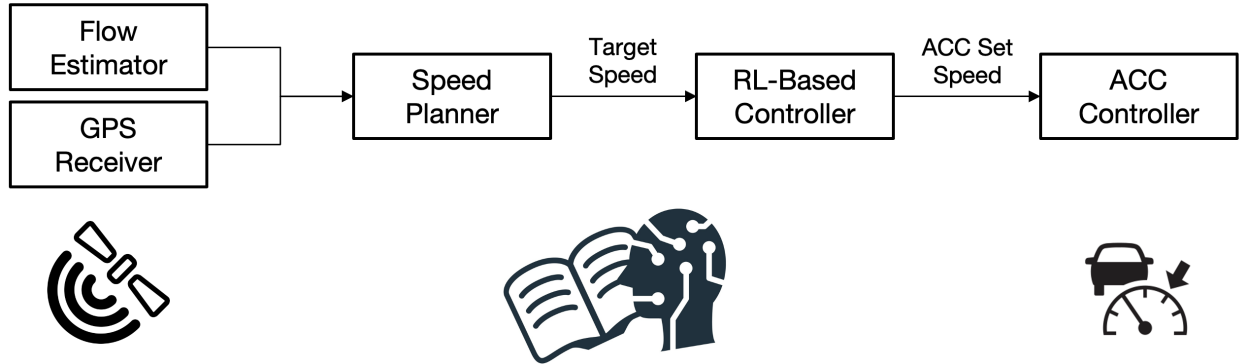


Figure 7.5: Block diagram showing the technical components of the systems developed in the MVT project.

The Speed Planner layer processes live data feeds provided by third parties, along with data from our control vehicles, to perform speed assignment. Its architecture supports the modular integration of standard control techniques, such as optimal control, model predictive control, kernel methods, and others.

The ACC Controller layer allows for flexible implementation of local controllers. Relevant techniques include deep reinforcement learning, model predictive control, and explicit controllers. Depending on the vehicle’s architecture, all onboard sensing data may be accessed by the local controllers, or only selected data. Similarly, control inputs vary across different automakers, ranging from torque or acceleration requests in some vehicles to electronic selection of ACC set points in others.

On a test day, the approximately 100 automated vehicles are dispatched from a parking facility near the test route at 6:00am, aiming for full deployment by 7:30am. The test route is

a section of westbound Interstate 24 near Nashville, TN, as illustrated in Figure 7.7. Drivers are instructed to repeat their routes for 2 hours, or until they request a break. On their return trip, they exit the highway at Exit 60 (Hickory Hollow) to return to the Field Headquarters parking lot and hand in their car keys at a desk before leaving. As automated vehicles return for a break, other drivers, already on a break, queue to take their place, maintaining a continuous flow of vehicles on the route during peak hours. For more details about the routes, logistical choices, driver training, daily schedule, or penetration rate estimates, see the article [4].



Figure 7.6: A view of the parking lot facility during a test day of the MVT.

Finally, we remark that although they were not tested in the MVT, the AKM method from Chapter 2 and the hierarchical MPC method from Chapter 3 are directly applicable, given that the hardware capabilities of the testbed are clearly adequate. Since the ACC systems are automated, the local AKM method can be seamlessly integrated. Furthermore, as each automated vehicle is capable of wireless communication with a remote server, which in turn can send real-time ETA estimates for each vehicle upon request, the nonlocal MPC method can also be implemented. These examples demonstrate that the MVT serves as a versatile experimental design framework that can accommodate a wide range of control methods, potentially informing future experiments.

7.2 Estimation Devices and Experiments

In parallel to the actuation devices and experiments discussed in the previous section, this section focuses on the estimation aspects of devices and experimental design for multi-agent research. We begin by addressing the estimation problem for indoor robotic experiments, exploring three potential approaches to the localization of indoor objects. Next, we supplement the materials from Chapter 4 by presenting the techniques and lessons learned from the collection and curation of the B3D dataset.

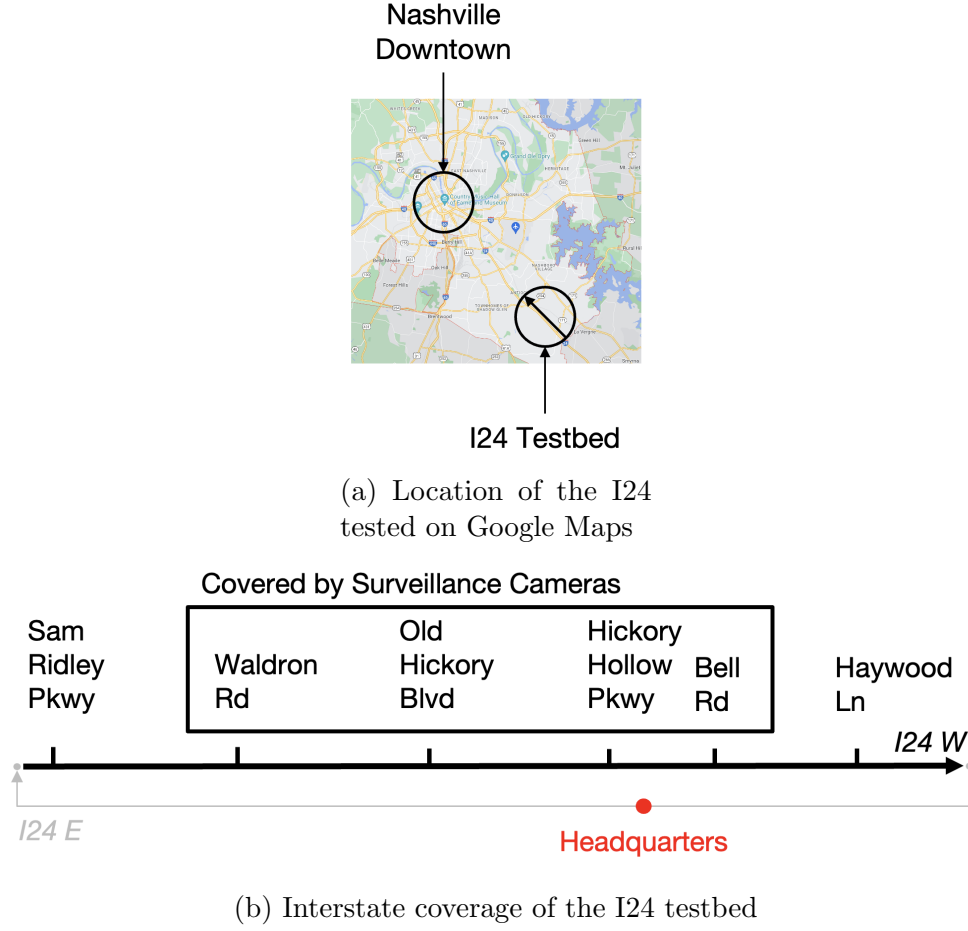


Figure 7.7: The I24 testbed near Nashville, TN.

