Title

Predictive Control for Motion Coordination of Connected and Automated Vehicles

Author

Fallah Firoozi, Roya

Publication Date

2021

Peer reviewed|Thesis/dissertation

Predictive Control for Motion Coordination of Connected and Automated Vehicles

by

Roya Fallah Firoozi

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering – Mechanical Engineering

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Francesco Borrelli, Chair
Professor Roberto Horowitz
Professor Murat Arcak
Professor Anil Aswani

Summer 2021

Predictive Control for Motion Coordination of Connected and Automated Vehicles

Abstract

Predictive Control for Motion Coordination of Connected and Automated Vehicles

by

Roya Fallah Firoozi

Doctor of Philosophy in Engineering – Mechanical Engineering

University of California, Berkeley

Professor Francesco Borrelli, Chair

Advances in vehicular communication technologies have the potential to facilitate autonomous and cooperative driving in the near future. Connectivity technologies such as Vehicle-to-Cloud (V2C) and Vehicle-to-Vehicle (V2V) communication enhance the vehicles' awareness and enable cooperation. Connected and Automated Vehicles (CAVs) are able to collaboratively plan and execute driving maneuvers by sharing their perceptual knowledge and future plans.

This thesis presents motion planning algorithms, control strategies and estimation techniques that incorporate the information received via connectivity technologies such as V2C and V2V communication to improve the safety and performance of a single autonomous vehicle or multiple coordinated autonomous vehicles.

Starting with a single vehicle case, the design of estimation, control and planning strategies which utilize V2C connectivity are presented. It is shown the vehicle localization performance is improved by exploiting prior knowledge of a road grade map built using V2C communication. Also, the design of a hierarchical control system including high-level long-term planner and low-level real-time control is presented that employs the traffic condition and road grade data using V2C connectivity.

For the two-vehicle case, the design of a leader-follower coordination using V2V communication is proposed. A safe adaptive cruise control system is presented in which the ego (follower) vehicle receives the predicted trajectory of the front (leader) vehicle. Safety is achieved by designing a robust control invariant set. The set computation is suitable for online applications and is less conservative compared to the state of the art.

For the case of multiple vehicles, the design of safe and efficient coordination algorithms which benefit from V2V communication technologies is presented . The coordination strategies are classified in the two categories of centralized and distributed. The proposed methods are general and applicable to various multi-robot settings. By using the theory of Model Pre-

dictive Control (MPC) and strong duality, provably safe algorithms are presented for collaborative navigation of multiple heterogeneous autonomous vehicles with different (arbitrary) polytopic shapes and different dynamical models in tight environments. The optimization-based centralized coordination strategies are presented for formation, reconfiguration and autonomous navigation of CAVs, traveling on public roads. Using the proposed approach, CAVs are able to form single or multi-lane platoons of various geometrical configurations. They are able to reshape and adjust their configurations according to changes in the environment.

In the last part of the thesis, a distributed coordination algorithm is presented that exploits the problem structure to decompose the large optimization problem into smaller local sub-problems solved in parallel. Using this approach, the vehicles cooperate (while communicating their intentions to the neighbors) and compute collision-free paths in a distributed way to navigate in tight environments in real-time.

Contents

List of Figures

List of Tables

Acknowledgments

Foremost, I would like to express my sincere gratitude to my PhD advisor, Professor Francesco Borrelli for the continuous support, valuable advice, guidance and patience during my PhD studies. I would like to extend my thanks to all the professors who shared their knowledge and experience with me. A special thanks to my committee members Professor Roberto Horowitz, Professor Murat Arcak and Professor Anil Aswani for their insightful comments and suggestions.

I would like to give special thanks to Dr. Xiaojing (George) Zhang, Dr. Jacopo Guanetti , Dr. Laura Ferranti, Dr. Satadru Dey and Dr. Shima Nazari, for their valuable guidance throughout my studies. A special acknowledgement goes to my lab-mates Ugo Rosalia and Monimoy Bujarbaruah for the fruitful conversations and to Youngkeun (Eric) Choi and Yeojun Kim for their help in experimental works. I would like to thank the undergraduate students that I worked with during my PhD, including Sebastian Nejadnik, Geoffrey Ding and Ryan O'Gorman for their hard work and enthusiasm.

I would like to thank my husband Saman Fahandezh-saadi for his love and support all through my studies. Finally, this thesis is dedicated to my parents Farahnaz Dahaj and Saeid Firoozi for all years of unwavering support, love and encouragement.

Chapter 1

Introduction

We are captivated by the promise that autonomous robots will soon be part of our daily lives [38]. The automotive industry is investing billions in self-driving technologies [74], maritime-transportation companies are already developing autonomous boats [50], and retail companies are investing in autonomous warehouse robots [65]. These technologies have the potential to reduce fatalities, to improve transportation efficiency, and to enhance the overall quality of life [22]. On the other hand, recent advances in communication technologies enhance the environmental perception of these robots and enable collaboration among them. As the number of robots in use increase year by year, reliable cooperation and collaboration among them is a necessity. They should be able to communicate and negotiate to navigate in the environment and accomplish different tasks. Hence, it is crucial to design and develop efficient and safe motion planning algorithms, control strategies as well as estimation techniques that incorporate the information received via connectivity and communication interfaces to further improve the safety and performance of autonomous systems.

Advances in vehicular communication technologies are expected to facilitate cooperative driving in the future. Imagine a group of Connected and Automated Vehicles (CAVs) traveling on a highway, as shown in Fig. 1.1 in a geometric formation pattern. They are able to are able to collaboratively plan and execute driving maneuvers by sharing their perceptual knowledge and future plans. For example, if an obstacle appears in a lane or when the traffic is slow in one lane, using Vehicle-to-Vehicle (V2V) communication technology, the vehicles can exchange information and cooperate to perform collaborative maneuvers and merge into lanes with faster traffic, as shown in Fig. 1.2.

Figure 1.1: The right arrow shows direction of motion.

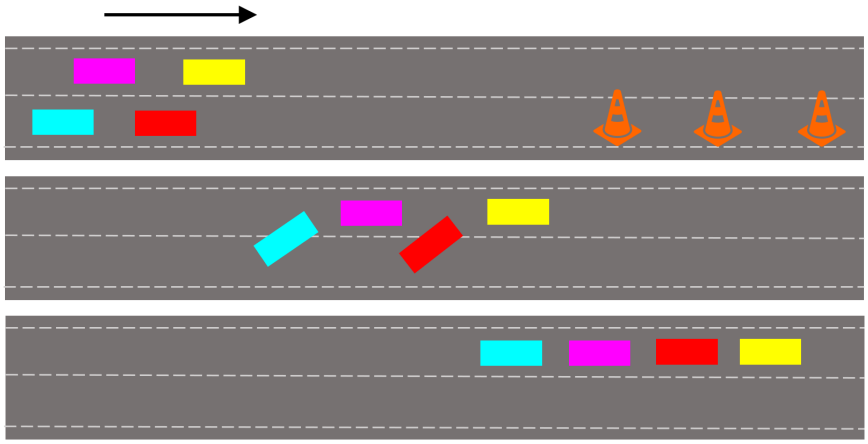


Figure 1.2: Collaborative maneuver of multiple vehicles: The vehicles perform cooperative lane-change and merge into a single lane and reconfigure to a new formation pattern.

These collaborative maneuvers are beneficial and can enhance transportation mobility, safety, energy efficiency, comfort and emission reduction. However, in planning and execution of these types of collaborative maneuvers different challenges arise due to tight available space, fast changing dynamic environment, limited computational resources for real-time implementation and more. When a group of cooperative vehicles travel on highway in a formation pattern, they maintain a close inter-vehicle distance. Therefore, both formation geometry (longitudinal and lateral inter-vehicle spacing) and road geometry (lane width) restrict the motion of the vehicles within the platoon and result in creating a tight environment. In addition, since the vehicles are traveling at highway speed, in a dynamic environment, the design of planning algorithms must be well-suited for real-time implementation considering hardware/software limitations. Furthermore, safety guarantees are crucial aspects of control design for these safety-critical systems in order to avoid collisions. In this work, these challenges are addressed. The main focus is to leverage connectivity and communication interfaces to further improve estimation techniques, motion planning algorithms and control strategies for single and multiple autonomous vehicles.

The coordination task among multiple autonomous agents is defined as achieving a desired formation or reaching a goal set, while avoiding collision with static and dynamic obstacles in the environment. In a constrained optimization framework, the goal set is incorporated in the cost function and collision avoidance relations are imposed as the constraints. Also, dynamic models, states and inputs limitations are enforced as constraints, as well. In this work, utilizing constrained optimization scheme, first a leader-follower coordination strategy for two-vehicle coordination is introduced. Then a centralized coordination scheme for multiple robots is presented. Afterwards, a distributed coordination algorithm is proposed that exploits the structure of the centralized problem to decompose the large optimization problem into smaller local sub-problems solved in parallel, which allows the robots to coordinate

in real-time. The thesis chapters are summarized below.

Chapter 2, deals with a single vehicle connectivity to cloud. The performance of different modules of autonomous driving including localization, planning and control is improved by leveraging the information received using Vehicle-to-Cloud (V2C) technology. For example, a vehicle connects to the Google Elevation API and builds an elevation map of a given route. The vehicle then uses the map to estimate its position along the route by extracting road slope as a localization feature and matching the inclination measurements with the map. Also, for long-term planing, exploiting traffic condition and road geometry information received via V2C, the vehicle optimizes its velocity trajectory, given the origin and destination of the trip. Additionally, for real-time control, a predictive controller employs the same road geometry data as preview information and finds the optimal control strategy accordingly.

Chapter 3, is devoted to leader-follower coordination design for two vehicles. The coordination design is presented as a safe Adaptive Cruise Control system (ACC). ACC controls the vehicle longitudinal dynamics. Using ACC, the ego (follower) vehicle automatically adjusts its acceleration and deceleration to maintain a proper minimum safe distance from the vehicle ahead (leader). In the presented design, ACC controller exploits the information obtained via V2C and V2V connectivity to improve performance and safety. In conventional ACC, the desired inter-vehicle gap is specified as constant space gap, or time gap or time headway policy. However, non of these approaches can guarantee safety. To guarantee safety at all time, the theory of invariant sets is used. A numerical approach to compute a control invariant set which makes use of the road grade preview and the future state trajectory of the leading car, transmitted through V2V communication, is presented. This set is incorporated as terminal constraint of the ACC controller to guarantee safety. Compared to the existing literature, the presented design does not require reachability analysis using level-set method [2] which is computationally expensive especially for nonlinear systems and therefore not well-suited for real-time applications. Compared to [42], the presented design computes the control invariant set for a vehicle nonlinear model and does not rely on a simplified linear dynamic model. Compared to [70], the presented design does not consider bounds on parameters such as road slope parameter and instead the presented design obtains the value of the road slope parameter using V2C, which leads to less conservative safe set and therefore a closer inter-vehicle spacing.

Chapter 4, deals with centralized motion coordination of multiple robots, in particular CAVs. An architecture for autonomous navigation of tight multi-lane platoons travelling on public roads is presented. Using the proposed approach, CAVs are able to form single or multi-lane platoons of various geometrical configurations. They are able to reshape and adjust their configurations according to changes in the environment. However, since reconfiguration of multi-lane platoons requires planning in tight environment, the road structure and the vehicles dimensions are modeled as exact sizes with no approximation or enlargement. The vehicles are described as polytopic sets and the collision avoidance among them is modeled as enforcing a minimum distance between the polytopes using strong duality theory. The proposed architecture consists of two main components: an offline motion planner system and an online hierarchical control system. The motion planner uses an optimization-based

approach for cooperative formation and reconfiguration in tight spaces. A constrained optimization scheme is used to plan smooth, dynamically feasible and collision-free trajectories for all the vehicles within the platoon. This chapter addresses online computation limitations by employing a family of maneuvers precomputed offline and stored on a look-up table on the vehicles. The online hierarchical control system is composed of three levels: a traffic operation system (TOS), a decision-maker, and a path-follower. The TOS determines the desired platoon reconfiguration. The decision-maker checks the feasibility of the reconfiguration plan based on real-time information about the surrounding traffic. The reconfiguration maneuver is executed by a low-level path-following feedback controller in real-time. Compared to the behavior-based planning [6], in which the desired formation is achieved using a sequence of motion primitives, the proposed method does not require extensive tuning efforts. Also compared to motion primitive approach, the proposed approach can provably enforce the collision avoidance constraints. In addition, compared to behavior-based planning, the proposed approach does not require solving a mixed-integer program which can be hard to formulate and analyze mathematically.

Chapter 5, is devoted to distributed motion coordination of multiple robots. The primal-dual setting introduced in the previous chapter leads itself to decomposition and parallelization. Starting from the centralized implementation of the NMPC problem, it is shown how to exploit the problem structure to allow the robots to cooperate (while communicating their intentions to the neighbors) and compute collision-free paths in a distributed way in real time. This chapter presents a distributed method for multi-robot coordination based on nonlinear model predictive control (NMPC) and dual decomposition. The presented approach allows the robots to coordinate in tight spaces (e.g., highway lanes, parking lots, warehouses, canals, etc.) by using a polytopic description of each robot's shape and formulating the collision avoidance as a dual optimization problem. The proposed method accommodates heterogeneous teams of robots (i.e., robots with different polytopic shapes and dynamic models can be part of the same team) and can be used to avoid collisions in n-dimensional spaces. By relying on an alternating optimization scheme, the proposed design decouples the optimization of the robot states and of the collision-avoidance variables to create real time coordination strategies. The proposed distributed coordination design is compared with the centralized NMPC design to show the computational benefits of the proposed distributed algorithm. Compared to [39], the proposed approach does not require the solution of a MILP problem that can be computationally expensive to solve. In addition, compared to [4, 16, 55] the presented approach does not require any linearization (which could reduce the solution space of the problem) of the collision-avoidance constraints. Also compared to [4], the presented approach does not require the use of motion primitives (the robot dynamics are directly included in the NMPC formulation). Compared to [72], the proposed strategy allows to specify a desired distance between the robots, instead of using separating hyperplanes. Inspired by [80, 79], the method uses dual optimization to formulate the collision avoidance constraints. Compared to [80, 79], however, the presented method exploits the structure of the coordination problem to solve it in a distributed fashion.

Chapter 2

Vehicle-to-Cloud Connectivity

2.1 Introduction

A single vehicle can leverage connectivity to cloud to enhance the performance of different modules of autonomous driving including localization, planning and control. In this chapter, the vehicle receives the traffic condition and road geometry information for a given route through V2C connectivity. Then a performance-enhanced localization design is introduced that exploits road geometry data. For example, the vehicle connects to the Google Elevation API and build an elevation map of a given route. The vehicle can then use the map to estimate its position along the route by extracting road slope as a localization feature and matching the inclination measurements with the map. In addition a hierarchical control system consisting of high-level planner and low-level control is designed that makes use of V2C connectivity and employ the received information to improve its performance. Also, for long-term planing, exploiting traffic condition and road geometry information received via V2C, the vehicle optimizes its velocity trajectory, given the origin and destination of the trip. Additionally, for real-time control, a predictive controller employs the same road geometry data as preview information and find the optimal control strategy accordingly.

Localization in autonomous driving applications refers to determining the vehicle's position and attitude. Navigation, motion planning and real-time control rely on accurate localization. Inaccurate localization can be detrimental to performance and may lead to accidents. The Global Positioning System (GPS) is able to provide positioning with meter-level accuracy in clear open sky, but since its accuracy suffers from degraded satellite availability or multi-path error in city areas, a stand-alone GPS is not reliable. A common method to overcome these limitations is using dead-reckoning techniques. GPS data and dead-reckoning information extracted from vehicle's on-board sensors can be fused to improve the position estimate of the vehicle [14], [59]. Since dead-reckoning estimates the current position based on a previously determined position, it is subject to significant cumulative error. In sparse GPS environments like tunnels, underground passages or forested paths, using only dead-reckoning may lead to large estimation errors. A solution to this problem is incorporating

sensor data with a prior road map to correct the position estimates. Map-aided localization approaches exploit road features including lane markings, traffic signs, curbs and buildings to correlate the vehicle states to the prior road map.

Today's self driving cars equipped with three-dimensional (3D) Light Detection and Ranging (LIDAR) scanners are capable of localizing themselves with centimeter-level accuracy. High accuracy localization requires a high definition map of the environment. LIDAR generates point clouds by illuminating laser pulses to the surroundings and measuring the reflected results. In order to build a high-resolution map of the environment, a survey vehicle is driven in the road and measurements of GPS, Inertial Measurement Unit (IMU), wheel odometry and LIDAR data are integrated together. Once an a-priori map of the environment is generated, the vehicle can match the features extracted from LIDAR point clouds with the features stored in the map to correct its position estimate in real-time [11], [75]. Since high-accuracy LIDAR is expensive, a cheaper way to do localization is to use LIDAR for mapping and camera for localization. A survey vehicle equipped with LIDAR generates a Point Cloud Map (PCL) of the road and autonomous cars can make use of just camera to localize itself within the (3D) pre-built PCL [15].