7.2.1 Indoor Object Localization

Indoor object localization is an estimation problem aimed at determining the real-time positions of objects within a controlled indoor environment. Given the controlled nature of such environments, numerous methods are available to address this challenge. In this dissertation, we review three prevalent approaches: 1) tag-based localization, 2) beacon-based localization, and 3) vision-based localization.

Tag-based localization identifies the positions of indoor objects using artificially constructed tags. A user places a specifically designed tag on a visible surface of an object, which serves as a fiducial marker. These tags, similar in function to QR codes, allow for easy detection of the position and orientation of the host object using a camera. Popular localization tags include ARTag [32], AprilTag [86], and ArUco markers [34], as illustrated in Figure 7.8. Particularly, ArUco markers are well-supported by OpenCV, making them an excellent choice for rapid prototyping. The major advantage of tag-based localization is its

ability to encode location, orientation, and object ID within a single marker. However, the method requires high camera resolution for accurate detection. Low-quality cameras, poor lighting conditions, or excessive distance from the tag can render detection unreliable.

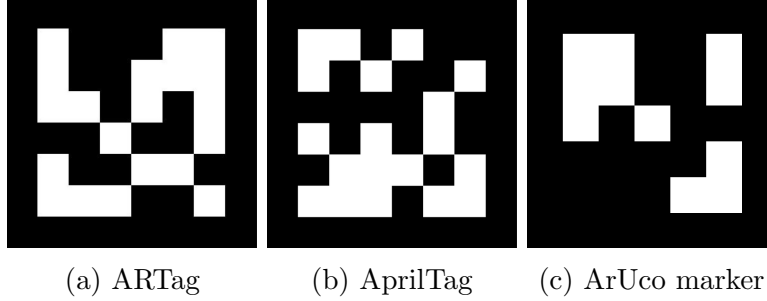


Figure 7.8: Examples of fiducial markers for indoor localization.

Beacon-based localization determines the locations of indoor objects using indoor beacons, capable of measuring both the distance and angle between the beacon and the object. We illustrate the geometry of a beacon-based localization system with an example involving three beacons B_1, B_2 , and B_3 , as shown in Figure 7.9. The known locations of these beacons are (x_1, y_1) , (x_2, y_2) , and (x_3, y_3) , respectively. The measured distances and angles from the beacons to the object are (d_1, α_1) , (d_2, α_2) , and (d_3, α_3) . Using trigonometry, the location of the object R can be determined from a single beacon if both distance and angle are known; with only angles, two beacons are necessary via line intersection; with only distances, three beacons are required via circle intersection. While beacon-based methods are well-understood and can handle noisy sensor data, the main challenges include the scarcity of high-quality, affordable indoor beacons and the complexity of synchronizing measurements from multiple beacons.

Vision-based localization identifies the locations of indoor objects using a range of computer vision methods, from simple color thresholding to advanced neural-network-based object detectors. We present a case study of an indoor localization system based on color thresholding, as depicted in Figure 7.10. In this setup, a green-colored tag is affixed atop each ground robot. By applying color thresholding, preferably in the HSV color space, a binary segmentation mask is generated. Subsequent clustering algorithms determine the positions of the robots in camera coordinates, which can then be transformed into world coordinates using perspective transformation. The primary advantage of color thresholding is its simplicity and robustness—it requires no training data, minimal computation, and is unaffected by resolution. However, the challenge lies in determining the appropriate color threshold, as this color must not appear on any irrelevant objects within the scene.

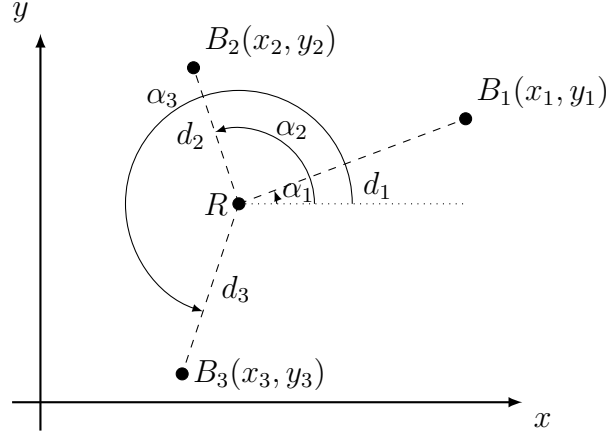


Figure 7.9: Localization based on spatial triangulation.



Figure 7.10: A view of the single-camera indoor localization system using color thresholding.

7.2.2 Aerial Video Collection

Lastly, we briefly describe the generation of data, including 1) the collection of video data, 2) the labeling of video data, and 3) the processing of video data and manual labels.

During the initial phase of video data collection, we positioned the quadcopter directly above the road of interest and maintained it in a hovering state for as long as the battery permitted. A top-view video was recorded at 4K resolution and 30 FPS using the built-in camera. A visualization of the experiment setup is shown in Figure 7.11. Due to limited battery capacity, each recording session was restricted to between 15 and 20 minutes.

After collecting video data, we created a set of image annotations to train a neural network model for vehicle detection. To this end, we manually cropped and labeled 16,002 images. We selected candidate images from the 20 videos by extracting one frame every 15 seconds. We then manually labeled an axis-aligned bounding box for each vehicle in the

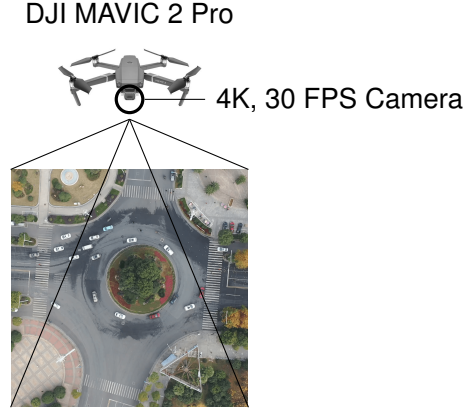


Figure 7.11: An illustration of the experiment setup. The quadcopter hovers over the road of interest, recording the top-view of each traffic participant as they enter and exit the scene.

selected frames using the annotation tool CVAT [107]. Note that an axis-aligned bounding box does not provide information about the orientation of the bounded object. Subsequently, we cropped around significant vehicle clusters in each frame, generating the final annotated image dataset.

This labeling-then-cropping approach is crucial for space efficiency and detection efficacy. Specifically, cropping reduces the size of candidate images by removing irrelevant parts of the frames, such as trees, rooftops, and parking lots. This reduction is particularly beneficial for training on GPUs, which typically have less memory than system RAM. Moreover, cropping ensures that candidate images are of regular proportions, which is vital for the region proposal step of vehicle detection. If we did not crop the frames and applied the trained model directly on elongated frames like those in Figure 4.2, we would encounter significantly reduced detection accuracy due to inaccurate region proposals.

With videos collected and labels annotated, we processed the data through a sequence of computer vision algorithms, as shown in Figure 7.12.

In the first two steps, the raw videos underwent additional post-processing, including stabilization, scaling, and masking. For stabilization, we manually identified the locations of three feature points every 30 frames. Using these sets of three feature points, we computed the homography that projects the corresponding image frame from one set to another. This process allowed us to achieve image stabilization by applying a homography that transforms the projection of every other frame to that of the initial frame. For scaling, we normalized all junction videos to a uniform scale of 20.825 pixels/meter, while all highway videos were normalized to another scale of 6.667 pixels/meter.

In the next three steps, we adopted the classical detect-then-track approach to estimate trajectories for each vehicle in the preprocessed videos. This approach is demonstrated in Figure 7.13. First, we trained a Faster R-CNN model with RetinaNet loss [68] to estimate axis-aligned bounding boxes for each vehicle in every frame. Next, we associated the bound-

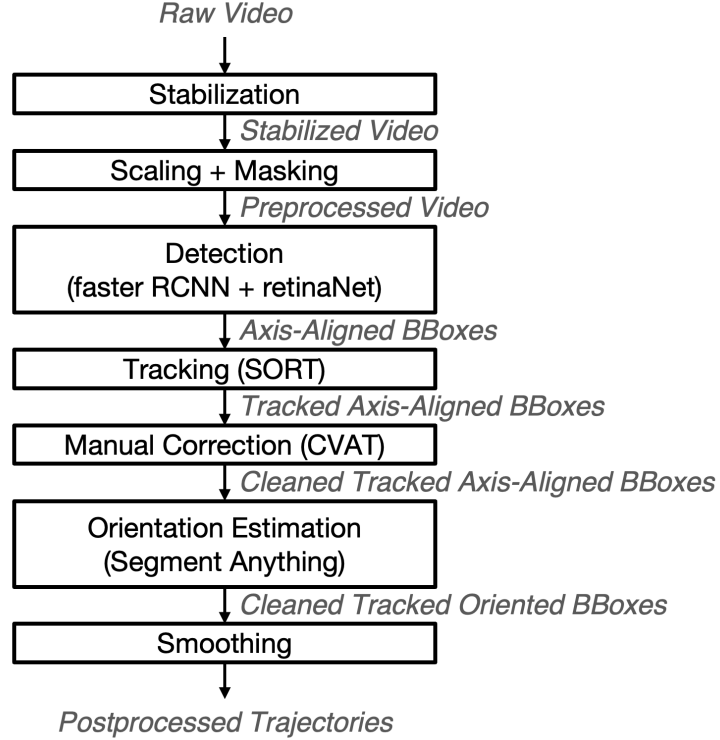


Figure 7.12: The computer vision pipeline deployed for producing the B3D dataset.

ing boxes across frames using SORT to estimate the trajectories of each distinct vehicle [12]. A manual correction step was then required to inspect the correctness of the estimated trajectories and correct errors when necessary, again using the annotation tool CVAT.

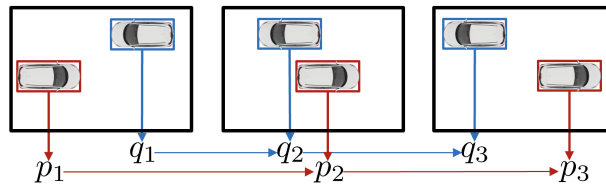


Figure 7.13: A typical detection then tracking approach.

In the final two steps, we refined the trajectories by estimating the orientation of each axis-aligned bounding box in every frame and then smoothing any estimation noises and outliers. While there are many methods to estimate the orientation of axis-aligned bounding boxes, we found that recent large segmentation models work very well. Initially, we used the axis-aligned bounding box to crop out irrelevant parts of the image frame. Next, we applied the Segment Anything model to find segmentation masks for each bounded object [60] and

then estimated the orientation by fitting a minimum-area rectangle to each segmentation mask. The distinction between an axis-aligned and an oriented bounding box is illustrated in Figure 7.14. Finally, with the orientations obtained, we applied a final smoothing step to remove noise and outliers in the estimated positions and orientations using a simple cubic spline smoothing method in the `UnivariateSpline` class implemented in SciPy [27, 126].

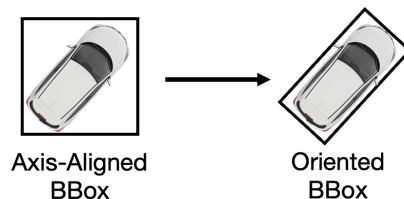


Figure 7.14: Using image segmentation methods to convert an axis-aligned bounding box to an oriented bounding box.

Chapter 8

Conclusion and Future Work

In this dissertation, we explored the planning and control problems in the field of Multi-Agent Robotic Systems (MARSs), particularly focusing on issues encountered in connected and automated driving. Through a series of model developments, theoretical analyses, and empirical experiments, we aimed to address critical challenges such as string instability, decentralized collision avoidance, and scalable coordination among autonomous agents. In this chapter, we summarize the key contents presented in this dissertation and outline a few important directions for future work.

8.1 Chapter Summary

First, we present a brief summary of the content of each key chapter below.

Chapter 2: Sequentially Chained Multi-Agent Systems with Local Observation. This chapter delved into the dynamics of highway platooning, focusing on the prevalent issue of stop-and-go waves caused by string instability. We introduced the Attenuative Kerner’s Model (AKM), a novel Adaptive Cruise Control (ACC) system that utilizes local observations to enhance traffic flow stability. Comprehensive simulations and field experiments demonstrated the AKM’s potential to significantly dampen disruptive traffic waves, reducing fuel consumption and emissions.

Chapter 3: Sequentially Chained Multi-Agent Systems with Non-Local Observation. Here, we enhanced car-following behaviors through the integration of non-local observation capabilities into control systems. The development of a hierarchical Model Predictive Control (MPC) framework marked a significant contribution, enabling more sophisticated handling of dynamic traffic conditions. This approach proved effective in mitigating oscillatory behaviors and improving traffic predictability.

Chapter 4: Partially Meshed Multi-Agent Systems. This chapter presented a decentralized collision avoidance approach, particularly useful in understructured road environments such as unsignalized intersections. The introduction of the Berkeley DeepDrive

Drone (B3D) dataset facilitated research in decentralized motion planning, capturing diverse human driving behaviors and fostering advancements in autonomous vehicle navigation.

Chapter 5: Centrally Connected Multi-Agent Systems. We explored scalable control and communication strategies for centrally connected MASs. A mixed-integer programming approach was developed to optimize trajectory waypoint time scheduling, demonstrating effectiveness in managing large-scale, interconnected agent networks. The proposed methods were validated through detailed simulations.