Vision-based matching of road features to a pre-built map of the environment can augment localization performance [51]. However, these feature-based localization methods yield accurate localization in feature-rich environments like urban areas. In rural and suburbans areas, where density of features decreases dramatically, these approaches might not be reliable. A feature that is present in both urban and country roads is road grade. This chapter proposes to make use of a road grade map as the a-priori map for localization. The proposed localization method uses an Extended Kalman Filter (EKF) that combines data provided by the on-board sensors dead-reckoning information with the grade map to accurately estimate the vehicle's longitudinal position. Although the application of the proposed approach is restricted to hilly roads, it can correct dead-reckoning in extended GPS blackout or augment IMU/GPS system where GPS signal is available. Also this method can be combined with other techniques discussed above to improve the localization performance.

Applications that benefit from localization include planning and control systems that rely on the route spatial data like road grade and curvature, speed limit, road optimal reference velocity trajectory and distance to the next stop sign or traffic signal. The proposed localization approach is validated on a control architecture comprised of two levels. At the high-level, the vehicle velocity trajectory over a given route is optimized, by incorporating traffic flow and road grade data into the problem. At the low-level, a Model Predictive controller (MPC) follows the optimal reference velocity trajectory calculated by high-level planner. The simulation results show how errors in position estimation affect energy consumption and confirm the effectiveness of the presented approach.

2.2 Overview

A map-aided vehicle localization method for GPS-denied environments is presented. This approach exploits prior knowledge of the road grade map and vehicle on-board sensor measurements to accurately estimate the longitudinal position of the vehicle. Real-time localization is crucial to systems that utilize position-dependent information for planning and control. The effectiveness of the localization method is validated on a hierarchical control system. The higher level planner optimizes the vehicle velocity to minimize the energy consumption for a given route by employing traffic condition and road grade data. The lower level is a cruise control system that tracks the position-dependent optimal reference velocity. Performance of the proposed localization algorithm is evaluated using both simulations and experiments.

The rest of this chapter is structured as follows. Section 2.3 describes road grade map generation and presents the localization approach using EKF formulation. Section 2.4 shows simulation and experimental results for localization. Section 2.5 explains the hierarchical control system. Section 2.6 illustrates the simulation results for control system and section 2.7 makes concluding remarks.

2.3 Localization

In case of temporary GPS loss, odometry sensors such as wheel speed encoder can be used to estimate the change of position. However, this method is sensitive to errors due to the integration of velocity measurements over time. To mitigate this cumulative error, the proposed design fuses the sensor measurements from wheel speed encoders and accelerometer with grade data extracted from a global road map using the EKF. This section describes road grade map generation and process and measurement models that are included in the EKF estimator.

Road Grade Map Generation

Roadway elevation data is crucial for many transportation applications. To generate a grade map for the entire road, a high-precision Real Time Kinematic (RTK) GPS can be used to measure latitude, longitude and altitude along the route. Also several APIs such as Google Elevation API provide elevation data for all locations on earth surface. The elevation data is queried for specified coordinates and the road elevation profile is obtained. This API also provides the resolution of each elevation sample, defined as the maximum horizontal distance between data points from which the elevation was interpolated. Digital Elevation Models (DEMs) data throughout the earth's surface has been provided by the NASA Shuttle Radar Topographic Mission (SRTM). This elevation data in the United States is available in National Elevation Dataset (NED), provided by the Geological Survey (USGS), at resolutions between 1 arc-second (about 30 meters) and 1/9 arc-second (about 3 meters) depending on

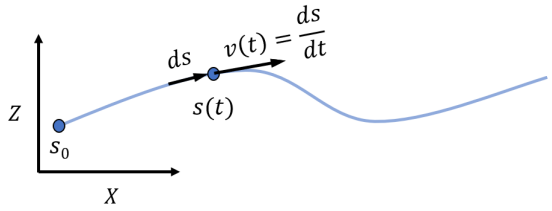


Figure 2.1: Point mass model of the vehicle traveling along a hilly road. The Z axis is the altitude and the X axis is the horizontal distance.

the location. According to [76] that assessed the accuracy of road elevation data extracted from Google Earth, root mean squared error (RMSE) and standard deviation of roadway elevation error are 2.27 meters and 2.27 meters, respectively, even in the areas where USGS NED provides 1/3 arc-second (about 10 meters) resolution. Thus, Google Elevation API provides sufficient accuracy and resolution for generating road grade map.

Extended Kalman Filter (EKF)

The vehicle is modeled as a point mass moving along a path with velocity v, as seen in Figure 2.1. The system state at time t is

$$x(t) = [s(t)\ v(t)]^T,$$

where $s(t)$ and $v(t)$ are the vehicle position and velocity respectively. The process model is a nonlinear function f of the state x and control input u. Process noise w is assumed to be additive and normally distributed with zero mean and covariance of Q, $w \sim \mathcal{N}(0, Q)$. The measurement model is also a nonlinear function h that maps the state x to output y. The measurement noise is assumed to be additive Gaussian noise with zero mean and covariance of R, $\eta \sim \mathcal{N}(0, R)$. EKF linearizes the nonlinear system about the current estimates.

$$\begin{aligned} \dot{x} &= f(x, u) + Gw \\ y &= h(x) + \eta. \end{aligned} \tag{2.1}$$

Process model

The longitudinal kinematic model of the vehicle is defined as

$$\begin{aligned} \dot{s} &= v \\ \dot{v} &= a + w, \end{aligned}$$

where a is the longitudinal acceleration treated as a known input. The system is discretized using Euler method with constant sampling time interval Δt.

$$s(t+1) = s(t) + v(t)\ \Delta t$$
$$v(t+1) = v(t) + (a(t) + w(t))\ \Delta t.$$

Since Δt is small, the second-order terms are neglected. The longitudinal acceleration $a(t)$ is derived from the acceleration measured by the vehicle accelerometer. When a vehicle is moving on an inclined road, the longitudinal acceleration of the sensor is affected by gravity,

$$a_{sensor}(t) = a(t) + g\sin(\theta(t)), \tag{2.2}$$

where a_{sensor} denotes acceleration measured by the accelerometer, g is gravitational acceleration and θ is slope of the road. By substituting $\sin(\theta)$ with a prior known grade map $p(s)$, longitudinal acceleration is calculated as

$$a(t) = a_{sensor}(t) - g\ p(s(t)).$$

Thus, the prediction model is

$$\begin{aligned} s(t+1) &= s(t) + v(t)\ \Delta t \\ v(t+1) &= v(t) + (a_{sensor}(t) - g\ p(s(t)))\Delta t + w(t)\ \Delta t. \end{aligned} \tag{2.3}$$

Measurement Model

The vehicle speed v_m is measured by the wheel speed encoders and the inclination $\theta(t)$ can be constructed by measurements of IMU accelerometer and wheel speed encoder. By reformulating (2.2) the inclination $\theta(t)$ can be indirectly measured as

$$\theta_m = \sin^{-1}\left(\frac{a_{sensor} - \dot{v}}{g}\right), \tag{2.4}$$

where $\dot{v}$ is the vehicle acceleration obtained by differentiating the longitudinal velocity measured from the wheel speed sensor. Measurement noises for both IMU and wheel encoder sensors are assumed to be additive with Gaussian distributions $\eta^v \sim \mathcal{N}(0, \sigma_v^2)$ and $\eta^\theta \sim \mathcal{N}(0, \sigma_\theta^2)$. The measurement covariance matrix is defined as $R = \text{diag}(\sigma_v^2,\ \sigma_\theta^2)$. The measurement model is

$$\begin{aligned} v_m(t) &= v(t) + \eta^v \\ \theta_m(t) &= \sin^{-1}(p(s(t))) + \eta^\theta. \end{aligned} \tag{2.5}$$

The equations (2.3) and (2.5) define the f and h functions in the general formulation (2.1), respectively.

Inertial Sensor Noise Filtering and Bias Removal

Since the acceleration signal measured by IMU is noisy, a low pass filter is used for noise reduction. Also, inertial sensors are subject to bias drift. In the absence of input, the IMU accelerometer reads non-zero output. This bias should be compensated, but the challenge is that bias drifts over time due to temperature changes and other factors. Hence, bias drift should be modeled mathematically and removed from the measurements. In this study it is assumed that the bias drifts linearly as a function of time. Therefore, equation 2.4 is reformulated as

$$\theta_m = \sin^{-1}\left(\frac{a_{sensor} - \dot{v}}{g}\right) - \underbrace{bt}_{\text{bias}}, \tag{2.6}$$

where t is the time and b is the bias parameter. The bias parameter estimation is discussed in more detail in the next section.

2.4 Localization Results

To validate the effectiveness of the proposed localization approach, the algorithm is tested with both simulations and experiments. Assuming that the GPS signal is lost, the localization performance of the proposed method is compared with velocity integration. In simulation, the road altitude map is assumed to be a polynomial function of position as depicted in Figure 2.2a. Measured velocity and inclination signals are generated by assuming Gaussian noise. Figure 2.2 shows the generated maps and signals. The absolute position error e is defined as

$$e = \hat{s} - s,$$

where $\hat{s}$ is the estimated position and s is the true position. In Table 2.1, root-mean-square error (RMSE) values for ten runs of simulation are displayed and their average values are calculated. As seen, the average RMSE using the EKF estimation approach is considerably smaller than using velocity integration. Absolute error between the estimate from velocity integration and the ground truth increases over time while the absolute error between Kalman estimate and ground truth is non-increasing and bounded, according to the simulation results.

The performance of the algorithm has also been examined through an experiment. The experiment was carried out in a hilly road located near University of California Berkeley. The test vehicle is a Hyundai Genesis equipped with OTS (Oxford Technical Solutions) RT2002 sensing system which is comprised of a high precision GPS and an IMU. RT2002 GPS without its base-station GPS receiver provides positioning with 0.6m Circular Error Probable (CEP) accuracy. The car is driven on the road and the position and altitude data measured by the RT2002 GPS system are collected to build a high-resolution road grade map. The obtained map is compared with the map generated by Google elevation API in Figure 2.3.

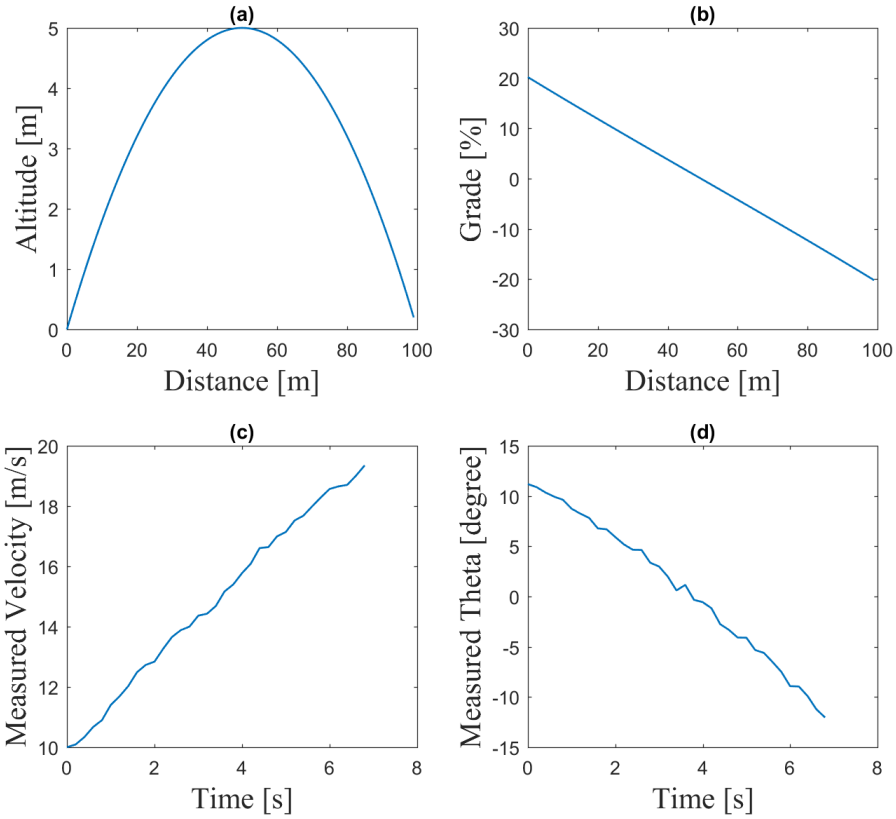


Figure 2.2: Altitude and grade maps as well as velocity and inclination signals are generated to be used by simulator.

Table 2.1: Simulation results: comparison of root-mean-square error

Position RMSE [m]		
Iteration	Velocity Integration	EKF Estimate
1	1.04	0.12
2	1.20	0.19
3	0.50	0.16
4	0.64	0.12
5	0.96	0.12
6	0.70	0.09
7	0.74	0.20
8	0.87	0.14
9	0.83	0.16
10	0.81	0.14
Average	0.83	0.14

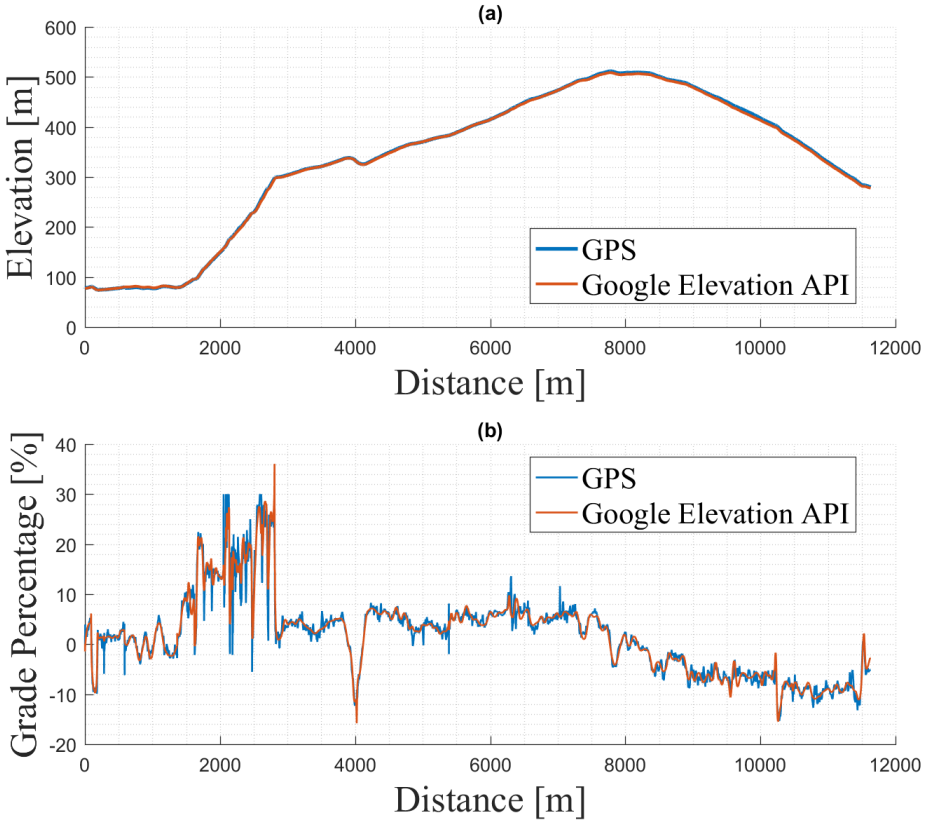


Figure 2.3: Elevation and grade maps are generated using both GPS and Google Elevation API for a hilly route located near UC Berkeley.

To measure the inclination, bias drift is modeled as a linear function of time and removed from the measurement. To determine the mathematical model of the acceleration bias drift, first the road slope profile for a segment of the road is computed using the equation (2.4) with collected acceleration and velocity data. Then the computed slope profile θ_n is compared against the slope profile obtained by Google Elevation API θ_{Google}. The difference between the two profiles is equivalent to the bias term in the equation (2.6),

$$e_{slope}(t) = \theta_n(t) - \theta_{Google}(t) \simeq bt.$$

The bias parameter b is estimated by fitting a linear function to the above slope error profile $e_{slope}(t)$ using least squares. In this study, the high-precision RT2002 IMU is used to obtain the acceleration measurements. However, by accurate modeling of the acceleration bias drift, the vehicle's on-board accelerometer can be used for acceleration measurement.

After generating the high-resolution grade map of the road, the position is estimated using the proposed localization approach. Although GPS data is not used to estimate the position, the data is still collected and used as the ground truth. Three experiments are run

Table 2.2: Experiment results: comparison of root-mean-square error

Position RMSE [m]		
Experiment	Velocity Integration	EKF Estimate
1	25.0	3.7
2	17.9	9.4
3	21.3	4.3
Average	21.4	5.8

Table 2.3: Experiment Results: Comparison of Drift Error of Final Position Estimate

Method	Absolute Error [m]	Error Percentage [%]
Velocity Integration	60.3	0.52
EKF estimate	2.4	0.02

and RMSE for both methods are calculated and reported in Table 2.2. As shown, the EKF algorithm achieves significantly smaller RMSE than velocity integration.

Table 2.3 compares the drift error of the two methods for just the first experiment. As seen, the estimated position calculated by velocity integration drifts over time. The absolute error between the velocity integration estimated position and the ground truth at final point is 30 times larger than EKF estimated position. The experiment results are in agreement with simulation results and EKF estimates are relatively matched with the truth, whereas position estimates calculated by integration of wheel speed diverge as time goes on.