Chapter 6: Learning-Augmented Planning and Control. In this chapter, we integrated machine learning techniques with traditional optimization methods to enhance planning and control processes in MAS. The Memory-Augmented Model Predictive Control (MAMPC) method, leveraging past experiences for quicker decision-making, showed substantial improvements in system performance, stability, and computational efficiency.

Chapter 7: Devices and Experiments for Multi-Agent Research. We detailed the development and utilization of devices in multi-agent research, such as unicycle robots, automated vehicles, indoor localization, and sensor-equipped quadcopters, which are critical for validating theoretical models. Advanced actuation and estimation devices were highlighted for their role in precise control and data collection in MAS experiments.

8.2 Future Directions

Finally, we highlight, among many, a few interesting directions for future work, as outlined below.

Extended Validation and Deployment. Future research should focus on the broader deployment of models like AKM and hierarchical MPC across diverse traffic scenarios and geographic settings to validate their effectiveness in real-world conditions.

Advanced Machine Learning Techniques.

Implementing sophisticated machine learning techniques to predict vehicle behaviors and enhance control systems remains a promising avenue. This includes deep reinforcement learning and neural networks, which could further optimize traffic flow and energy consumption.

Integration with Urban Traffic Systems. Exploring the integration of MAS models with urban traffic management systems could facilitate smoother transitions between autonomous and human-driven vehicles, enhancing overall traffic efficiency.

Scalability and Cross-Domain Applications. Investigating the scalability of proposed models and control systems within larger and more complex networks, as well as their applicability in other fields such as aerospace, healthcare, and industrial automation, could demonstrate their versatility and broader impact.

8.3 Concluding Remarks

This dissertation aims to address critical challenges within the field of MARSs, particularly in the context of connected and automated driving. The innovative models and control frameworks developed herein offer promising solutions for improving traffic flow stability, reducing emissions, and enhancing the coordination of autonomous agents. By pursuing the identified future directions, we can further refine these systems, ensuring their robustness and applicability in diverse and dynamic environments.

In conclusion, the advancements presented in this work lay a solid foundation for future research and development, paving the way for more efficient, reliable, and adaptive multi-agent systems in the transportation sector and beyond.

Bibliography

- [1] United States Federal Highway Administration. Next generation simulation (ngsim) vehicle trajectories and supporting data, 2016.
- [2] Alessandro Alessio and Alberto Bemporad. A survey on explicit model predictive control. In *Nonlinear Model Predictive Control*, pages 345–369. Springer, 2009.
- [3] Kostas Alexis, George Nikolakopoulos, and Anthony Tzes. Model predictive quadrotor control: attitude, altitude and position experimental studies. *IET Control Theory and Applications*, 6(12):1812–1827, 2012.
- [4] Mostafa Ameli, Sean McQuade, Jonathan W. Lee, Ryan DeLorenzo, Maria Laura Delle Monache, Sharon Hornstein, Malaika Mahmood, Mikail Mahmood, Matthew Nice, Matthew Bunting, Dan Timsit, Riley Wagner, Rita Xu, Benjamin Seibold, Daniel B. Work, Jonathan Sprinkle, Benedetto Piccoli, and Alexandre M. Bayen. Live traffic control experiment via automated vehicles. *IEEE Control Systems Magazine*, 2024.
- [5] Martin Andersen, Joachim Dahl, and Lieven Vandenberghe. CVXOPT: Convex optimization. *Astrophysics Source Code Library*, pages ascl–2008, 2020.
- [6] Brian D. O. Anderson and John B Moore. *Optimal Control: Linear Quadratic Methods*, pages 53–54. Dover Publications, 2018.
- [7] Guillen-Perez Antonio and Cano Maria-Dolores. Multi-agent deep reinforcement learning to manage connected autonomous vehicles at tomorrow’s intersections. *IEEE Transactions on Vehicular Technology*, 71(7):7033–7043, 2022.
- [8] Alexandre M Bayen, Claire J Tomlin, Yinyu Ye, and Jiawei Zhang. An approximation algorithm for scheduling aircraft with holding time. In *2004 43rd IEEE Conference on Decision and Control (CDC)(IEEE Cat. No. 04CH37601)*, volume 3, pages 2760–2767. IEEE, 2004.
- [9] Ernest Benedito, Arnau Doria-Cerezo, Cristian Kunusch, and Josep M Olm. Traffic flow-oriented design and analysis of an adaptive cruise control system. In *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5. IEEE, 2018.

- [10] Ksenia Bestuzheva, Mathieu Besançon, Wei-Kun Chen, Antonia Chmiela, Tim Donkiewicz, Jasper van Doornmalen, Leon Eifler, Oliver Gaul, Gerald Gamrath, Ambros Gleixner, Leona Gottwald, Christoph Graczyk, Katrin Halbig, Alexander Hoen, Christopher Hojny, Rolf van der Hulst, Thorsten Koch, Marco Lübbecke, Stephen J. Maher, Frederic Matter, Erik Mühmer, Benjamin Müller, Marc E. Pfetsch, Daniel Rehfeldt, Steffan Schlein, Franziska Schlösser, Felipe Serrano, Yuji Shinano, Boro Sofranac, Mark Turner, Stefan Vigerske, Fabian Wegscheider, Philipp Wellner, Dieter Weninger, and Jakob Witzig. The SCIP Optimization Suite 8.0. Technical report, Optimization Online, December 2021.
- [11] Ksenia Bestuzheva, Mathieu Besançon, Wei-Kun Chen, Antonia Chmiela, Tim Donkiewicz, Jasper van Doornmalen, Leon Eifler, Oliver Gaul, Gerald Gamrath, Ambros Gleixner, Leona Gottwald, Christoph Graczyk, Katrin Halbig, Alexander Hoen, Christopher Hojny, Rolf van der Hulst, Thorsten Koch, Marco Lübbecke, Stephen J. Maher, Frederic Matter, Erik Mühmer, Benjamin Müller, Marc E. Pfetsch, Daniel Rehfeldt, Steffan Schlein, Franziska Schlösser, Felipe Serrano, Yuji Shinano, Boro Sofranac, Mark Turner, Stefan Vigerske, Fabian Wegscheider, Philipp Wellner, Dieter Weninger, and Jakob Witzig. The SCIP Optimization Suite 8.0. ZIB-Report 21-41, Zuse Institute Berlin, December 2021.
- [12] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *IEEE International Conference on Image Processing*, pages 3464–3468, 2016.
- [13] Rahul Bhadani, Matthew Bunting, Benjamin Seibold, Raphael Stern, Shumo Cui, Jonathan Sprinkle, Benedetto Piccoli, and Daniel B Work. Real-time distance estimation and filtering of vehicle headways for smoothing of traffic waves. In *Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems*, pages 280–290, 2019.
- [14] Gerardo Bledt, Matthew J. Powell, Benjamin Katz, Jared Di Carlo, Patrick M. Wensing, and Sangbae Kim. MIT cheetah 3: Design and control of a robust, dynamic quadruped robot. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2245–2252. IEEE, 2018.
- [15] Francesco Borrelli, Alberto Bemporad, and Manfred Morari. *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press, 2017.
- [16] Francesco Borrelli, Paolo Falcone, Tamas Keviczky, Jahan Asgari, and Davor Hrovat. MPC-based approach to active steering for autonomous vehicle systems. *International Journal of Vehicle Autonomous Systems*, 3(2-4):265–291, 2005.
- [17] Fanping Bu, Han-Shue Tan, and Jihua Huang. Design and field testing of a cooperative adaptive cruise control system. In *Proceedings of the 2010 American Control Conference*, pages 4616–4621. IEEE, 2010.