2.5 Control System Architecture

In this section an example application of the proposed localization algorithm is described, a hierarchical control system that exploits position-dependent information for planning and control. The system architecture is shown in Figure 2.4. The goal is to plan an energy-optimal velocity trajectory for a given route offline and to follow the obtained reference trajectory in real-time. Both the offline and online formulations with the vehicle longitudinal model are explained in the following sections.

Vehicle Model

The longitudinal motion of a vehicle moving on an inclined road can be modeled by the following force balance in the longitudinal direction

$$m\dot{v} = \underbrace{F_{traction} - F_{brake}}_{\text{u(t)}} - F_{airdrag} - F_{rolling} - F_{gravity}, \tag{2.7}$$

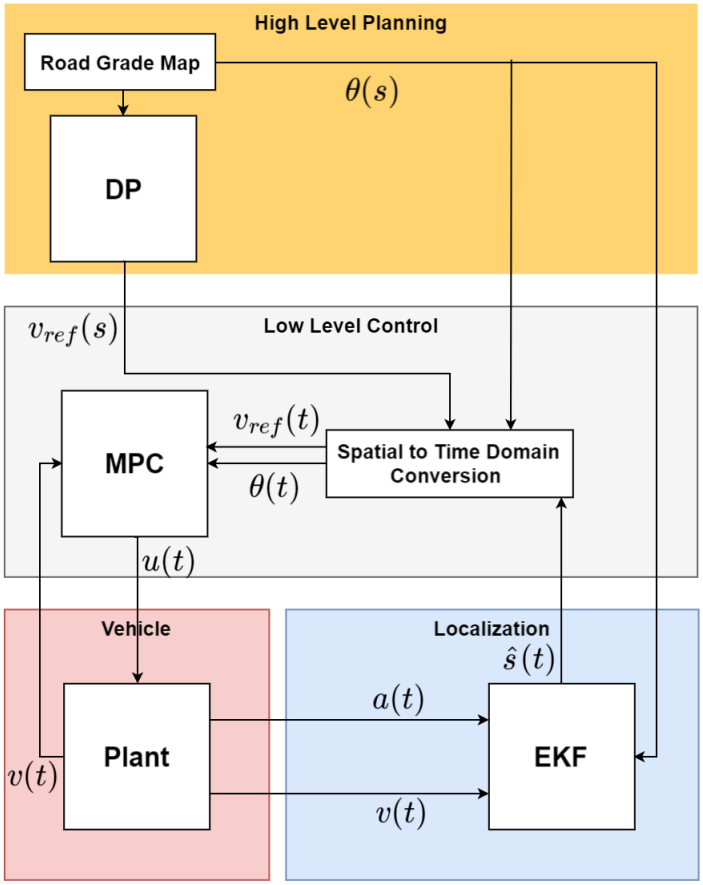


Figure 2.4: System Architecture: position-dependent data including reference velocity $v_{ref}(s)$ and road grade $\theta(s)$ are converted from spatial domain to time domain

where m is mass of the vehicle, $\dot{v}$ is the longitudinal acceleration and $F_{traction}$ and F_{brake} are throttle and brake forces, respectively and $F_{traction} - F_{brake}$ is the total force taken as control input $u(t)$. The aerodynamic drag is determined by vehicle speed v, air density ρ, air drag coefficient C_d and frontal area A_f.

$$F_{airdrag} = \frac{1}{2}\rho C_d A_f v^2.$$

The rolling resistance is defined as

$$F_{rolling} = mgC_r \cos(\theta),$$

where g is gravitational force, C_r is rolling friction coefficient and θ is road slope. The gravity force due to road grade can be expressed as

$$F_{gravity} = mg \sin(\theta).$$

High-Level Planner: Dynamic Programming Formulation

Traffic condition and roadway grade have a significant impact on the vehicle's energy consumption and emission. For a given route with specified origin and destination, the energy-optimal velocity trajectory along the route can be determined by employing information about traffic condition and road grade. To find the optimal position-dependent velocity profile, it is convenient to convert the optimization problem formulation from time domain to spatial domain,

$$\dot{v} = \frac{dv}{dt} = \frac{dv}{ds}\frac{ds}{dt} = \frac{dv}{ds}v.$$

Velocity (v) is a state variable and travelled distance (s) is the independent variable. The vehicle longitudinal dynamics (2.7) in the spatial domain is

$$\frac{dv}{ds} = \frac{1}{mv}\underbrace{(u(s) - F_{airdrag} - F_{rolling} - F_{gravity})}_{F_{total}} \tag{2.8}$$

and after Euler discretization the dynamics is reformulated as

$$v(s+1) = v(s) + \frac{\Delta s}{mv(s)}F_{total},$$

where F_{total} is the total longitudinal force. This formulation is singular when the velocity is zero. To avoid the singularity, by assuming a constant acceleration over the integration interval ds, the longitudinal dynamics 2.8 is discritized as

$$v(s+1) = \big(v(s)^2 + \frac{2\Delta s}{m}F_{total}\big)^{\frac{1}{2}}, \tag{2.9}$$

with trapezoid discretization instead of Euler method.

The goal is to minimize the total power consumption over the entire route within a reasonable trip time interval. Therefore, the cost function is a trade-off between power consumption and trip time. The first term of the cost function (2.10) penalizes the wheel power consumption and the second term compensates deviation from maximum speed profile to guarantee a short trip time. The objective

$$J(s_{cur}) = \sum_{s=s_{cur}}^{s_{final}} \big(v(s)u_p(s) + \gamma(v(s) - v_{max}(s))^2\big)\Delta s \tag{2.10}$$

is minimized, where s_{cur} and s_{final} represent the current and final positions respectively. γ is a weight factor, $u_p(s)$ denotes acceleration force in longitudinal direction and $v_{max}(s)$ is velocity upper bounds at position s. Although $u(s)$ can hold positive or negative values,

$u_p(s)$ in the cost function (2.10) refers to only positive inputs. The problem is formulated as

$$\begin{aligned}
\min_{u(s)} \quad & J = \sum_{s=s_{cur}}^{s_{final}} \big(v(s)u_p(s) + \gamma(v(s) - v_{max}(s))^2\big)\Delta s \\
\text{s.t.} \quad & v(s+1) = f(v(s), u(s)), \\
& v_{min}(s) \leq v(s) \leq v_{max}(s), \\
& u_{min} \leq u(s) \leq u_{max}, \\
& u_p(s) = u(s) \quad \text{if} \quad u(s) > 0 \\
& u_p(s) = 0 \quad \text{if} \quad u(s) \leq 0 \\
& v(0) = 0, \\
& v(s_{final}) = 0,
\end{aligned} \tag{2.11}$$

where the objective (2.10) is minimized while satisfying the longitudinal dynamics (2.8), denoted as f, as well as state and input constraints and boundary conditions. $v_{min}(s)$ and $v_{max}(s)$ are velocity lower and upper bounds at position s and u_{min}, u_{max} are control inputs lower and upper bounds, respectively. The velocities at the origin and destination are assumed to be zero.

The nonlinear optimization problem (2.11) is solved by dynamic programming (DP) with the solver from [67]. The state and input spaces are discretized according to their corresponding upper and lower bounds. To incorporate the effect of traffic condition, the velocity at each step is upper bounded according to the traffic flow data acquired for a specific route. If data is not available, the road speed limit $v_{max}(s)$ is used as upper bound. Road slope information of the route is also included in the vehicle dynamic model (2.9). Thus, the DP algorithm makes use of the road geometry data to find the optimal reference velocity trajectory. Starting from destination, the DP proceeds backward and evaluates the optimal cost-to-go function (2.10) at each node based on Bellman's principle of optimality. The backward sweep outputs a map of optimal cost and optimal policy over the state grid. In the forward sweep, DP starts from the route origin and applies the obtained optimal control policy to calculate the optimal velocity trajectory. The optimized velocity $v(s)$ will be set as the reference velocity $v_{ref}(s)$ for the low-level controller.

Low-Level Controller: MPC Formulation

The low-level controller is a cruise control system that controls the longitudinal dynamics of the vehicle to track the reference velocity generated by the high-level planner and minimize the control effort in real time. To design the controller, a Model Predictive Control (MPC) algorithm is implemented. The presented controller solves the constrained finite horizon

optimization problem

$$\begin{aligned} \min_{u(.|t)} \quad & J = \sum_{k=t}^{t+N} ||v(k|t) - v_{ref}|| \\ & + \gamma \sum_{k=t}^{t+N-1} ||u(k|t)|| + p(v(t+N|t)) \\ \text{s.t.} \quad & v(k+1|t) = f(v(k|t), u(k|t)), \\ & v_{min} \leq v(k|t) \leq v_{max}, \\ & u_{min} \leq u(k|t) \leq u_{max}, \end{aligned} \tag{2.12}$$

where N is the MPC horizon, $v(k|t)$ and $u(k|t)$ are the state variable and control input at step k predicted at time t, respectively, γ is a weight factor, f represents the vehicle longitudinal dynamics (2.7) and v_{ref} is the reference velocity trajectory obtained from the high level planner. A terminal cost p is introduced as a constant large value.

The presented MPC controller extracts grade data from the provided road grade map. It incorporates this prior knowledge to accurately predict the future longitudinal velocity of the vehicle over the horizon and plan accordingly. Both grade and optimal reference velocity are defined as position-dependent profiles for a specific route. In order to employ these data, vehicle has to localize itself and estimate its position with respect to the map. Since the presented MPC operates in the time domain and route information is available in the spatial domain, these forecasts are projected in the time domain assuming constant velocity over the horizon of MPC.

2.6 Control Results

High-level planner

An urban route near UC Berkeley (from UC Berkeley Etcheverry Hall to the Richmond Field Station) is selected and traffic flow velocity as well as grade map are obtained through the HERE API and Google Elevation API, respectively. The traffic flow velocity profile is taken as the upper-bound for velocity. The lower-bound is taken as half of the upper bound. The parameters of the longitudinal dynamic model are shown in Table 2.4. The controller parameters are presented in Table 2.5. Figure 2.5 illustrates the traffic flow velocity profile as well as the velocity profile obtained by trajectory planning optimization carried out via DP. The grade map of the road is also illustrated. Table 2.6 presents energy consumption as well as total trip time for traffic flow velocity profile and optimal velocity profile. As seen, the energy consumption with the optimal velocity profile (for $\gamma = 10$) is 47.8% less than with the traffic flow profile, while the trip time is 26.0% longer.

Table 2.4: Model Parameters

m	vehicle mass	kg	1360
A_f	vehicle frontal surface	m^2	2.30
ρ	air density	kg/m^3	1.225
C_d	vehicle drag coefficient	-	0.24
C_r	vehicle roll coefficient	-	0.01
T_s	sampling time	s	0.2

Table 2.5: Controller Parameters

v_{min}	minimum velocity	m/s	traffic speed lower bound
v_{max}	maximum velocity	m/s	traffic speed upper bound
u_{min}	minimum control input	kN	-3
u_{max}	maximum control input	kN	3

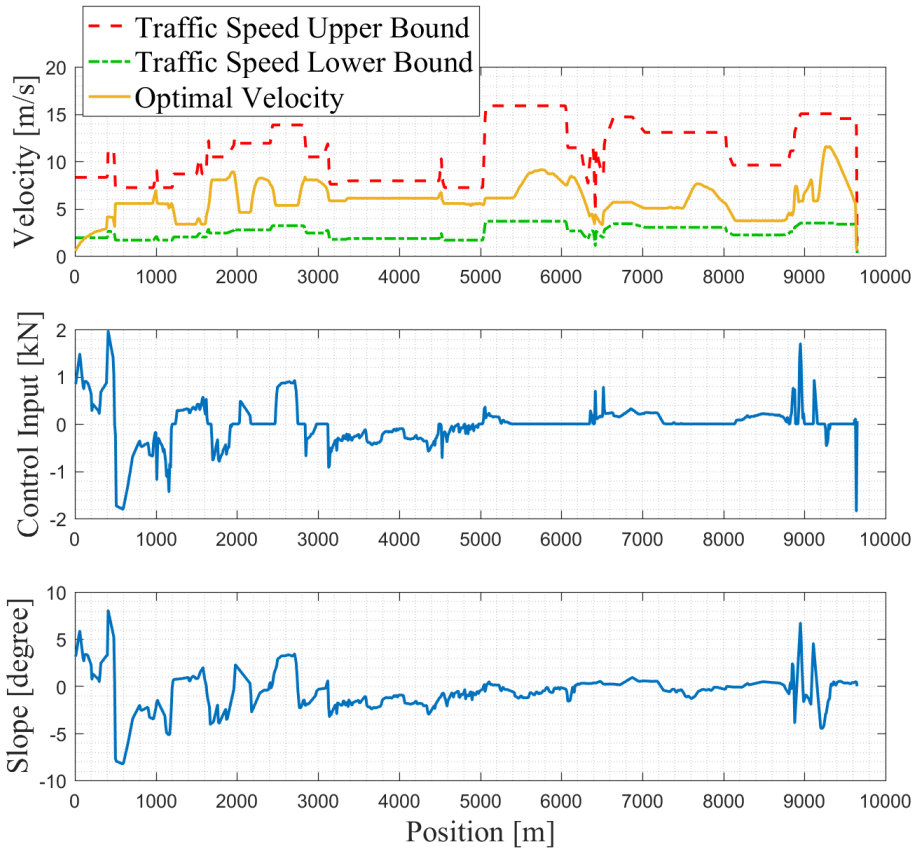


Figure 2.5: Optimal velocity is bounded by traffic speed profile upper and lower bounds and control policy is optimized based on road slope map.

Table 2.6: DP Results: comparison of the two velocity profiles

Velocity Profile	Energy Consumption [kWh]	Trip Time [min]
Traffic Flow	0.9	20.4
Optimal ($\gamma = 10$)	0.47	25.7
Optimal ($\gamma = 0.1$)	0.4	31.4
Improvement [%]($\gamma = 10$)	47.8 %	-26.0 %
Improvement [%] ($\gamma = 0.1$)	55.5 %	-53.9 %

Table 2.7: Energy Consumption Comparison

Position	Energy Consumption [kWh]
True Position	0.20
60 meters ahead	0.23
0.5% error	15%

Low-Level Controller

The optimization problem (2.12) is formulated in YALMIP [45]. MPC parameters are the same as DP parameters presented in Table 2.5 and the MPC horizon is selected as 5. Figure 2.6 shows the simulation results of the MPC controller in closed-loop with the vehicle longitudinal dynamics model. As illustrated, the controller tracks the optimal reference velocity calculated in previous section. At the same time it minimizes the control effort by employing the road prior grade knowledge.

Now by assuming 0.5% of localization error based on the results in the table 2.3, after about less than 12 km of traveling, the position estimate can be off by 60 meters. If the controller extracts position-dependent data including optimal reference velocity and road grade associated with incorrect position, the energy efficiency can significantly be affected by this error. As an example, a portion of the road is selected shown in Figure 2.6 between position 8500 m to 9500 m and assumed that the controller employs knowledge of 60 meters ahead instead of its true position. Table 2.7 compares the energy consumption for both cases. The results show that energy efficiency can be affected by about 15% in case of a 0.5% localization error.

2.7 Conclusion

A localization method is presented for autonomous driving in GPS-denied areas using a prior grade map. According to the results when there is no GPS signal, vehicle can use its own on-board sensors and a prior road grade map to localize itself relative to the map and correct its position estimate. Furthermore, an energy-efficient control system is developed

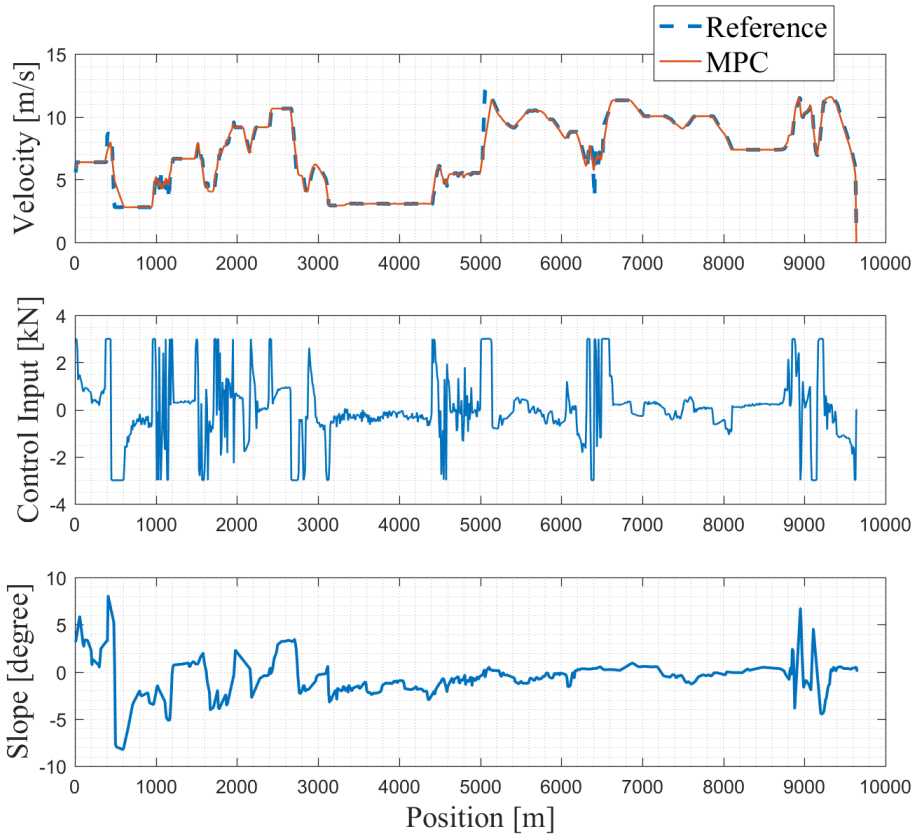


Figure 2.6: MPC controller tracks the optimal reference velocity trajectory obtained from high-level planner and simultaneously minimizes the control input by incorporating road grade data in its short-term planning.

by exploiting traffic information as well as road slope and speed limit data. It is verified that errors in localization can significantly impact energy efficiency.

Chapter 3

Leader-Follower Coordination

3.1 Introduction

Connected and automated vehicles will enhance mobility and safety by integrating autonomous driving technologies with connectivity. On the other hand recent advances in autonomous driving have accelerated the need for high performance and reliable Advanced Driving Assistance Systems (ADAS) which guarantee safety and comfort in various driving conditions. Connectivity enables ADAS to leverage Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communication to further improve performance [31].

Adaptive cruise control (ACC) is a widely used ADAS module that controls the vehicle longitudinal dynamics. ACC is triggered once a preceding vehicle is detected within a certain distance range from the ego vehicle (follower). It automatically adjusts the vehicle acceleration and deceleration to maintain a proper minimum safe distance from the vehicle ahead (leader). ACC enhances mobility, improves safety and comfort, and reduces energy consumption. The design of ACC based on Model Predictive Control (MPC) is common in the literature. For example, in [48, 60, 19, 20, 69, 44], the authors present ACC design using MPC based on one or more particular performance criteria including traffic flow, energy efficiency and safety or comfort considerations.

Conventional adaptive cruise control systems operate in two modes: ACC and Cruise Control (CC), depending on the presence or absence of the lead vehicle in detection range of the ego vehicle. In ACC mode the objective is to maintain a safe distance from the lead car, whereas in CC mode the objective is to track the reference velocity set by the driver or the maximum speed limit of the road. In the literature, ACC and CC modes are also referred to as distance tracking and velocity tracking modes, respectively. Discrete switching between the modes may result in aggressive control action or repetitive mode change. In [63, 81], the authors suggest more sophisticated rule-based switching strategies to prevent chattering caused by switching. The presented controller is formulated and designed without employing any switch in order to adapt the velocity and distance automatically and smoothly, if a lead car is detected.

In conventional ACC systems, the desired inter-vehicle gap can be specified as a constant space or time gap between the vehicles. The desired gap also might be defined via a constant time headway t_h policy which relates the inter-vehicle desired distance Δs_{des} to the current velocity v of the ego vehicle as $\Delta s_{des} = \Delta s + t_h v$, where Δs is the static gap or the distance between the cars when they are at standstill and $t_h v$ is the dynamic gap which changes with velocity. In general these simple approaches do not guarantee safety. To guarantee safety at all times, the theory of invariant sets can be used to compute the safe distance. In [2], reachability analysis with level set method is proposed to compute the safe set for heavy duty platooning. The authors of [42] use a kinematic model to describe the system as a linear time-invariant (LTI) system and apply backward reachable set analysis to calculate a polytopic control invariant set for ACC. In [70], a control invariant set is presented as the safe set for platooning by considering the lower and upper bounds on the acceleration. The aforementioned studies compute the control invariant set by bounding the unknown parameters such as road slope or the front car's motion. However, computing the control invariant safe set by considering the lower and upper bounds on the unknown parameters can lead to an overly conservative safe set and thus an undesirable large gap between the vehicles. In the presented study, instead of bounding these parameters, the availability of road grade preview from a high fidelity map and the availability of lead car's future state trajectory from a V2V communication device, are assumed. Employing these parameters for computing the control invariant safe set, instead of their boundary values, yields a less conservative safe set and consequently a closer inter-vehicle safe distance.

Road grade can considerably affect the ACC controller performance. The work in [44] considers an ACC which takes into account road elevation data to improve energy efficiency. In [49] real-time estimation of road grade and vehicle mass are utilized to improve comfort. This study investigates how the ACC controller performance improves in terms of safety, comfort, tracking accuracy and energy efficiency, by exploiting the road grade knowledge.

The contribution of this chapter can be summarized as follows:

- The design of an ACC controller is presented, which incorporates the road grade preview information using a prior grade map of the road to predict the vehicles' longitudinal dynamics. Furthermore, the presented controller is formulated to switch automatically and smoothly between distance and velocity tracking.
- A numerical approach to compute a control invariant set which makes use of the road grade preview and the future state trajectory of the leading car, transmitted through V2V communication, is presented. This set is incorporated as terminal constraint of the ACC controller to guarantee safety.
- It is demonstrated through two example scenarios that the presented approach is more efficient compared to the controller that does not use grade preview.

The design of a safe Adaptive Cruise Control (ACC) is presented, which uses road grade and lead vehicle motion preview. The ACC controller is designed by using a Model Predictive Control (MPC) framework to optimize comfort, safety, energy-efficiency and speed

tracking accuracy. Safety is achieved by computing a robust invariant terminal set. This chapter presents a novel approach to compute such set which is less conservative than existing methods. The presented controller ensures safe inter-vehicle spacing at all times despite changes in the road grade and uncertainty in the predicted motion of the lead vehicle. Simulation results compare the presented controller with a controller that does not incorporate prior grade knowledge on two scenarios including car-following and autonomous intersection crossing. The results demonstrate the effectiveness of the presented control algorithm.

The rest of the chapter is structured as follows: Section 3.2 introduces some preliminaries about vehicle model and road grade map generation. Section 3.3 describes the problem statement and the MPC formulation. Section 3.4 explains the design of control invariant set. Section 3.5 describes car-following and autonomous intersection passing scenarios as two example applications of the proposed approach. Section 3.6 illustrates the simulation results and finally Section 3.7 makes the concluding remarks.

3.2 Preliminaries

Vehicle Model

The vehicle is modeled as a point mass moving along a road. The system state at time t is

$$x(t) = [s(t)\ v(t)]^T,$$

where $s(t)$ and $v(t)$ are the vehicle position and velocity, respectively. The longitudinal motion of the vehicle shown in Fig. 3.1 can be described by the following equations

$$\begin{aligned} \dot{s} &= v \\ \dot{v} &= \frac{1}{m}(\underbrace{F_t - F_b - F_a - F_r - F_g}_{\mathrm{F}_{total}}), \end{aligned} \tag{3.1}$$

where F_t and F_b are traction and braking forces, respectively and F_{total} is the total longitudinal force. The aerodynamic drag is determined by vehicle speed v, air density ρ, air drag coefficient C_d and frontal area A_f.

$$F_a = \frac{1}{2}\rho C_d A_f v^2. \tag{3.2}$$

The rolling resistance is defined as

$$F_r = mgC_r \cos\theta, \tag{3.3}$$

where g is gravitational force and C_r is rolling friction coefficient. The gravity force due to road slope can be expressed as

$$F_g = mg \sin\theta. \tag{3.4}$$

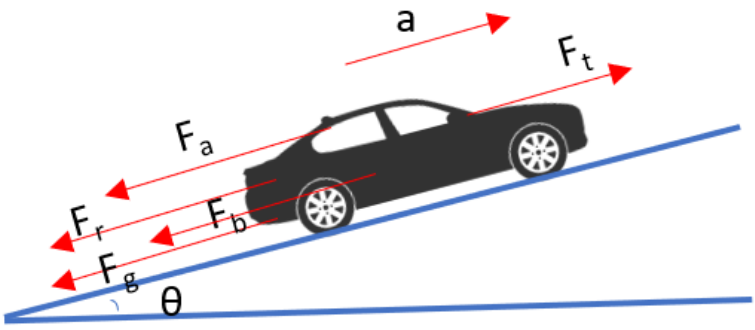


Figure 3.1: The forces exerted in the direction of motion (longitudinal direction) on a vehicle traveling on an inclined road are shown.

Table 3.1: Vehicle Model Parameters

m	vehicle mass	kg	2278
A_f	vehicle frontal surface	m^2	2.63
ρ	air density	kg/m^3	1.206
C_d	vehicle drag coefficient	-	0.2791
C_r	vehicle roll coefficient	-	0.0089
Δt	sampling time	s	0.5

The system (3.1) is discretized using Euler method with constant sampling time interval Δt.

$$\begin{aligned} s(t+1) &= s(t) + v(t)\Delta t, \\ v(t+1) &= v(t) + \frac{\Delta t}{m} F_{total} \end{aligned} \tag{3.5}$$

The parameters of model (3.5) are then estimated by nonlinear least squares from data collected on a test vehicle, at various speeds and accelerations. All the data are collected in a proving ground on level roads with homogeneous surfaces, i.e. $\theta = 0$ and C_r is constant for the purpose of model identification. Fig. 3.2 shows the fit of the identified model to the experimental data, while Table 3.1 summarizes the identified model parameters. Specifically, 7 datasets are used for a total duration of about 16 minutes; the predicted data in Fig. 3.2 are generated simulating model (3.5) with the identified parameters and the measured input torque from the initial condition to the end of the dataset.

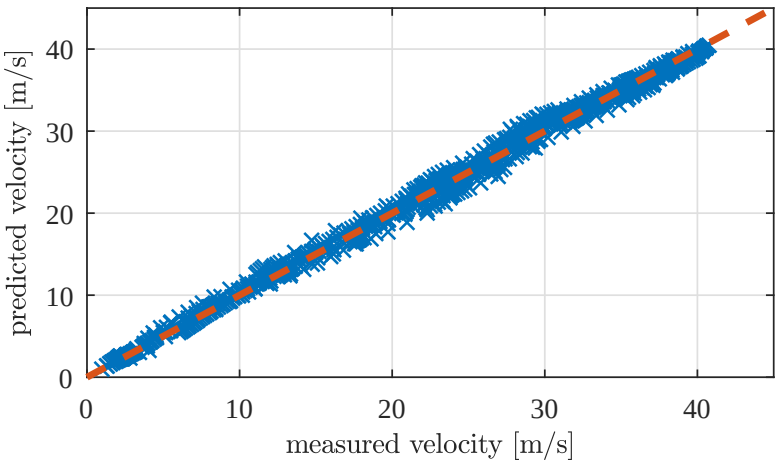


Figure 3.2: Fit of the identified longitudinal dynamics model to the experimental data.

Road Grade Map Generation

Road topology data is a valuable source of information in autonomous driving applications. On roads with up-down slopes, ADAS systems like ACC benefit from accurate prior-known road grade. ACC can exploit the grade preview extracted from a map to ensure keeping a safe distance from the lead vehicle. Different methods for generating a high-resolution grade map of the road have been discussed in the previous work [28]. The road grade profile can be obtained using a survey vehicle equipped with high-precision GPS system. An alternative approach is using APIs like Google Elevation API. Google Elevation API provides elevation data for all locations on earth surface. Elevation data can be queried for specific coordinates to generate road elevation profile. In this study, Google Elevation API is utilized to generate the road grade profile.

3.3 Problem definition and formulation

The goal is to design an ACC and CC controller with safety guarantees. The presented ACC/CC controller is formulated as Model Predictive Control (MPC) which repeatedly

solves the following constrained finite horizon nonlinear optimization problem:

$$\underset{U_{t\to t+N|t}}{\text{minimize}} \quad J = Q \sum_{k=t}^{t+N-1} ||v(k|t) - v_{\text{ref}}||_2 \tag{3.6a}$$

$$+ R_u \sum_{k=t}^{t+N-1} ||u(k|t)||_2 \tag{3.6b}$$

$$+ R_{\Delta u} \sum_{k=t}^{t+N-1} ||\Delta u(k|t)||_2 \tag{3.6c}$$

$$+ P||v(t+N|t) - v_{\text{ref}}||_2 \tag{3.6d}$$

subject to

$$x(k+1|t) = f(x(k|t), u(k|t), \theta(k|t)), \tag{3.6e}$$

$$v_{\min} \leq v(k|t) \leq v_{\max}, \tag{3.6f}$$

$$u_{\min} \leq u(k|t) \leq u_{\max}, \tag{3.6g}$$

$$x(t|t) = x(t), \tag{3.6h}$$

$$d_{\text{safe}}\big(s_{\text{lead}}(k|t) - s(k|t), v(k|t)$$
$$v_{\text{lead}}(k|t), (\theta(t|t), ..., \theta(t_{\text{stop}}|t)\big) \tag{3.6i}$$

$$\leq s_{\text{lead}}(k|t) - s(k|t), \tag{3.6j}$$

$$k = t, ..., t+N,$$

where $x(k|t) = [s(k|t) \quad v(k|t)]^T$ and $u(k|t)$ are the state and control input at step k predicted at time t, respectively. The MPC horizon is N and $U_{t\to t+N|t}$ denotes the sequence of control inputs $\{u(t|t), ..., u(t+N-1|t)\}$. The multi-objective quadratic cost function J represents the trade-off between minimizing reference tracking error (3.6a), control effort (3.6b) and jerk (3.6c) and Q, R_u and $R_{\Delta u}$ are their corresponding weight factors, respectively. The terminal cost (3.6d) is weighted by P. The state and input are constrained as (3.6f) and (3.6g) to lie within their lower and upper bounds denoted as $v_{\min}$, $u_{\min}$ and $v_{\max}$, $u_{\max}$, respectively. The reference velocity v_{ref} is the desired velocity set for the ego vehicle by cruise control. The vehicle longitudinal dynamics (3.5), denoted as f, introduce nonlinear constraints, parameterized by the road grade θ. Since the road grade map is available as position-dependent data, the controller has to convert it from spatial domain to time domain to make use of the grade preview. To do so, the ego car's velocity is assumed to remain constant over the MPC horizon and grade data corresponding to the predicted positions is extracted. The lead car's velocity and position at step k predicted at time t are denoted as $v_{\text{lead}}(k|t)$ and $s_{\text{lead}}(k|t)$, respectively. The predicted future trajectory of the lead car is transmitted through V2V communication. The safe distance between the two vehicles, denoted as d_{safe}, is obtained by calculating the control invariant set, described in Section 3.4. d_{safe} is a function of the position and velocity of the ego and lead vehicles, and of the road grade from the current time t till t_{stop}, which is the time at which the lead car comes to full

stop if it exert its maximum braking force. The optimal solution of the problem (3.6) is

$$U^*(t) = \{u^*(t|t), ..., u^*(t+N-1|t)\},$$

and the receding horizon control law is obtained by applying the first control input

$$u_{\text{MPC}}(t) = u^*(t|t). \tag{3.7}$$

The above formulation includes switching behavior from distance tracking (ACC) mode to reference velocity tracking (CC) mode implicitly. In other words, there is no explicit term for discrete switching between the two modes, but the controller is designed to operate in ACC mode as long as the front vehicle is detected and automatically and smoothly adapts the velocity to CC reference once there is no vehicle in the front.

The multi-objective cost J in problem (3.6) simultaneously fulfills multiple performance criteria. To assess the performance of the controller a performance index is introduced for each objective. The first term of the cost function (3.6a) denotes reference tracking error and *tracking performance index* can be defined as

$$\text{Tracking Performance Index} = \sum_{t=0}^{T} |v(t) - v_{\text{ref}}|, \tag{3.8}$$

where T is the total time of simulation or experiment, $v(t)$ is the ego vehicle velocity at time t obtained as the closed-loop state of system (3.5) controlled with u_{MPC} described in (3.7). The second term of the cost (3.6b) penalizes the control effort and presents energy consumption criteria. The *energy consumption performance index* is introduced as

$$\text{Energy Performance Index} = \sum_{t=0}^{T} \max(0, u_{\text{MPC}}(t)), \tag{3.9}$$

assuming no penalty for braking. The third term in the cost (3.6c) minimizes the change of acceleration (jerk) which is a performance criteria for longitudinal ride comfort. The *comfort performance index* is

$$\text{Comfort Performance Index} = \sum_{t=0}^{T} |u_{\text{MPC}}(t+1) - u_{\text{MPC}}(t)|. \tag{3.10}$$

In both (3.9) and (3.10), $u_{\text{MPC}}(t)$ is the closed-loop control input described in (3.7).