- [18] Giuseppe C Calafiore and Laurent El Ghaoui. *Optimization Models*. Cambridge University Press, 2014.
- [19] Rohan Chandra and Dinesh Manocha. Gameplan: Game-theoretic multi-agent planning with human drivers at intersections, roundabouts, and merging. *IEEE Robotics and Automation Letters*, 7(2):2676–2683, 2022.
- [20] Shaoru Chen, Mahyar Fazlyab, Manfred Morari, George J Pappas, and Victor M Preciado. Learning lyapunov functions for hybrid systems. In *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*, pages 1–11, 2021.
- [21] Steven W Chen, Tianyu Wang, Nikolay Atanasov, Vijay Kumar, and Manfred Morari. Large scale model predictive control with neural networks and primal active sets. *Automatica*, 135:109947, 2022.
- [22] H-D Chiang, Morris W Hirsch, and Felix Fulih Wu. Stability regions of nonlinear autonomous dynamical systems. *IEEE Transactions on Automatic Control*, 33(1):16–27, 1988.
- [23] Han-Lim Choi, Luc Brunet, and Jonathan P How. Consensus-based decentralized auctions for robust task allocation. *IEEE transactions on robotics*, 25(4):912–926, 2009.
- [24] Hong-Cheol Choi and Inseok Hwang. Data-driven trajectory-based consensus approach to traffic management for manned and unmanned aviation. In *AIAA SCITECH 2024 Forum*, page 0537, 2024.
- [25] Brendon K Colbert and Matthew M Peet. Using trajectory measurements to estimate the region of attraction of nonlinear systems. In *IEEE Conference on Decision and Control*, pages 2341–2347. IEEE, 2018.
- [26] Moritz Diehl, Hans Georg Bock, and Johannes P Schlöder. A real-time iteration scheme for nonlinear optimization in optimal feedback control. *SIAM Journal on Control and Optimization*, 43(5):1714–1736, 2005.
- [27] Paul Dierckx. *Curve and surface fitting with splines*. Oxford University Press, 1995.
- [28] Kurt Dresner and Peter Stone. A multiagent approach to autonomous intersection management. *Journal of artificial intelligence research*, 31:591–656, 2008.
- [29] Gregory Dudek, Michael RM Jenkin, Evangelos Milios, and David Wilkes. A taxonomy for multi-agent robotics. *Autonomous Robots*, 3:375–397, 1996.

- [30] Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Ben Sapp, Charles Qi, Yin Zhou, Zoey Yang, Aurelien Chouard, Pei Sun, Jiquan Ngiam, Vijay Vasudevan, Alexander McCauley, Jonathon Shlens, and Dragomir Anguelov. Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9710–9719, 2021.
- [31] Timm Faulwasser, Tobias Weber, Pablo Zometa, and Rolf Findeisen. Implementation of nonlinear model predictive path-following control for an industrial robot. *IEEE Transactions on Control Systems Technology*, 25(4):1505–1511, 2016.
- [32] M. Fiala. Artag, a fiducial marker system using digital techniques. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 590–596. IEEE, 2005.
- [33] M. R. Flynn, A. R. Kasimov, J.-C. Nave, R. R. Rosales, and B. Seibold. Self-sustained nonlinear waves in traffic flow. *Phys. Rev. E*, 79:056113, 2009.
- [34] Sergio Garrido-Jurado, Rafael Muñoz-Salinas, Francisco José Madrid-Cuevas, and Manuel Jesús Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280–2292, 2014.
- [35] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research*, 32(11):1231–1237, 2013.
- [36] Derek Gloudemans, Yanbing Wang, Junyi Ji, Gergely Zachár, William Barbour, Eric Hall, Meredith Cebelak, Lee Smith, and Daniel B. Work. I-24 motion: An instrument for freeway traffic science. *Transportation Research Part C: Emerging Technologies*, 155:104311, 2023.
- [37] Alexander L Gratzner, Sebastian Thormann, Alexander Schirrer, and Stefan Jakubek. String stable and collision-safe model predictive platoon control. *IEEE Transactions on Intelligent Transportation Systems*, 23(10):19358–19373, 2022.
- [38] George Gunter, Derek Gloudemans, Raphael E Stern, Sean McQuade, Rahul Bhadani, Matt Bunting, Maria Laura Delle Monache, Roman Lysecky, Benjamin Seibold, Jonathan Sprinkle, Benedetto Piccoli, and Work Daniel. Are commercially implemented adaptive cruise control systems string stable? *IEEE Transactions on Intelligent Transportation Systems*, 22(11):6992–7003, 2020.
- [39] Shane Halbach, Phillip Sharer, Sylvain Pagerit, Aymeric P Rousseau, and Charles Folkerts. Model architecture, methods, and interfaces for efficient math-based design and simulation of automotive control systems. Technical report, SAE Technical paper, 2010.

- [40] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.
- [41] Dirk Helbing and Peter Molnar. Social force model for pedestrian dynamics. *Physical review E*, 51(5):4282, 1995.
- [42] Boris Houska, Hans Joachim Ferreau, and Moritz Diehl. An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range. *Automatica*, 47(10):2279–2285, 2011.
- [43] Shuangyao Huang, Haibo Zhang, and Zhiyi Huang. Multi-uav collision avoidance using multi-agent reinforcement learning with counterfactual credit assignment. *arXiv preprint arXiv:2204.08594*, 2022.
- [44] Dom Huh and Prasant Mohapatra. Multi-agent reinforcement learning: A comprehensive survey. *arXiv preprint arXiv:2312.10256*, 2023.
- [45] Jiaxi Huo, Ronghao Zheng, Meiqin Liu, and Senlin Zhang. Integer-programming-based narrow-passage multi-robot path planning with effective heuristics. *arXiv preprint arXiv:2107.12219*, 2021.
- [46] Eunjeong Hyeon, Youngki Kim, Niket Prakash, and Anna G Stefanopoulou. Influence of speed forecasting on the performance of ecological adaptive cruise control. In *Dynamic Systems and Control Conference*, volume 59148, page V001T08A003. American Society of Mechanical Engineers, 2019.
- [47] Daniel Ioan, Ionela Prodan, Sorin Olaru, Florin Stoican, and Silviu-Iulian Niculescu. Mixed-integer programming in motion planning. *Annual Reviews in Control*, 51:65–87, 2021.
- [48] P Ioannou and Z Xu. Throttle and brake control systems for automatic vehicle following. *IVHS Journal*, 1(4):345–377, 1994.
- [49] Ali Jadbabaie, Jie Lin, and A Stephen Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on automatic control*, 48(6):988–1001, 2003.
- [50] Yanzhao Jia, Tomohiro Saito, Yutaro Itoh, Yernar Nukezhanov, and Daniel Görges. Energy-optimal adaptive cruise control in time domain based on model predictive control. *IFAC-PapersOnLine*, 51(31):846–853, 2018.
- [51] Liming Jiang, Yuanchang Xie, Xiao Wen, Danjue Chen, Tienan Li, and Nicholas G Evans. Dampen the stop-and-go traffic with connected and automated vehicles—a deep reinforcement learning approach. In *2021 7th International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, pages 1–6. IEEE, 2021.

- [52] Tor A. Johansen. Toward dependable embedded model predictive control. *IEEE Systems Journal*, 11(2):1208–1219, 2014.
- [53] Mikael Johansson and Anders Rantzer. Computation of piecewise quadratic lyapunov functions for hybrid systems. In *1997 European Control Conference (ECC)*, pages 2005–2010. IEEE, 1997.
- [54] Benjamin Karg and Sergio Lucia. Stability and feasibility of neural network-based controllers via output range analysis. In *IEEE Conference on Decision and Control*, pages 4947–4954. IEEE, 2020.
- [55] Boris S Kerner. Physics of automated driving in framework of three-phase traffic theory. *Physical Review E*, 97(4):042303, 2018.
- [56] Arne Kesting, Martin Treiber, Martin Schönhof, and Dirk Helbing. Adaptive cruise control design for active congestion avoidance. *Transportation Research Part C: Emerging Technologies*, 16(6):668–683, 2008.
- [57] Hassan K. Khalil. *Nonlinear Systems*. Prentice Hall, 3rd edition, 2002.
- [58] Nour Khoudari, Sulaiman Almatrudi, Rabie Ramadan, Joy Carpio, Mengsha Yao, Kenneth Butts, Alexandre M Bayen, Jonathan W Lee, and Benjamin Seibold. Reducing detailed vehicle energy dynamics to physics-like models. *arXiv preprint arXiv:2310.06297*, 2023.
- [59] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.
- [60] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023.
- [61] Sébastien Kleff, Avadesh Meduri, Rohan Budhiraja, Nicolas Mansard, and Ludovic Righetti. High-frequency nonlinear model predictive control of a manipulator. In *IEEE International Conference on Robotics and Automation*, 2021.
- [62] Robert Krajewski, Julian Bock, Laurent Kloecker, and Lutz Eckstein. The highd dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems. In *International Conference on Intelligent Transportation Systems*, pages 2118–2125, 2018.
- [63] Scott Kuindersma, Robin Deits, Maurice Fallon, Andrés Valenzuela, Hongkai Dai, Frank Permenter, Twan Koolen, Pat Marion, and Russ Tedrake. Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot. *Autonomous robots*, 40(3):429–455, 2016.

- [64] Michal Kvasnica, Juraj Hledík, Ivana Rauová, and Miroslav Fikar. Complexity reduction of explicit model predictive control via separation. *Automatica*, 49(6):1776–1781, 2013.
- [65] Jonathan Lee, George Gunter, Rabie Ramadan, et al. Integrated framework of vehicle dynamics, instabilities, energy models, and sparse flow smoothing controllers. In *DI-CPS’21: Proceedings of the Workshop on Data-Driven and Intelligent Cyber-Physical Systems*, New York, NY, USA, 2021. Association for Computing Machinery.
- [66] Chi-Ying Liang and Huei Peng. Optimal adaptive cruise control with guaranteed string stability. *Vehicle System Dynamics*, 32(4-5):313–330, 1999.
- [67] Nathan Lichtlé, Eugene Vinitsky, Matthew Nice, Benjamin Seibold, Dan Work, and Alexandre M. Bayen. Deploying traffic smoothing cruise controllers learned from trajectory data. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2884–2890, 2022.
- [68] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2980–2988, 2017.
- [69] Zuxin Liu, Baiming Chen, Hongyi Zhou, Guru Koushik, Martial Hebert, and Ding Zhao. Mapper: Multi-agent path planning with evolutionary reinforcement learning in mixed dynamic environments. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11748–11754. IEEE, 2020.
- [70] Xiao-Yun Lu and Alexander Skabardonis. Freeway traffic shockwave analysis: exploring the ngsim trajectory data. In *86th Annual Meeting of the Transportation Research Board, Washington, DC*. Citeseer, 2007.
- [71] Li-hua Luo, Hong Liu, Ping Li, and Hui Wang. Model predictive control for adaptive cruise control with multi-objectives: comfort, fuel-economy, safety and car-following. *Journal of Zhejiang University SCIENCE A*, 11(3):191–201, 2010.
- [72] Will Maddern, Geoff Pascoe, Chris Linegar, and Paul Newman. 1 year, 1000km: The oxford robotcar dataset. *The International Journal of Robotics Research*, 36(1):3–15, 2017.
- [73] Zhi-Hong Mao, Eric Feron, and Karl Bilimoria. Stability of intersecting aircraft flows under decentralized conflict avoidance rules. In *18th Applied Aerodynamics Conference*, page 4271, 2000.
- [74] John-Jairo Martinez and Carlos Canudas-de Wit. A safe longitudinal control for adaptive cruise control and stop-and-go scenarios. *IEEE Transactions on Control Systems Technology*, 15(2):246–258, 2007.