3.4 Control Invariant Set

The adaptive cruise control is considered to be safe if the follower vehicle can avoid collisions with the front vehicle regardless of the front vehicle action. Consider the combined model

of the front and follower vehicles in Adaptive Cruise Control (ACC) problem,

$$\begin{bmatrix} \dot{s}_l - \dot{s}_e \\ \dot{v}_e \\ \dot{v}_l \end{bmatrix} = \begin{bmatrix} v_l - v_e \\ \frac{1}{m_e}\big(F_{t,e} - F_{b,e} - F_{a,e} - F_{r,e} - F_{g,e}\big) \\ \frac{1}{m_l} F_{total,lead} \end{bmatrix} \tag{3.11}$$

$F_{t,e}$, $F_{b,e}$, $F_{a,e}$, $F_{r,e}$ and $F_{g,e}$ denote the ego vehicle tractive force, braking force, aerodynamic drag force, rolling resistance force, and gravity force respectively defined in (3.2)-(3.4). m_e and m_l are the ego and lead vehicle masses, respectively. v_e and s_e are the ego vehicle velocity and position. v_l and s_l are the lead vehicle velocity and position. In this system, the ego (follower) vehicle braking/tractive force is the control input and the lead (front) vehicle's total longitudinal force denoted as $F_{total,Lead}$ is treated as disturbance to the system (3.11) and described based on (3.1). The set of all admissible states for this system are defined as follows:

$$\mathcal{X} = \{[s_l - s_e \; v_e \; v_l \,]^T : \; (s_l - s_e) > l_{min}, \; v_e > 0, \; v_l > 0\}$$

where l_{min} is the minimum required distance between the mass center of the two vehicles. The robust control invariant set $\mathcal{C} \subseteq \mathcal{X}$ is defined as a set with the following property:

$$\begin{aligned} &\text{if } x(t) \in \mathcal{C} \Longrightarrow \exists u(t) \in \mathcal{U} \text{ such that} \\ &f(x(t), u(t), \theta(t)) \in \mathcal{C}, \\ &\forall F_{total,Lead}(t) \in \mathcal{F}_{total,Lead}, \;\; , \; \forall t \in \mathcal{N}^+. \end{aligned}$$

The set $\mathcal{C}$ is a function of the road slope $\theta(\cdot)$ at time t until the time t_{stop} at which the front car comes to full stop after exerting the brake at time t. Therefore, $\mathcal{C} = \mathcal{C}(\theta(t), ..., \theta(t+t_{stop}))$. $x(t)$ denotes the state of the system (3.11) at time t, $\mathcal{U}$ is the input feasible set (3.6g), f represents the system dynamics (3.11). $\mathcal{F}_{total,Lead}$ is the set of all possible longitudinal forces of the lead vehicle. Hence, the robust control invariant set $\mathcal{C}$ for the above ACC system is such that for any maneuver of the front vehicle, there is a control signal that keeps the system (3.11) within $\mathcal{C}$ for all future times [7].

A conservative estimation of the closed form of $\mathcal{C}$ for this problem is presented in [70] that computes the control invariant set by finding the bounds on the road grade and calculating the bounds on accelerations of the lead car. In this work a numerical method is employed to compute the boundary of the safe set more accurately using the road slope preview and future trajectory of the lead car. By using such preview information, the exact maneuver of the front car is calculated by assuming it exerts maximum braking force at each time step, using a two-step approach.

- In the first step, the equations of motion of the front vehicle are integrated using forward Euler discretization, assuming the front vehicle applies its maximum braking

force, $F_{b,l}^{\max}$,

$$v_l(k+1) = v_l(k) + \frac{dt}{m_l}\Bigg(- F_{b,l}^{\max} - \frac{1}{2}\rho C_{d,l} A_{f,l} v(k)^2 - m_l g C_{r,l} \cos\Big(\theta\big(s_l(k)\big)\Big) - m_l g \sin\Big(\theta\big(s_l(k)\big)\Big)\Bigg), \tag{3.12a}$$

$$s_l(k+1) = s_l(k) + dt v_l(k) \tag{3.12b}$$

where the initial condition for (3.12a) and (3.12b) is the current velocity and position of the front vehicle. In (3.12a) the road grade depends on the position of the front vehicle at each time instant. k is the integration interval. The integration stops when the front vehicle reaches zero velocity, $v_l(k) = 0$. Define the position when the front vehicle reaches $v_l = 0$ as $s_l^{stop} = s_l(k)$.

- In the second step, the equations of motion of the following vehicle are integrated backwards in time, assuming that the following vehicle is also applying its maximum braking force, $F_{b,e}^{max}$, with the initial condition of $v_e(0) = 0$ and $s_e(0) = s_l^{stop} - l_{min}$. The integration continues till the velocity reaches its upper bound, $v_e(k) = v_{max}$.

$$v_e(k-1) = v_e(k) - \frac{dt}{m_e}\Bigg(- F_{b,e}^{\max} - \frac{1}{2}\rho C_{d,e} A_{f,e} v_e(k)^2 - m_e g C_{r,e} \cos\Big(\theta\big(s_e(k)\big)\Big) - m_e g \sin\Big(\theta\big(s_e(k)\big)\Big)\Bigg) \tag{3.13a}$$

$$s_e(k-1) = s_e(k) - dt v_e(k) \tag{3.13b}$$

This method, although computationally more expensive compared to the closed form, takes into account the exact road grade profile. The minimum safe distance at each step time of the above integration is computed as follows,

$$d_{safe}^{min}(k) = s_l(0) - s_e(k). \tag{3.14}$$

Forward and backward integration steps are illustrated in Fig. 3.3. Note (3.12b) and (3.13b) require the backward and forward simulation of absolute vehicle positions. One can rewrite the dynamics to highlight that only $\Delta s = s_l - s_e$ is important, as expressed in (3.11).

Fig. 3.4 shows examples of the safe set boundary for various velocity values of the front car. The safe set boundary is obtained by fitting a second-degree polynomial on data points calculated using the above two-step approach. Sufficiently small integration interval increases the data points and results in achieving higher fitting accuracy. The polynomial as the function of ego vehicle velocity defines the required minimum safe distance which is imposed

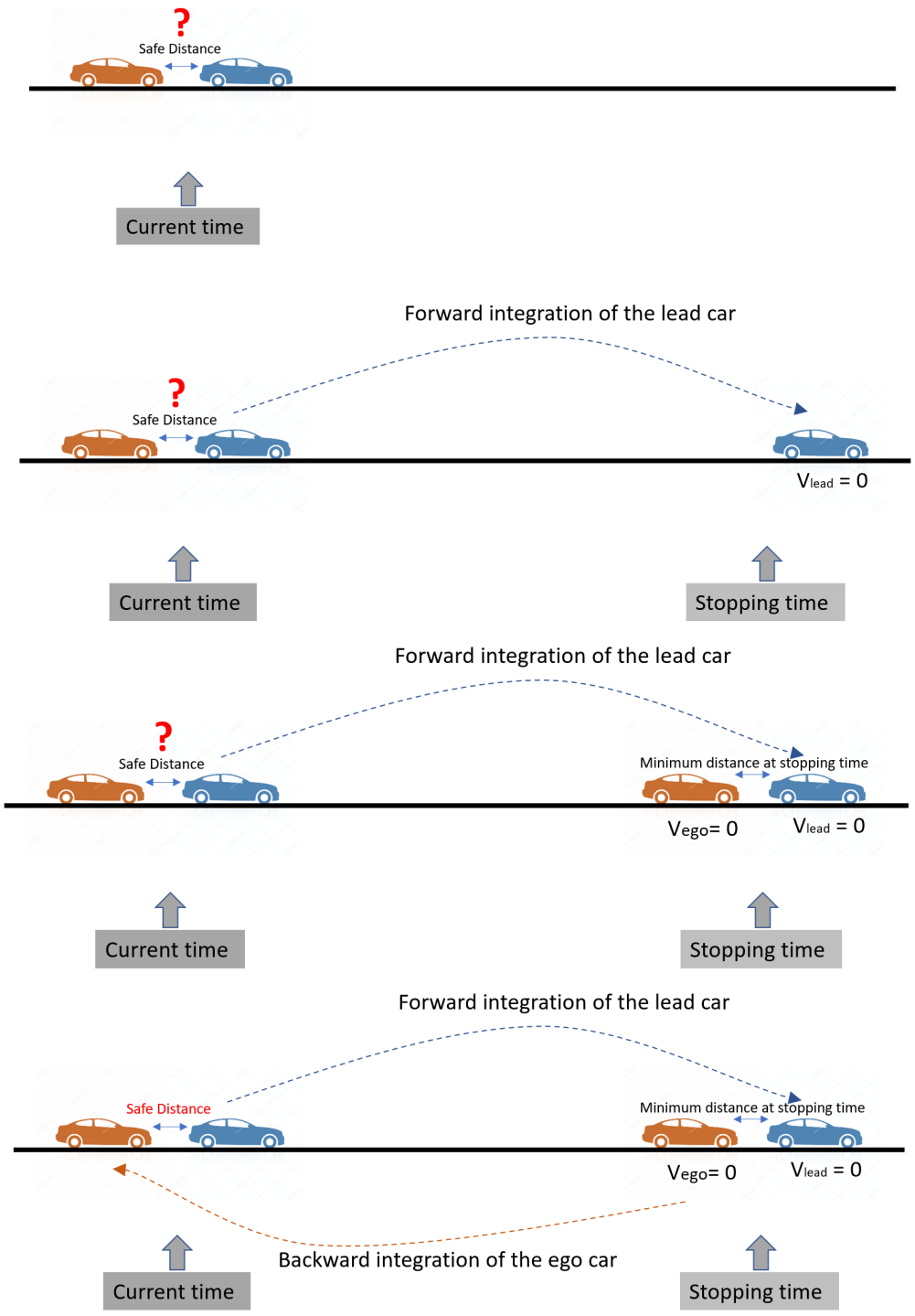


Figure 3.3: Safe set computation.

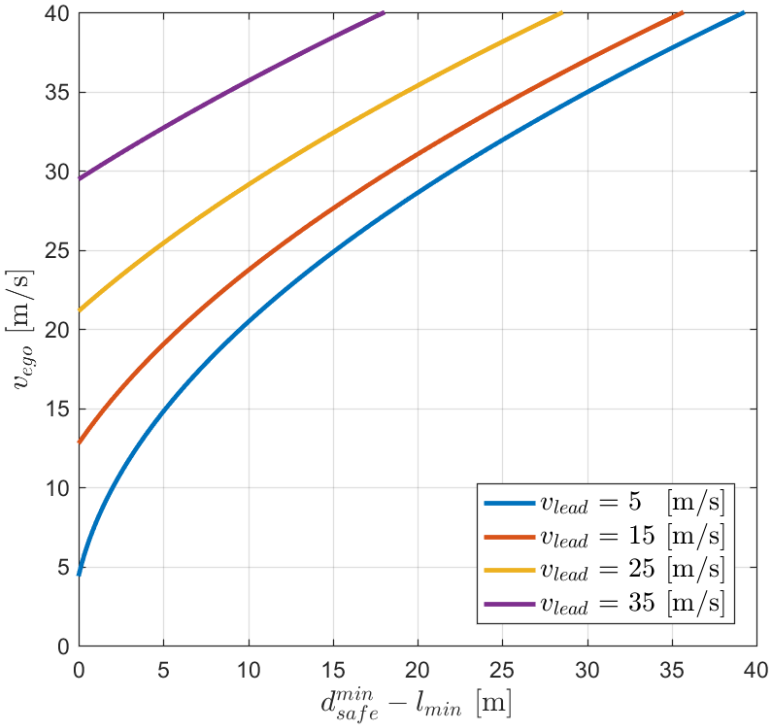


Figure 3.4: The boundary of control invariant set for the ACC problem.

as safety constraint (3.6j) in the MPC formulation (3.6). Also Fig. 3.5 demonstrates the safe time gap plot corresponding to the Fig. 3.4 safe distances. The time gap is computed for the cases that ego and lead vehicles' velocity are the same.

To guarantee feasibility of problem (3.6) at all times, persistent feasibility should be proven by showing the existence of a feasible control sequence at all times when starting from a feasible initial point. Assume that at time t_0 a lead vehicle be in front of the follower vehicle and the problem (3.6) be feasible. Let $x(t)$ be the state of the system (3.1) in closed-loop with the MPC controller (3.6) at $t > t_0$. Since the problem is feasible at $x(0)$, there exist an optimal control sequence $\{u_0^*, u_1^*, ..., u_{N-1}^*\}$ at t_0. Apply u_0^* and let the system evolve to $x(1)$. At $x(1)$, apply u_{min} at the end of the MPC horizon. The control sequence $\{u_1^*, u_2^*, ..., u_{min}\}$, since u_{min} is input feasible and state feasible. In fact, from the construction of the safe set, by applying u_{min} at step N will guarantee that $x(N+1)$ will be at a safe distance $d_{\text{safe}}(N+1)$. In conclusion, the closed-loop system is persistently feasible.

3.5 Example Application Scenarios

The application of an ACC controller that adapts the vehicle's longitudinal velocity based on the other vehicle's states, communicated over V2V network, is not restricted to the

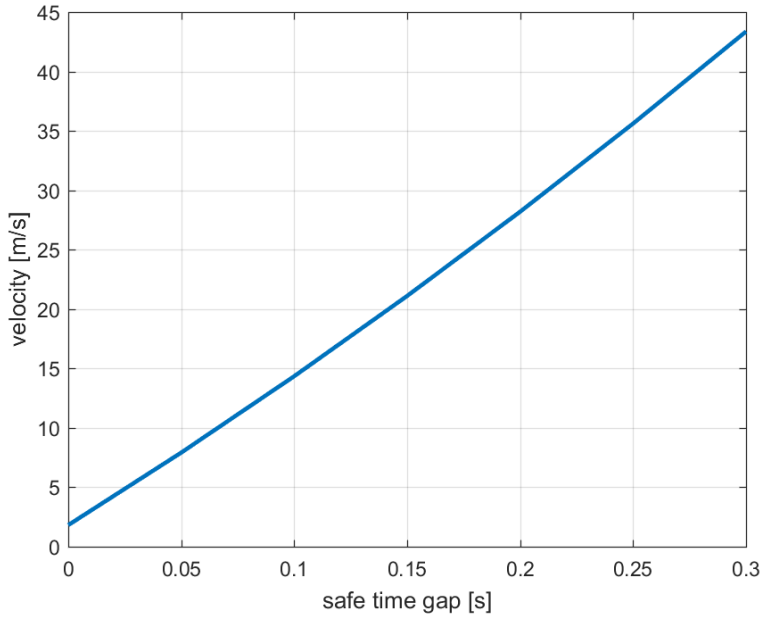


Figure 3.5: Safe time gap corresponding to the safe minimum required distances reported at Fig. 3.4.

car-following scenario. The same formulation of ACC can be extended further to other applications like autonomous intersection crossing. In an uncontrolled intersection, two cars that simultaneously approach an intersection at crossing directions, communicate with each other and adapt their speed to avoid collision and pass the intersection safely and efficiently [57]. A possible approach to coordinate the vehicles in an autonomous intersection is to define a circle centered at the center of the intersection and with radius of the range of communication R, as shown in Fig. 3.6(a). Once the cars enter this virtual circle, they communicate with each other and assign the priority to each other based on their current velocity. Priority assignment for autonomous intersection is not the focus of this work and has been discussed in [57]. It is assumed to be known that which car is prioritized to pass through the intersection first and play the role of the leader for the other car. The car with lower priority is the one that runs the ACC controller and adapts its velocity and distance to the intersection corresponding to the lead car. At each time step, the lead car in the crossing direction is projected in front of the ego car, as shown in Fig. 3.6(b), as a virtual car that the ego vehicle follows.

3.6 Simulation Results and Discussion

The optimization problem (3.6) is modeled using YALMIP [45] and solved using IPOPT. The vehicle model and MPC parameters are presented in Table 3.1 and 3.2, respectively. To

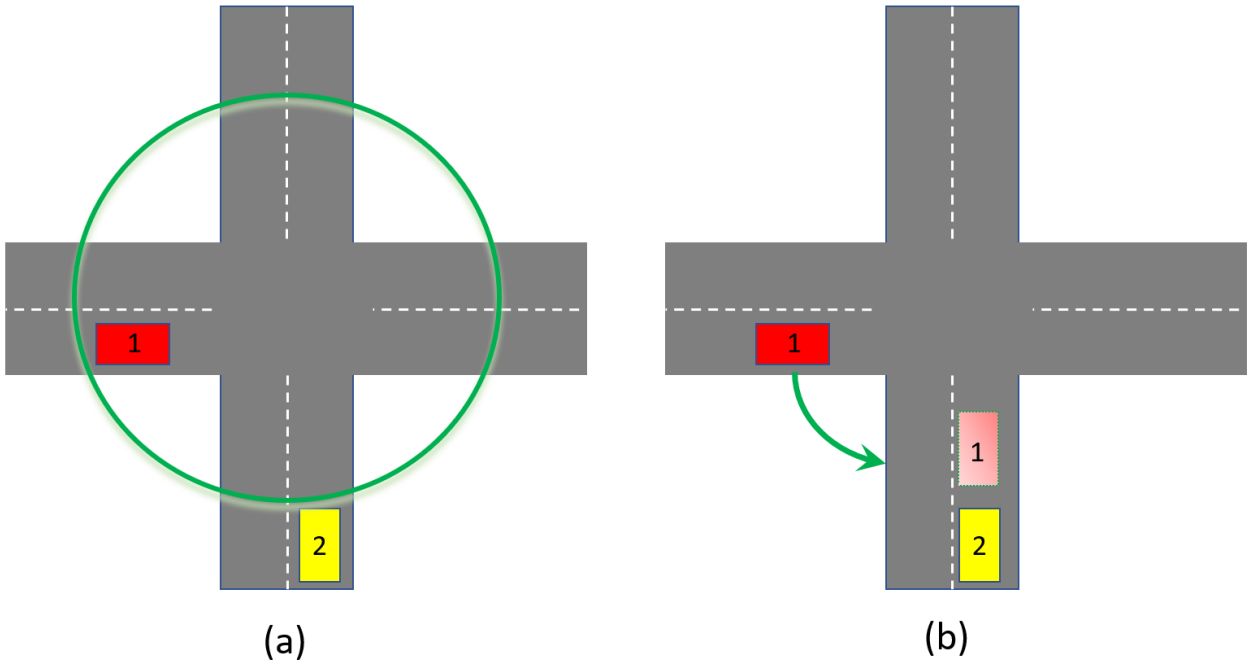


Figure 3.6: a) The radius of green circle around the intersection represents the range of V2V communication. Since the red car has entered the circle sooner than the yellow one, the red car is prioritized to pass the intersection. So the red car is the leader and the yellow car is the follower. The yellow car has to adapt its speed according to the leader car speed and pass through the intersection with second priority. b) The lead car (red) is projected at each time step in front of the follower car (yellow) which has the second priority. Light red rectangle shows the virtual car in front of the yellow car.

Table 3.2: MPC Controller Parameters

$v_{\min}$	minimum velocity	m/s	0
$v_{\max}$	maximum velocity	m/s	30
$u_{\min}$	minimum control input	kN	-3
$u_{\max}$	maximum control input	kN	3
Δt	sampling time	s	0.2
Q	tracking error cost weight	-	10
R	control effort cost weight	-	1
$R_{\Delta u}$	jerk cost weight	-	10
P	terminal cost weight	-	100

conduct realistic simulations, a car is driven on roads with significant slope (located near UC Berkeley) and collected latitude, longitude and velocity data. The grade maps of the roads is generated by querying the Google Elevation API, and obtaining elevation data for the collected latitudes and longitudes. The recorded velocity data of the car is considered as the front car's velocity in the simulations. The simulation tests are run with different real roads grade profiles and the performance of the presented approach is assessed against the baseline approach. The presented ACC considers the road grade preview in both planning and safe set calculation, while the baseline controller does not include knowledge of grade.

Fig. 3.7 represents the results for a car-following scenario. The first graph compares the velocity of the front and the ego car; the tracking accuracy with and without knowledge of grade is essentially the same. The second graph depicts the relative distance between the ego and front vehicles, with and without grade knowledge. The safe distance in the graph represents the minimum required distance between the vehicles calculated by the robust control invariant set. The relative distance obtained by the presented controller is lower-bounded by the safe distance. However, the relative distance obtained by the baseline controller (that has no knowledge of the road grade) is not safe, since the relative distance violates the minimum required safe distance. The third graph represents the control input u_{MPC}. As seen, the control input obtained by the proposed controller is much smoother compared to the baseline controller. The fourth graph depicts the road grade profile obtained by Google Elevation API.

To assess the performance of both controllers quantitatively, the performance indexes are introduced in Section 3.3. Table 3.3 compares, for various simulation runs, the energy consumption, tracking and comfort performance indexes when the road grade preview is either known or not known. The total cost, which is the sum of all performance indexes, is also reported; the average total cost without grade knowledge is considerably higher than with grade preview.

For the autonomous intersection scenario the results are similar to the car-following case, and are not reported here for brevity. In this scenario, the road grade profiles for the leader and follower cars are different; at each time step the vehicles' distances to the center of intersection are measured and their relative distance is calculated by projecting the lead car in front of the follower car.

Both scenarios (car-following and autonomous intersection crossing) are set up in the PreScan simulation environment. PreScan has an interface with MATLAB/Simulink and is a suitable platform for developing ADAS systems as well as modeling V2V communication. The road grade for both scenarios is modeled in PreScan environment with sinusoidally varying profiles. The lead vehicle velocity is also generated as another sinusoidal profile. The PreScan video for both scenarios is available online.[1] In the car-following video the lead car is moving with a sinusoidal velocity profile, until it applies full braking and comes to full stop in the middle of the road. The follower car is able to maintain the safe distance and avoid collision using the presented approach. This is a visualization that shows how the control

[1] https://www.youtube.com/watch?v=Qi9Lehtvqjc

Segment	Energy Consumption		Tracking		Comfort		Total Cost	
	W/ grade	W/O grade	W/ grade	W/O grade	W/ grade	W/O grade	W/ grade	W/O grade
1	311.9	324.3	68.5	127.2	49.0	50.0	430.3	501.5
2	615.1	625.2	47.1	64.2	60.9	86.5	723.1	775.9
3	0	0.81	28.7	30.9	30.8	189.7	59.6	221.4
4	0	0	45.0	37.0	52.3	205.6	97.3	242.6
5	0	0.11	32.6	30.6	36.2	286.4	68.8	317.1
6	149.1	158.0	70.2	106.0	109.8	175.0	329.1	439

Table 3.3: Performance indexes for different segments of a hilly road located near UC Berkeley are reported for two cases of w/grade (proposed controller with grade knowledge) and wo/grade (the baseline controller without grade knowledge). The average of the total costs over all the segments is 416.3 W/O grade knowledge which is considerably larger then the 284.7 obtained with the proposed controller.

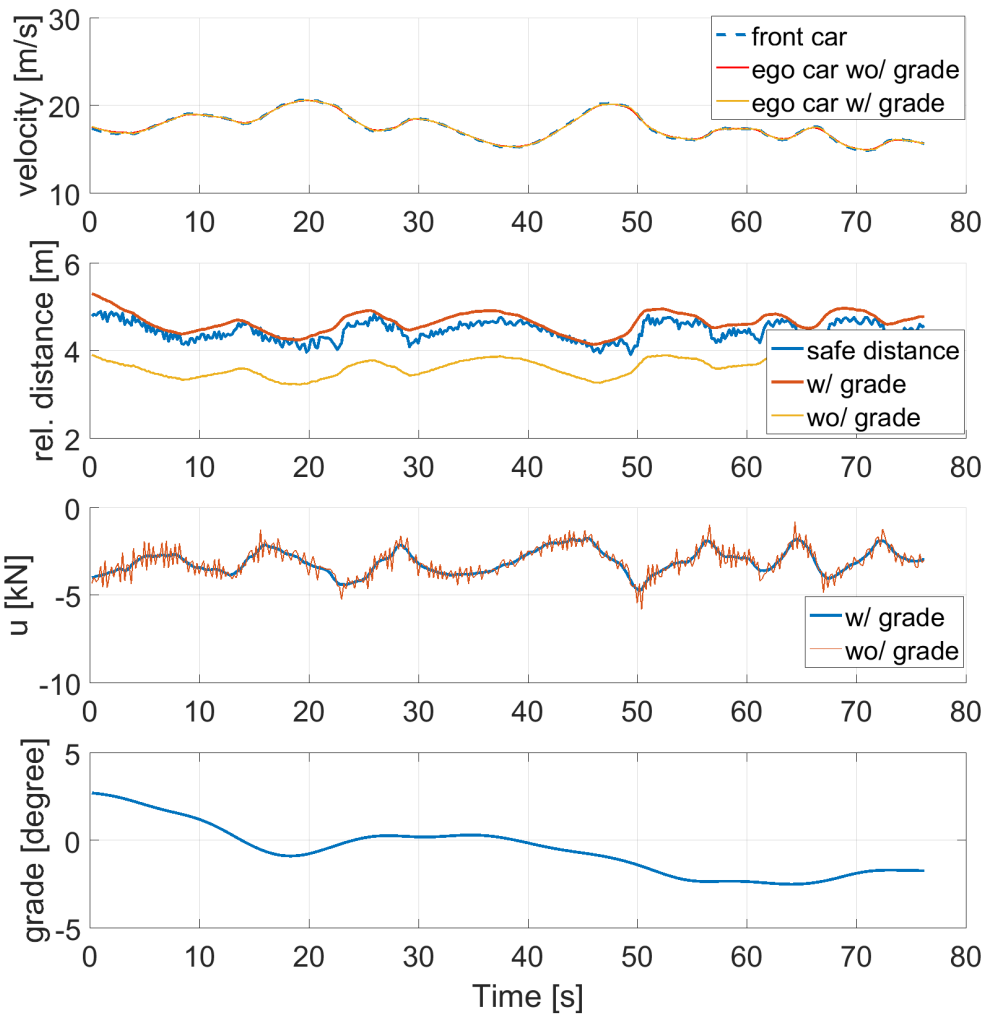


Figure 3.7: Car-following scenario: w/grade is the case that grade knowledge is available to the controller (presented controller), wo/grade is the case that grade knowledge is not available (baseline controller). The front car's velocity and road grade profile are realistic data.

algorithm is robust with respect to aggressive maneuvers of the lead car on roads with any arbitrary grade profile. In the autonomous intersection video the vehicles communicate with each other and after prioritization, the follower adapts its velocity based on the lead car's velocity and avoids collision by maintaining the safe distance calculated using the control invariant set. Both vehicles pass the intersection safely for an arbitrary road grade profile using the proposed approach.

As previously described, the presented controller is capable of automatically, safely and smoothly switching between CC and ACC modes. Fig. 3.8 shows the results for this case.

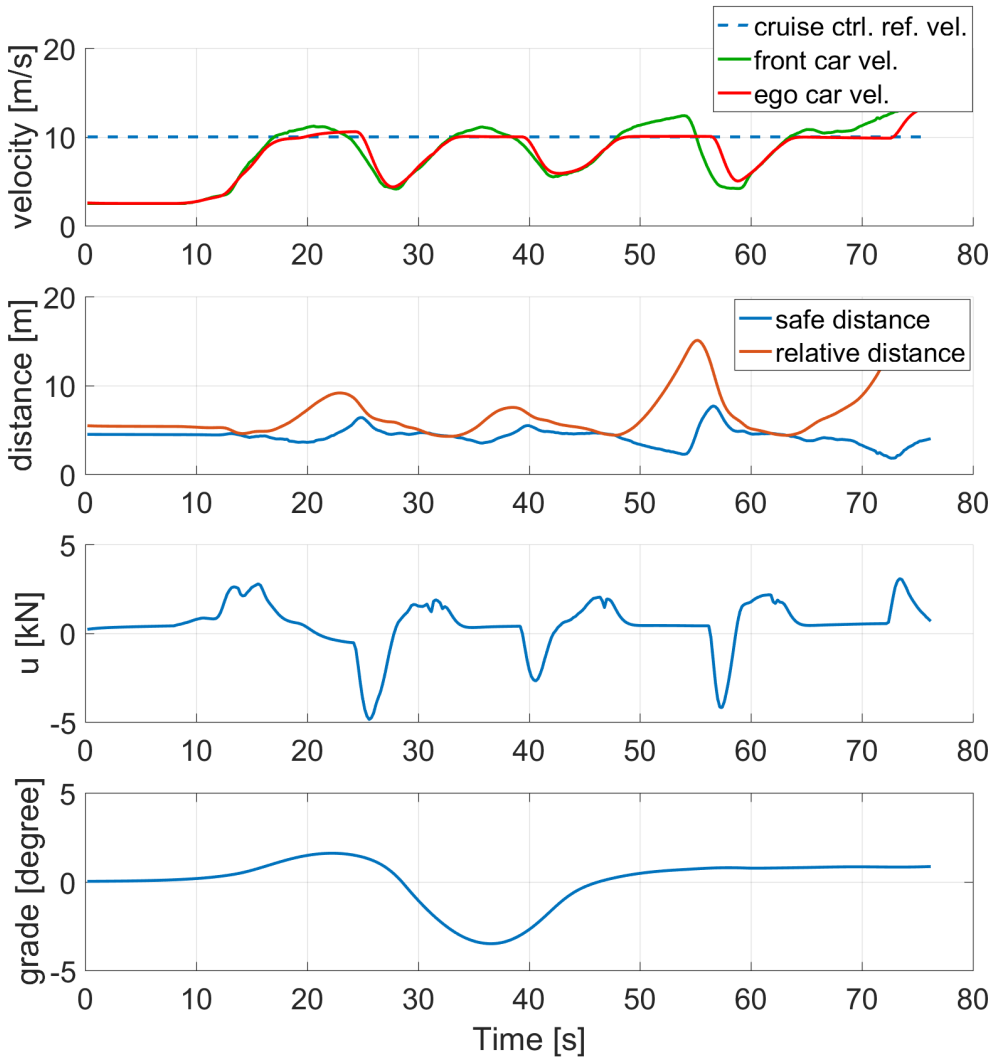


Figure 3.8: Automatic, smooth and safe switching between ACC and CC modes.

The front car's velocity is obtained by collecting real driving data and the associated road grade map is created using Google Elevation API. The cruise control speed is set to a constant value. By comparing the velocity and distance plots, one can see that the ego car velocity tracks the cruise control reference velocity when the front car is far, for example between the time of 15 to 25. Afterwards, for example between the time 28 to 34, since the front car's velocity is less than the cruise control reference speed, and the distance between the cars is closer, the ego car tracks the front cars' velocity instead. The relative distance shown in the second plot is also lower-bounded by the safe distance.

3.7 Conclusion

A performance-enhanced ACC controller is designed that exploits preview information about road grade as well as lead vehicle's motion to predict the evolution of the system and plan accordingly. To guarantee recursive feasibility of the closed-loop system, a less conservative robust control invariant set is computed with a numerical approach that calculates the minimum required safe distance at each time step in accordance with the associated road grade data and the lead car states. The presented controller is robust with respect to any aggressive braking of the lead vehicle as well as any arbitrary road slope.

Simulations are conducted using realistic data of lead car's velocity and road grade profile. It is verified through simulation for two application scenarios that the presented controller improves the performance in terms of comfort, safety and energy efficiency compared to the baseline controller. In addition, it is showed that the presented ACC design is able to switch between CC and ACC mode automatically, smoothly and safely.

Chapter 4

Centralized Coordination

4.1 Introduction

Vehicular wireless communication systems including vehicle-to-vehicle (V2V), vehicle-to-cloud (V2C) and vehicle to infrastructure (V2I) enhance cooperative driving by providing a communication network for information exchange between the vehicles to coordinate and plan conflict-free trajectories [31], [3]. Grouping multiple cooperative vehicles into single-lane or multi-lane formation is referred to as platooning. Using communication technologies, connected vehicles within the platoon can navigate in close proximity of each other, self-organize themselves to form certain configurations, keep tight formations and transit from one formation to another. Platooning improves traffic congestion, energy efficiency and safety [1], [66]. It increases road traffic throughput by allowing small inter-vehicle distances. Furthermore, moving with close spacing reduces aerodynamic drag and thus contributes to energy efficiency.

Platooning in classical setting refers to a group of vehicles that form a road train in a single lane [37], [64]. Single-lane platooning study and demonstrations date back to the '80s [54], [35]. The main drawback of forming a single-lane platoon is that a long train-like platoon may prevent other vehicles to change lane and consequently affect the traffic flow and reduce the mobility. Also in case of presence of obstacles on the road it might be impossible for a long platoon to find enough gap to change lane. Platoon formation in multiple lanes incorporates the advantages of platooning described earlier and at the same time is shape-reconfigurable and is able to facilitate lane change maneuvers as needed. In this paper, a multi-lane platoon with small number of interconnected vehicles (three up to ten) referred to as *mini-platoon* is considered. Adding another degree of freedom in multi-lane platoon increases structure flexibility and can further improve mobility, the traffic network throughput, energy efficiency and safety compared to single-lane platoon. For example, in terms of energy efficiency, when there is slow traffic ahead in one lane, multi-lane platoon can reconfigure its shape and perform opportunistic lane change to save the energy consumption by avoiding braking and changing the lane to a faster lane [32]. In terms of safety, once an

obstacle is detected in one lane, the multi-lane platoon can reconfigure and accommodate the vehicles in the blocked lane to merge into another lane to avoid the obstacle and minimize the risk of possible collision.

Although single-lane platooning (one-dimensional 1D) is well studied in the literature, literature on multi-lane platoons (two-dimensional 2D) is limited and reviewed in the next section. The focus of this chapter is to present a general architecture for autonomous navigation of tight multi-lane platoons. The contributions are summarized as follows.

1. An *architecture* for autonomous navigation of multi-lane platoons on public roads is presented. It comprises an *offline motion-planning system* and an *online hierarchical control system*.

2. A set of formation patterns also referred to as single-lane and multi-lane platoon configurations is identified. The offline motion planner uses an optimization-based algorithm to create various reconfiguration maneuvers that allow smooth transitioning from one pre-identified configuration to another. The resulting reconfiguration maneuvers are stored in a look-up table.

3. The online hierarchical control system is composed of three levels: a *traffic operation system (TOS)*, a *decision-maker*, and a *path-follower*. The top level TOS operates in the cloud and determines the desired platoon reconfiguration by monitoring the traffic. The middle-level decision-maker operates on the platoon leader vehicle. It makes use of the following information:

   - the desired reconfiguration from TOS, via V2C communication,
   - the look-up table computed by motion-planner, pre-stored on the vehicles,
   - and the shared future plans of the surrounding traffic (outside platoon) vehicles, via V2V communication.