- [75] Carlos Mastalli, Rohan Budhiraja, Wolfgang Merkt, Guilhem Saurel, Bilal Hammoud, Maximilien Naveau, Justin Carpentier, Ludovic Righetti, Sethu Vijayakumar, and Nicolas Mansard. Crocoddyl: An efficient and versatile framework for multi-contact optimal control. In *IEEE International Conference on Robotics and Automation*, pages 2536–2542. IEEE, 2020.
- [76] Jacob Mattingley and Stephen Boyd. CVXGEN: A code generator for embedded convex optimization. *Optimization and Engineering*, 13(1):1–27, 2012.
- [77] Sean T McQuade, Chris Denaro, Malaika Mahmood, Jonathan W Lee, Gracie Gumm, Jonathan M Sprinkle, Daniel B Work, Benedetto Piccoli, Benjamin Seibold, and Alexandre M Bayen. Medium-scale to large-scale implementation of cyber-physical human experiments in live traffic. *IFAC-PapersOnLine*, 55(41):83–88, 2022.
- [78] Hanna Michalska and David Q. Mayne. Robust receding horizon control of constrained nonlinear systems. *IEEE transactions on automatic control*, 38(11):1623–1633, 1993.
- [79] Vicente Milanés, Steven E Shladover, John Spring, Christopher Nowakowski, Hiroshi Kawazoe, and Masahide Nakamura. Cooperative adaptive cruise control in real traffic situations. *IEEE Transactions on intelligent transportation systems*, 15(1):296–305, 2013.
- [80] Gerrit JL Naus, Rene PA Vugts, Jeroen Ploeg, Marinus JG van De Molengraft, and Maarten Steinbuch. String-stable cacc design and experimental validation: A frequency-domain approach. *IEEE Transactions on vehicular technology*, 59(9):4268–4279, 2010.
- [81] Luke Ng, Christopher M Clark, and Jan Paul Huissoon. Reinforcement learning of dynamic collaborative driving part i: Longitudinal adaptive control. *International journal of vehicle information and communication systems*, 1(3-4):208–228, 2008.
- [82] Matthew Nice, Matthew Bunting, George Gunter, William Barbour, Jonathan Sprinkle, and Dan Work. Sailing cavs: Speed-adaptive infrastructure-linked connected and automated vehicles. *arXiv preprint arXiv:2310.06931*, 2023.
- [83] Matthew Nice, Nathan Lichtle, et al. The I-24 Trajectory Dataset, September 2021.
- [84] Julian Nubert, Johannes Köhler, Vincent Berenz, Frank Allgöwer, and Sebastian Trimpe. Safe and fast tracking on a robot manipulator: Robust MPC and neural network control. *IEEE Robotics and Automation Letters*, 5(2):3050–3057, 2020.
- [85] Kwang-Kyo Oh, Myoung-Chul Park, and Hyo-Sung Ahn. A survey of multi-agent formation control. *Automatica*, 53:424–440, 2015.
- [86] Edwin Olson. Apriltag: A robust and flexible visual fiducial system. In *2011 IEEE international conference on robotics and automation*, pages 3400–3407. IEEE, 2011.

- [87] Jun Ota. Multi-agent robot systems as distributed autonomous systems. *Advanced engineering informatics*, 20(1):59–70, 2006.
- [88] Brian Paden, Michal Čáp, Sze Zheng Yong, Dmitry Yershov, and Emilio Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on intelligent vehicles*, 1(1):33–55, 2016.
- [89] Thomas Parisini, M Sanguineti, and R Zoppoli. Nonlinear stabilization by receding-horizon neural regulators. *International Journal of Control*, 70(3):341–362, 1998.
- [90] Thomas Parisini and Riccardo Zoppoli. A receding-horizon regulator for nonlinear systems and a neural approximation. *Automatica*, 31(10):1443–1451, 1995.
- [91] Simon Parsons and Michael Wooldridge. Game theory and decision theory in multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 5:243–254, 2002.
- [92] Joel A Paulson and Ali Mesbah. Approximate closed-loop robust model predictive control with guaranteed stability and constraint satisfaction. *IEEE Control Systems Letters*, 4(3):719–724, 2020.
- [93] Laurent Perron and Frédéric Didier. Cp-sat.
- [94] Xiaobo Qu, Yang Yu, Mofan Zhou, Chin-Teng Lin, and Xiangyu Wang. Jointly dampening traffic oscillations and improving energy consumption with electric, connected and automated vehicles: A reinforcement learning based approach. *Applied Energy*, 257:114030, 2020.
- [95] R Rajamani and S.E Shladover. An experimental comparative study of autonomous and co-operative vehicle-follower control systems. *Transportation Research Part C: Emerging Technologies*, 9(1):15–31, 2001.
- [96] James B. Rawlings, David Q. Mayne, and Moritz M. Diehl. *Model Predictive Control: Theory, Computation and Design*. Nob Hill Publishing, 2017.
- [97] Wei Ren. Consensus based formation control strategies for multi-vehicle systems. In *2006 American Control Conference*, pages 6–pp. IEEE, 2006.
- [98] Craig W Reynolds. Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 25–34, 1987.
- [99] Adil EL Rharbali, Hamza Bousfiha, Manal Hasri, and Yassine Salih-Alj. Smart cruise control system with stop&go and predictive road mapping. In *2012 8th IEEE International Symposium on Instrumentation and Control Technology (ISICT) Proceedings*, pages 116–121. IEEE, 2012.

- [100] Arthur Richards and Jonathan P How. Aircraft trajectory planning with collision avoidance using mixed integer linear programming. In *Proceedings of the 2002 American Control Conference (IEEE Cat. No. CH37301)*, volume 3, pages 1936–1941. IEEE, 2002.
- [101] Spencer M Richards, Felix Berkenkamp, and Andreas Krause. The lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems. In *Conference on Robot Learning*, pages 466–476. PMLR, 2018.
- [102] Stefan Richter, Colin N Jones, and Manfred Morari. Real-time input-constrained mpc using fast gradient methods. In *IEEE Conference on Decision and Control*, pages 7387–7393. IEEE, 2009.
- [103] William Riguelle. ‘look out! get back!’ horse-drawn traffic and its challenges in belgian cities in the early modern period. *Urban History*, 50(3):387–403, 2023.
- [104] A Robicquet, A Sadeghian, A Alahi, and S Savarese. Learning social etiquette: Human trajectory prediction in crowded scenes. In *European Conference on Computer Vision*, volume 2, 2020.
- [105] Roman Schmied, Harald Waschl, Rien Quirynen, Moritz Diehl, and Luigi del Re. Nonlinear mpc for emission efficient cooperative adaptive cruise control. *IFAC-PapersOnLine*, 48(23):160–165, 2015.
- [106] Michele Segata, Renato Lo Cigno, Rahul Kumar Bhadani, Matthew Bunting, and Jonathan Sprinkle. A lidar error model for cooperative driving simulations. In *2018 IEEE Vehicular Networking Conference (VNC)*, pages 1–8. IEEE, 2018.
- [107] Boris Sekachev, Nikita Manovich, Maxim Zhiltsov, Andrey Zhavoronkov, Dmitry Kalinin, Ben Hoff, TOSmanov, Dmitry Kruchinin, Artyom Zankevich, DmitriySidnev, Maksim Markelov, Johannes222, Mathis Chenuet, a andre, telenachos, Aleksandr Melnikov, Jijoong Kim, Liron Ilouz, Nikita Glazov, Priya4607, Rush Tehrani, Seungwon Jeong, Vladimir Skubriev, Sebastian Yonekura, vugia truong, zliang7, lizhming, and Tritin Truong. opencv/cvat: v1.1.0, 8 2020.
- [108] G.M. Sessions. *Traffic Devices: Historical Aspects Thereof*. Institute of Traffic Engineers, 1971.
- [109] Ola Shorinwa, Trevor Halsted, Javier Yu, and Mac Schwager. Distributed optimization methods for multi-robot systems: Part 2—a survey. *IEEE Robotics & Automation Magazine*, 2024.
- [110] David Silver. Cooperative pathfinding. In *Proceedings of the aaai conference on artificial intelligence and interactive digital entertainment*, pages 117–122, 2005.