   By incorporating all these information, the decision-maker checks whether the desired reconfiguration planned by TOS is feasible or not. The feasible maneuvers are broadcasted to all the vehicles within the platoon via V2V communication to be executed by the low-level path-following feedback controller in real-time.

4. The vehicles' shapes are modeled as polytopic sets and the collision avoidance constraints among them are reformulated into a set of smooth constraints using strong duality theory. These smooth constraints can be handled efficiently by standard nonlinear solvers. This approach allows navigation through tight spaces at highway speed.

5. Compared to existing literature, the three novel contributions discussed above address real-time implementation, tight maneuvering and hard constraint satisfaction. Uncertainty is not addressed in this work and is topic of ongoing research.

4.2 Centralized Planning Overview

Advances in vehicular communication technologies are expected to facilitate cooperative driving in the future. Connected and Automated Vehicles (CAVs) are able to collaboratively plan and execute driving maneuvers by sharing their perceptual knowledge and future plans. In this paper, an architecture for autonomous navigation of tight multi-lane platoons travelling on public roads is presented. Using the proposed approach, CAVs are able to form single or multi-lane platoons of various geometrical configurations. They are able to reshape and adjust their configurations according to changes in the environment. The proposed architecture consists of two main components: an offline motion planner system and an online hierarchical control system. The motion planner uses an optimization-based approach for cooperative formation and reconfiguration in tight spaces. A constrained optimization scheme is used to plan smooth, dynamically feasible and collision-free trajectories for all the vehicles within the platoon. This chapter addresses online computation limitations by employing a family of maneuvers precomputed offline and stored on a look-up table on the vehicles. The online hierarchical control system is composed of three levels: a traffic operation system (TOS), a decision-maker, and a path-follower. The TOS determines the desired platoon reconfiguration. The decision-maker checks the feasibility of the reconfiguration plan based on real-time information about the surrounding traffic. The reconfiguration maneuver is executed by a low-level path-following feedback controller in real-time. The effectiveness of the approach is demonstrated through simulations of three case studies: 1) formation reconfiguration 2) obstacle avoidance, and 3) benchmarking against behavior-based planning in which the desired formation is achieved using a sequence of motion primitives. Videos and software can be found online here

4.3 Literature Review

Coordinated formation methods for multiple autonomous vehicles are well-studied in the literature and can be categorized in three main approaches: Leader-follower, virtual structure, and behavior-based approach. In leader-follower approach the follower agents track the coordinates of the leader [47], [18]. This method is effective for conventional single-lane train-like platoon, but since the follower must follow the same reference trajectory as the leader, it is not applicable to reconfigurable multi-lane platoons, in which the planned motions for the vehicles are not the same. In virtual structure method, the formation is represented as a virtual rigid structure. Each robot is considered as a node in the rigid structure [53]. The main drawback of this method is that, the formation as a rigid structure is not flexible and reshapable.

Behavior-based approaches include methodologies such as flocking and particle swarm optimization algorithms, artificial potential fields, and sequence of motion primitives. Most of the studies on flocking algorithms consider the agents as a group of particles that interact with each other based on Reynolds heuristic rules of cohesion, separation and alignment

[56]. Cohesion enforces the particles to stay together and separation penalizes the collision between the particles. In artificial potential field method, potential fields are built so that the robot is attracted by the goal region and repelled by the obstacle region. In formation control, in addition to goal and obstacle potential fields, a swarm attractive field is introduced to achieve the desired formation pattern. The potential-based planning does not impose hard constraint on collision avoidance and cannot guarantee collision avoidance with constrained control input. In addition, all these particle-based methods model the vehicles as particles with radial gap among them and do not take the actual size of the vehicles into account. Furthermore, the dynamic model is considered to be the particle's dynamic with first, second or third-order point-mass models, which are not the representation of the actual nonlinear dynamics of the vehicles.

Another behavior-based method is to construct the formation maneuvers as sequences of motion primitives [6]. Motion primitives are identified as various behaviors such as lane change and obstacle avoidance. Among all the described formation approaches, this method is more effective for multi-lane platooning, but its disadvantage is that it is difficult to mathematically analyze and solve for sequence of motion primitives.

Combinations of the aforementioned approaches have also been studied. In [52], for example, the authors use the Reynolds rules to define the potential forces between the agents. Cohesion and separation are modeled as pairwise attractive and repulsive potential forces between the particle, respectively and a multi-objective cost function is constructed to satisfy all the rules simultaneously. In [43], the authors propose virtual leader approach with attractive potential field to track a desired path and achieve a desired formation and repulsive potential fields to avoid agents collisions. Also a Lyapunov function is constructed to prove the closed-loop stability. In [36], the authors use a similar approach for flocking of multiple non-holonomic vehicles and prove the convergence using LaSalle's invariant principle.

4.4 Preliminaries

Vehicle Model

The vehicles set composing the platoon is defined as $\mathcal{V}$. The number of vehicles are considered to be N_v and each vehicle is identified through its index $i \in \mathcal{V} := \{1, 2, ..., N_v\}$. The nonlinear behavior of every vehicle i within the set is modeled by the vehicle kinematic bicycle model, which is a common modeling approach in path planning. In this model, the ith vehicle state vector is $\mathbf{z}^i = [x^i, y^i, \psi^i, v^i]^\top$, where x^i and y^i represent longitudinal and lateral positions of the vehicle, respectively, ψ^i is the heading angle and v^i denotes the velocity at center of gravity (C.G.) of the vehicle, as seen in Fig. 4.1. The control input vector is defined as $\mathbf{u}^i = [a^i, \delta^i]^\top$, where a^i is the acceleration and δ^i is the steering angle. The vehicle dynamics

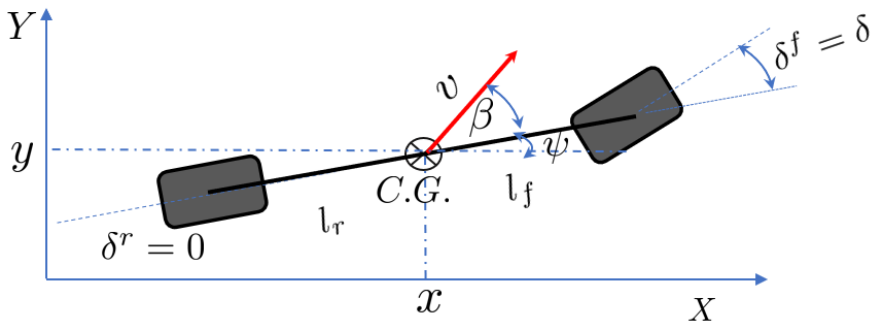


Figure 4.1: The kinematic bicycle model

is given as follows

$$\begin{aligned}
\dot{x}^i &= v^i \cos(\psi^i + \beta^i), \\
\dot{y}^i &= v^i \sin(\psi^i + \beta^i), \\
\dot{\psi}^i &= \frac{v^i \cos\beta^i}{l_f^i + l_r^i}(\tan\delta^i), \\
\dot{v}^i &= a^i,
\end{aligned} \tag{4.1}$$

where $\beta^i = \arctan\left(\tan\delta^i(\frac{l_r^i}{l_f^i+l_r^i})\right)$ is the side slip angle, l_f^i and l_r^i are the distance from the center of gravity to the front and rear axles, respectively. Superscript i denotes the ith vehicle in the platoon. Using Euler discretization, the model (4.1) is discretized as follows

$$\begin{aligned}
x^i(t+1) &= x^i(t) + \Delta t\ v^i(t)\cos(\psi^i(t) + \beta^i(t)), \\
y^i(t+1) &= y^i(t) + \Delta t\ v^i(t)\sin(\psi^i(t) + \beta^i(t)), \\
\psi^i(t+1) &= \psi^i(t) + \Delta t\ \frac{v^i(t)\cos\beta^i(t)}{l_f^i + l_r^i}(\tan\delta^i(t)), \\
v^i(t+1) &= v^i(t) + \Delta t\ a^i(t),
\end{aligned} \tag{4.2}$$

where Δt is the sampling time.

Platoon Configuration

Various platoon formation patterns or configurations are considered in this work, including one-lane (train-like) and multi-lane (rectangle, diamond, wedge shape, etc.), as shown in Fig. 4.2. The platoon configuration $\mathcal{C}$ is parameterized as $\mathcal{C}(n_v, l, p)$, where $n_v \in \mathbb{Z}$ is the maximum number of vehicles in each lane within the platoon, $l \in \{0,1\}^{n_l}$ is an indicator vector that specifies which lanes are occupied, n_l is the maximum number of lanes within

the platoon. The jth element of l is defined as

$$l(j) = \begin{cases} 0 & \text{if no vehicle is in } j\text{th lane} \\ 1 & \text{if at least one vehicle is in } j\text{th lane,} \end{cases}$$

where j denotes the lane index. The parameter matrix $p \in \mathbb{R}^{n_l \times n_v}$ represents the platoon geometrical pattern specified as the relative distances between the vehicles. Every jth row of matrix p is defined as $p(j) = [d_{j,\text{shift}}, d_{j1}, ..., d_{j(n_v-1)}]$, where $d_{j1}, ..., d_{j(n_v-1)}$ denote the horizontal inter-vehicle distances at jth lane as shown in Fig. 4.2(b) and $d_{j,\text{shift}}$ is the horizontal shifting distance of the front-most vehicle at each lane with respect to the front end of the reference vehicle. The right-most lane in direction of travel is the reference lane for jth lane, as shown in Fig. 4.2(b) and the reference vehicle is the front-most vehicle at reference lane. For the cars ahead of the reference vehicle, $d_{j,\text{shift}}$ is considered as negative. The values of $d_{j1}, ..., d_{j(n_v-1)}$ and $d_{j,\text{shift}}$ are design parameters and might be chosen as different values for each lane. For example, the platoon configuration in Fig. 4.2(b) is defined as

$$\mathcal{C} = \mathcal{C}(3, [1, 1, 1], p), \; p = \begin{bmatrix} 0 & 1 & 1 \\ 2 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix},$$

where $d_{1,\text{shift}}$, $d_{2,\text{shift}}$ and $d_{3,\text{shift}}$ associated with 1st, 2nd and 3rd lanes are $0m$, $2m$ and $1m$, respectively. Also $d_{11}, d_{12}, d_{21}, d_{31}$ and d_{32} are all $1m$ in this configuration. For trajectory optimization purposes, it is convenient to convert the configuration $\mathcal{C}$ to position coordinates $(x, y)^i$ of each vehicle i within the platoon. The function

$$g : \mathcal{C}(n_v, l, p) \rightarrow ((x, y)^1, \ldots, (x, y)^{N_v}), \tag{4.3}$$

gets the configuration $\mathcal{C}$ as input and outputs the position coordinates (x, y) for all the vehicles. The origin O, as shown in Fig. 4.2, is defined as the position of the rear-most vehicle at the right-most lane of the platoon configuration and all the coordinates are determined with respect to that origin.

Simple Reference Generator Model

A simple integrator function is defined which is used in Section 4.5, to generate the reference trajectories for each vehicle. The function $h : \mathbb{R}^3 \rightarrow \mathbb{R}^T$, is defined as

$$h : (x(0), v_{\max}, T) \rightarrow x_{\text{Ref}} = [x(0), x(1), ..., x(T)], \tag{4.4}$$

which determines x_{Ref} for all the vehicles within the platoon. The trajectory is obtained by $x(t+1) = x(t) + v_{\max}\Delta t, \quad \forall t \in \{0, 1, ..., T\}$, where $x(t)$ is the vehicle longitudinal position at time t, $v_{\max}$ is the maximum speed limit of the road, T is the final time of simulation and Δt is the simulation sampling time.

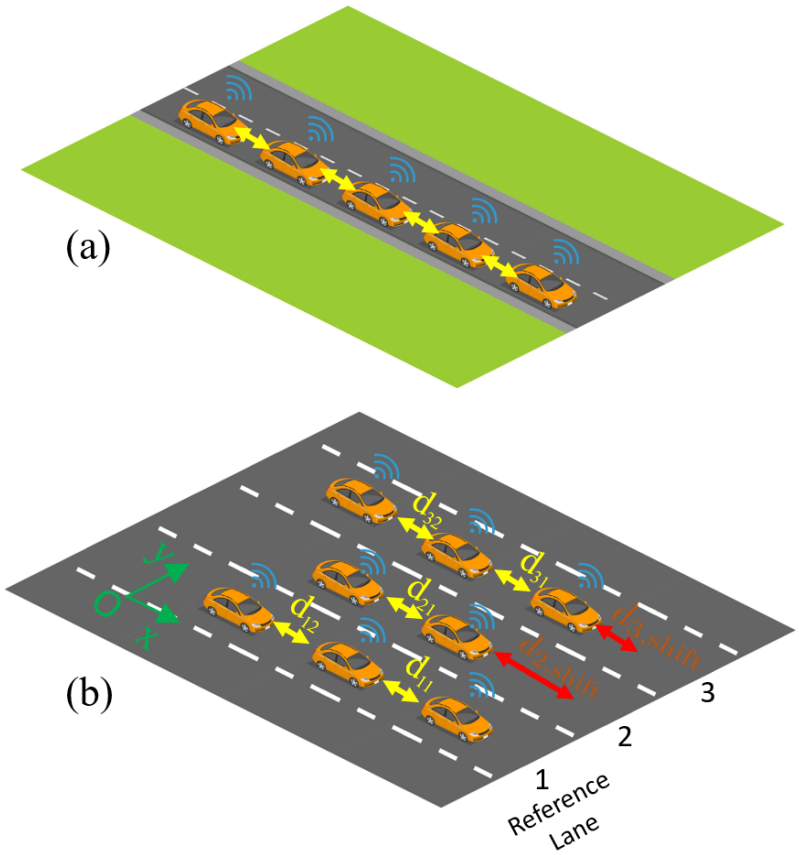


Figure 4.2: (a) Single-lane platoon: a train-like group of vehicles travelling at close distance behind each other. (b) Multi-lane platoon in multiple lanes. Yellow arrow depicts horizontal inter-vehicle distance at each lane and red arrow shows $d_{j,\text{shift}}$ at each lane. Each lane has its own label and the right-most lane is the reference lane.

Platoon Reconfiguration

Transitioning from an initial configuration denoted as $\mathcal{C}_i$ to a final configuration denoted as $\mathcal{C}_f$ is defined as platoon reconfiguration. An example of platoon reconfiguration is shown in Fig. 4.3. The top snapshot shows a multi-lane platoon with initial configuration $\mathcal{C}_i$ that is going forward at steady state (right-headed arrow shows the direction of motion). The middle snapshot shows transition maneuvers T_r and the vehicles change their lanes. Whenever the transition maneuver is completed, another configuration $\mathcal{C}_f$, which in this example is single-lane platoon, is achieved as shown in the bottom snapshot. A finite number of platoon configurations are identified as known configurations. The configuration set $\mathbb{C} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots\}$ captures all these pre-defined platoon configurations. The platoon reconfiguration scenarios are restricted to transition between these pre-defined configurations.

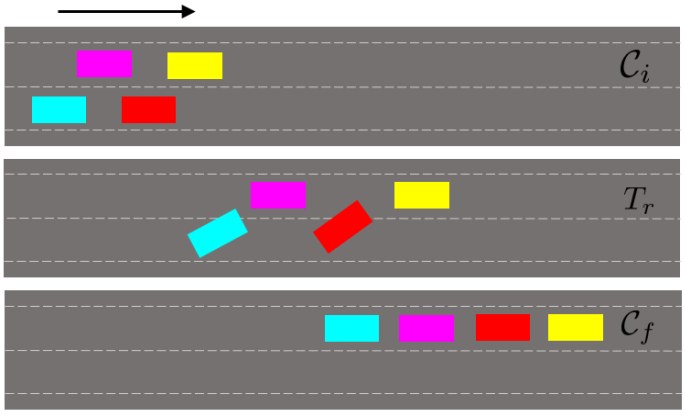


Figure 4.3: An example of reconfiguration from multi-lane platoon to single-lane platoon is shown.

Remark 1 *The traffic operation system (TOS) selects the desired platoon configurations among all the pre-identified configurations within the set $\mathbb{C}$, in such a way to improve traffic mobility and to reduce traffic congestion. The vehicles communicate with this level via V2C communication and receive $\mathcal{C}_i$ and $\mathcal{C}_f$. The platoon might not always be initially in a pre-identified $\mathcal{C}_i$ configuration, due to the changes in the surrounding traffic. Therefore, to initiate the reconfiguration, the vehicles are first controlled to reach $\mathcal{C}_i$ configuration. After reaching such a pre-defined configuration, the reconfiguration maneuver is initiated. Single-lane (1D) platoon formation from an unknown configuration has been studied for a long time. One way to reach the $\mathcal{C}_i$ is to first form a simple 1D platoon and initiate the reconfiguration from that known simple platoon. Another way (recommended in this paper) is to control the vehicles to reach $\mathcal{C}_i$. Each vehicle individually plans and controls to reach its corresponding location and when reaching the goal is not possible, due to the surrounding traffic condition, the vehicle informs TOS that reaching to $\mathcal{C}_i$ is infeasible and then TOS re-plans the reconfiguration.*

Surrounding Traffic

The vehicles set composing the surrounding traffic (the vehicles travelling close to the platoon, but do not belong to the platoon) is defined as the set $\mathcal{S} = \{1, ..., n\}$, where n is the total number of the surrounding vehicles. Each vehicle is identified through its index $q \in \mathcal{S}$. These vehicles are in the communication range of the platoon and share their future planned trajectories with the platoon leader. (The front-most vehicle in the reference lane within the platoon, is chosen as platoon leader.)