- [111] Edwin Solano-Araque, Guillaume Colin, Guy-Michel Cloarec, Ahmed Ketfi-Cherif, and Yann Chamaillard. Determining vehicle acceleration from noisy non-uniformly sampled speed data for control purposes. *IFAC-PapersOnLine*, 52(5):66–71, 2019.
- [112] Raphael E Stern, Yuche Chen, Miles Churchill, Fangyu Wu, Maria Laura Delle Monache, Benedetto Piccoli, Benjamin Seibold, Jonathan Sprinkle, and Daniel B Work. Quantifying air quality benefits resulting from few autonomous vehicles stabilizing traffic. *Transportation Research Part D: Transport and Environment*, 67:351–365, 2019.
- [113] Raphael E Stern, Shumo Cui, Maria Laura Delle Monache, et al. Dissipation of stop-and-go waves via control of autonomous vehicles: Field experiments. *Transportation Research Part C: Emerging Technologies*, 89:205–221, 2018.
- [114] Yuki Sugiyama, Minoru Fukui, Macoto Kikuchi, Katsuya Hasebe, Akihiro Nakayama, Katsuhiro Nishinari, Shin-ichi Tadaki, and Satoshi Yukawa. Traffic jams without bottlenecks—experimental evidence for the physical mechanism of the formation of a jam. *New journal of physics*, 10(3):033001, 2008.
- [115] Sean Summers, Colin N. Jones, John Lygeros, and Manfred Morari. A multiresolution approximation method for fast explicit model predictive control. *IEEE Transactions on Automatic Control*, 56(11):2530–2541, 2011.
- [116] Jie Sun, Zuduo Zheng, and Jian Sun. The relationship between car following string instability and traffic oscillations in finite-sized platoons and its use in easing congestion via connected and automated vehicles with idm based controller. *Transportation Research Part B: Methodological*, 142:58–83, 2020.
- [117] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Sheng Zhao, Shuyang Cheng, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2446–2454, 2020.
- [118] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT press, 2018.
- [119] Darbha Swaroop and J Karl Hedrick. String stability of interconnected systems. *IEEE Transactions on Automatic Control*, 41(3):349–357, 1996.
- [120] Shin-ichi Tadaki, Macoto Kikuchi, et al. Phase transition in traffic jam experiment on a circuit. *New Journal of Physics*, 15(10):103034, 2013.

- [121] Claire Tomlin, George J Pappas, and Shankar Sastry. Conflict resolution for air traffic management: A study in multiagent hybrid systems. *IEEE Transactions on automatic control*, 43(4):509–521, 1998.
- [122] Ufuk Topcu, Andrew Packard, and Peter Seiler. Local stability analysis using simulations and sum-of-squares programming. *Automatica*, 44(10):2669–2675, 2008.
- [123] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. Congested traffic states in empirical observations and microscopic simulations. *Physical Review E*, 62(2):1805, 2000.
- [124] Bambang Riyanto Trilaksono. Distributed consensus control of robot swarm with obstacle and collision avoidance. In *2015 2nd International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE)*, pages 2–2. IEEE, 2015.
- [125] Cornelis Van Tilburg. *Traffic and Congestion in the Roman Empire*. Routledge, 2007.
- [126] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.
- [127] Jinran Wang, Peng Dong, Zhongliang Jing, and Jin Cheng. Consensus-based filter for distributed sensor networks with colored measurement noise. *Sensors*, 18(11):3678, 2018.
- [128] Meng Wang, Winnie Daamen, Serge P Hoogendoorn, and Bart Van Arem. Connected variable speed limits control and car-following control with vehicle-infrastructure communication to resolve stop-and-go waves. *Journal of Intelligent Transportation Systems*, 20(6):559–572, 2016.
- [129] Cathy Wu, Alexandre M. Bayen, and Ankur Mehta. Stabilizing traffic with autonomous vehicles. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6012–6018, 2018.
- [130] Cathy Wu, Abdul Rahman Kreidieh, Kanaad Parvate, Eugene Vinitzky, and Alexandre M Bayen. Flow: A modular learning framework for mixed autonomy traffic. *IEEE Transactions on Robotics*, 38(2):1270–1286, 2021.
- [131] Fangyu Wu. Memory-augmented model predictive control: Numerical experiments. <https://github.com/fyw85/mampc>, 2021.

- [132] Fangyu Wu and Alexandre M Bayen. A hierarchical mpc approach to car-following via linearly constrained quadratic programming. *IEEE Control Systems Letters*, 7:532–537, 2022.
- [133] Fangyu Wu, Raphael E Stern, Shumo Cui, Maria Laura Delle Monache, Rahul Bhadani, Matt Bunting, Miles Churchill, Nathaniel Hamilton, Benedetto Piccoli, Benjamin Seibold, Jonathan Sprinkle, and Daniel Work. Tracking vehicle trajectories and fuel rates in phantom traffic jams: Methodology and data. *Transportation Research Part C: Emerging Technologies*, 99:82–109, 2019.
- [134] Fangyu Wu, Dequan Wang, Minjune Hwang, Chenhui Hao, Jiawei Lu, Jiamu Zhang, Christopher Chou, Trevor Darrell, and Alexandre Bayen. Decentralized vehicle coordination: The berkeley deepdrive drone dataset. *arXiv preprint arXiv:2209.08763*, 2022.
- [135] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2, 2019.
- [136] Wei Zhan, Liting Sun, Di Wang, Haojie Shi, Aubrey Clausse, Maximilian Naumann, Julius Kummerle, Hendrik Konigshof, Christoph Stiller, Arnaud de La Fortelle, and Masayoshi Tomizuka. INTERACTION Dataset: An INTERnational, Adversarial and Cooperative moTION Dataset in Interactive Driving Scenarios with Semantic Maps. *arXiv:1910.03088 [cs, eess]*, September 2019.
- [137] Xiaojing Zhang, Monimoy Bujarbaruah, and Francesco Borrelli. Safe and near-optimal policy learning for model predictive control using primal-dual neural networks. In *2019 American Control Conference*, pages 354–359. IEEE, 2019.
- [138] Riccardo Zoppoli, Marcello Sanguineti, Giorgio Gnecco, and Thomas Parisini. *Neural Approximations for Optimal Control and Decision*. Springer, 2020.