Different traffic scenario examples are shown in Fig. 4.4. In these examples, the multi-lane platoon (shown in red) can reconfigure to improve the traffic flow. In Fig. 4.4(a), a three-lane platoon is moving in the lanes 2,3,4. Since the traffic is slow in the lanes 2 and 4, a possible reconfiguration is that platoon can merge into the lane 3 and reconfigure as

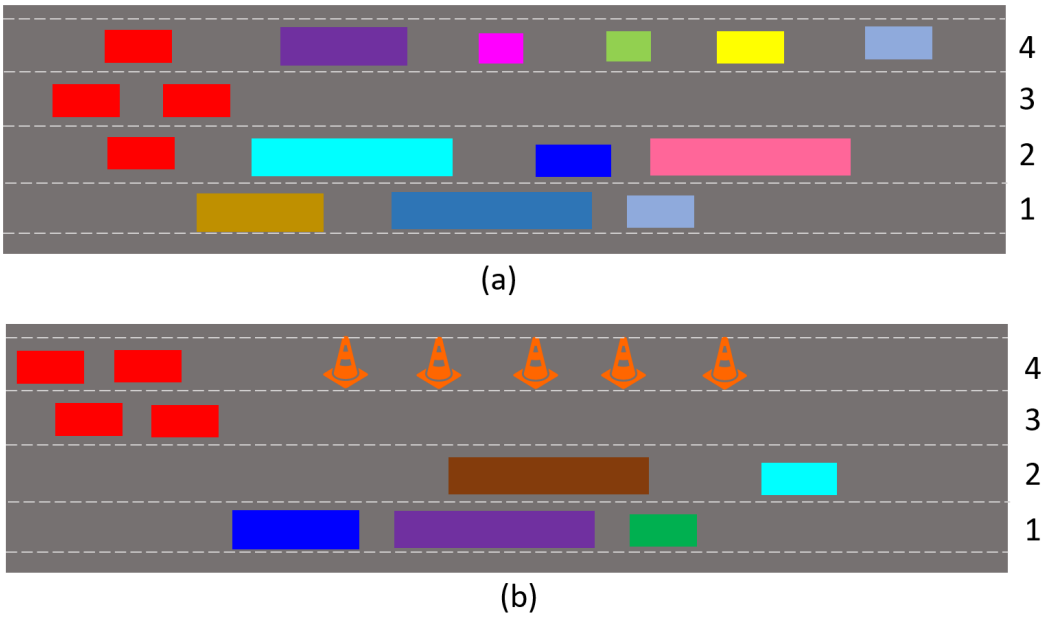


Figure 4.4: The vehicles within the platoon are shown in red and the surrounding traffic (vehicles not in platoon) are shown with different colors. (a): The traffic is slow in the lanes 2 and 4. (b): The lane 4 is closed.

a single-lane platoon. In Fig. 4.4(b), the lane 4 is closed due to an accident, the platoon vehicles in the lane 4 can merge between the platoon vehicles in the lane 3.

Notations

Common used notations along with their definitions are reported in the Table 4.1. The configurations are denoted using $\mathcal{C}$ and the trajectories are denoted using τ. The superscription i indicates ith vehicle.

4.5 Architecture

The proposed architecture for cooperative multi-vehicle systems consists of two main components: an offline motion planning system and an online hierarchical control system. Fig. 4.5 shows the architecture. The inputs of *motion planning* system are various initial $\mathcal{C}_i$ and final $\mathcal{C}_f$ configurations and the output of this system is a look-up table of precomputed safe maneuvers for transition from $\mathcal{C}_i$ to $\mathcal{C}_f$. The motion-planer uses an offline optimization-based approach for cooperative formation and reconfiguration. The *online hierarchical control system* is composed of three levels: traffic operating system, decision making and path following. The *traffic operating system (TOS)* monitors the traffic and determines the desired initial $\mathcal{C}_i$ and final $\mathcal{C}_f$ configurations of the platoon to improve traffic mobility and reduce road

Notation	Definition
$\mathcal{C}$	platoon configuration
$\mathcal{C}_i$	initial platoon configuration
$\mathcal{C}_f$	final platoon configuration
$\mathbb{C}$	the set of pre-defined/known configurations
N_v	number of vehicles
$\mathcal{S}$	surrounding traffic vehicles (not in platoon)
$\mathcal{V}$	set of all the vehicles
i	index of ith vehicle
$\mathcal{N}_i$	set of neighbor vehicles (within platoon) of ith vehicle
$\mathbf{z}^i$	states of ith vehicle
$\mathbf{u}^i$	inputs of ith vehicle
x^i	longitudinal position of ith vehicle
y^i	lateral position of ith vehicle
ψ^i	heading angle of ith vehicle
v^i	velocity of ith vehicle
$\mathbf{z}^i_{\text{Ref}}$	reference states of ith vehicle
$\mathcal{P}$	polytopic representation of the vehicle
T	final simulation (maneuver) time
$\mathbf{R}$	rotation matrix
$\mathbf{t}_r$	translation vector
$\mathbf{A}$ and $\mathbf{b}$	polytopic representation
len	the vehicle length
w	the vehicle width
$d_{\min}$	minimum safe distance
t	time step
k	horizon step
N	horizon
$\boldsymbol{\lambda}$, $\boldsymbol{\mu}$, $\mathbf{s}$	dual variables
$\boldsymbol{\tau}$	trajectory
$\boldsymbol{\tau}^i_{\mathbf{z}_{\text{Ref}}}$	ith vehicle reference state trajectory
$\boldsymbol{\tau}^i_{\mathbf{z}_{\text{Target}}}$	ith vehicle target trajectory (look-up table)
$\boldsymbol{\tau}^q_{\mathbf{z}}$	shared planned trajectory of qth surrounding vehicle
ρ	coefficient affecting the start of lane-change, $\rho \in (0, 1)$

Table 4.1: Common used notations

congestion. The *decision-maker* receives the desired initial $\mathcal{C}_i$ and final $\mathcal{C}_f$ configurations from TOS. Also it receives future planned trajectories from the surrounding traffic $\mathcal{S}$. Based on the given desired $\mathcal{C}_i$ and $\mathcal{C}_f$ and the surrounding traffic information, the decision-maker selects a feasible transition maneuver from the look-up table to reconfigure the platoon from $\mathcal{C}_i$ to $\mathcal{C}_f$. Once the transition maneuver is selected by the decision-maker, the maneuver is executed by the *path-follower* controller on each vehicle in real-time.

The following assumptions have been made:

(A1) The vehicles are fully autonomous and connected through vehicle-to-vehicle (V2V) and vehicle-to-cloud (V2C) communications.

(A2) All the platoon configurations $\mathcal{C}$ are selected from a pre-identified set of configurations.

(A3) The desired initial $\mathcal{C}_i$ and final $\mathcal{C}_f$ configurations are available from the topmost level of the architecture, which is the traffic operation system (TOS). The vehicles communicate with TOS via V2C communication.

(A4) Reconfiguration (transition maneuvers between initial $\mathcal{C}_i$ and final $\mathcal{C}_f$ configurations) always starts from a known (predefined) initial configuration $\mathcal{C}_i$. If the vehicles' current configuration is not identified as one of predefined configurations, the vehicles are controlled to reach the point for which $\mathcal{C}_i$ is available.

(A5) The road is assumed to remain straight along the reconfiguration maneuver.

(A6) Uncertainty due to communication delay or model mismatch is not considered; perfect knowledge of the states for all the vehicles is assumed.

Motion Planning

The motion planning is performed offline. For various identified initial $\mathcal{C}_i$ and final $\mathcal{C}_f$ platoon configurations, the transition maneuvers to reconfigure the platoon from $\mathcal{C}_i$ to $\mathcal{C}_f$ are computed by motion planner. These precomputed trajectories are stored in a look-up table to be executed by online hierarchical control system. The motion planning system has a hierarchical structure. At the high level, reference trajectories $\boldsymbol{\tau}^i_{\mathbf{z}_{\text{Ref}}}$ for each of the vehicles are generated based on initial and final configuration. These trajectories can cause collisions, which are resolved by a low level planner. At the low level, a trajectory optimization is formulated as a finite time constrained optimal control (FTCOC) problem to plan smooth, dynamically feasible and collision-free trajectories for all the vehicles in a centralized optimization problem. The motion planner incorporates the collision avoidance between the vehicles as constraints of optimization problem and obtains longitudinal a^i and lateral δ^i control inputs for all the vehicles.Solving a single FTCOC optimization for the entire maneuver (until time T) is computationally intractable due to the large number of decision variables. Therefore, multiple FTCOC with a shorter horizon N is solved, in a receding horizon fashion ($N < T$).

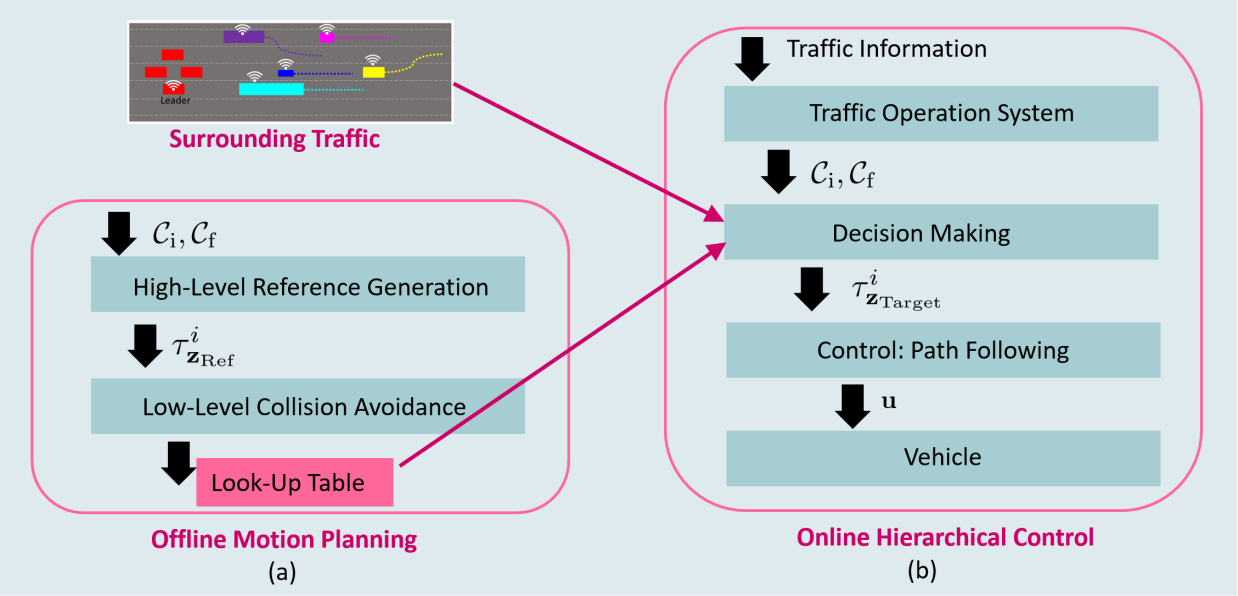


Figure 4.5: The architecture: (a) offline motion planning system. (b) online hierarchical control system

High-Level Reference Generation

The reference state for ith vehicle is denoted as $\mathbf{z}^i_\mathrm{Ref} = [x^i_\mathrm{Ref}, y^i_\mathrm{Ref}, \psi^i_\mathrm{Ref}, v^i_\mathrm{Ref}]$. The reference state trajectory, denoted as $\boldsymbol{\tau}^i_{\mathbf{z}_\mathrm{Ref}}$, is defined for the interval $[0, 1, 2, \ldots, T]$, from the initial time 0 until the final maneuver time T and $\boldsymbol{\tau}^i_{\mathbf{z}_\mathrm{Ref}} = \{\mathbf{z}^i_\mathrm{Ref}(0), \mathbf{z}^i_\mathrm{Ref}(1), \mathbf{z}^i_\mathrm{Ref}(2), \ldots, \mathbf{z}^i_\mathrm{Ref}(T)\}$. and is computed based on initial $\mathcal{C}_i$ and final $\mathcal{C}_f$ configurations of the platoon. First, the position coordinate of all the vehicles are specified using $g(\mathcal{C}_i(n_v, l, p)) = (x(0), y(0))^i \quad \forall i \in \mathcal{V}$, which is previously defined in Section 4.4. Then, the longitudinal position reference trajectory $\boldsymbol{\tau}^i_{x_\mathrm{Ref}}$ $= \{x^i_\mathrm{Ref}(0), \ldots, x^i_\mathrm{Ref}(T)\}$ is generated using the integrator model (4.4),

$$\boldsymbol{\tau}^i_{x_\mathrm{Ref}} = h(x^i(0), v_\mathrm{max}, T), \tag{4.5}$$

The lateral position reference trajectory $\boldsymbol{\tau}^i_{y_\mathrm{Ref}}$ is the y coordinate of the road centerline for each vehicle. For the first portion of simulation $(0, \ldots, \rho T)$, y^i_Ref is obtained from initial configuration $\mathcal{C}_i$ and the rest $((\rho T + 1), \ldots, T)$ is determined by final configuration $\mathcal{C}_f$, $g(\mathcal{C}_f(n_v, l, p)) = (x^i(T), y^i(T)) \quad \forall i \in \mathcal{V}$,

$$\{y^i_\mathrm{Ref}(0), \ldots, y^i_\mathrm{Ref}(\rho T)\} = y^i(0), \tag{4.6}$$

$$\{y^i_\mathrm{Ref}(\rho T + 1), \ldots, y^i_\mathrm{Ref}(T)\} = y^i(T), \tag{4.7}$$

the parameter $\rho \in (0, 1)$ is a tuning parameter, which can be determined by a design engineer. It is the coefficient that affects the start of the lane change. ψ^i_Ref is zero

$$\boldsymbol{\tau}^i_{\psi_\mathrm{Ref}} = \{\psi^i_\mathrm{Ref}(0), \ldots, \psi^i_\mathrm{Ref}(T)\} = 0, \tag{4.8}$$

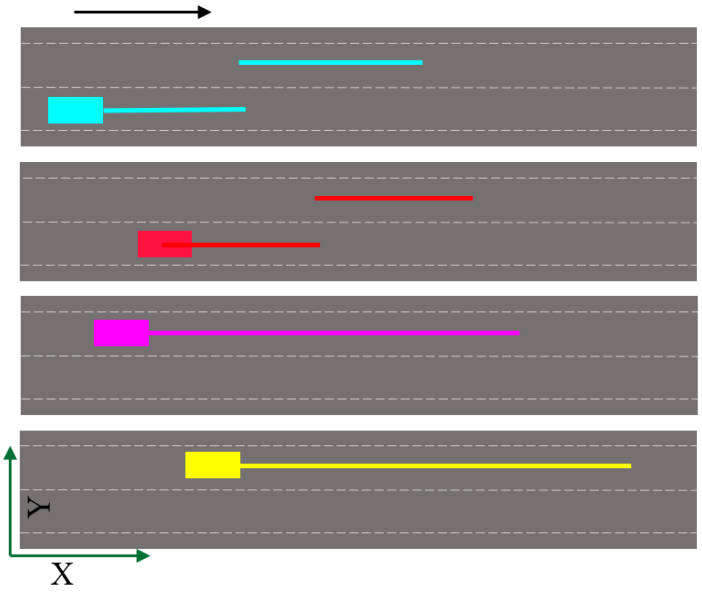


Figure 4.6: The generated reference trajectories are shown for the example scenario of Fig. 4.3. The parameter $\rho = 0.5$ for the blue and red vehicles.

assuming the road remains straight along the maneuver and v^i_{Ref} is set as maximum speed limit of the road or average traffic flow $v_{\max}$.

$$\boldsymbol{\tau}^i_{v_{\text{Ref}}} = \{v^i_{\text{Ref}}(0), \ldots, v^i_{\text{Ref}}(T)\} = v_{\max}. \tag{4.9}$$

The reference trajectory $\boldsymbol{\tau}^i_{\mathbf{z}_{\text{Ref}}}$ for ith vehicle is defined using (4.5), (4.6), (4.8) and (4.9). The generated trajectory is a naive initialization that might collide with obstacles. The low-level planner ensures collision avoidance among the vehicles. Fig. 4.6 shows the generated reference trajectories for the reconfiguration scenario example Fig. 4.3. The reference trajectories of blue and red vehicles are not straight lines, since they change their lanes. In Fig. 4.6 $\rho = 0.5$ for both blue and red vehicles. Note that the parameter ρ determines when the lane change starts. For example, $\rho = 0.5$ means the lane change is performed in the middle of the total duration of maneuver. For pink and yellow vehicles, ρ can be any value in the interval $(0, 1)$, excluding the boundaries, since pink and yellow do not change lane.

Low-Level Collision Avoidance

multi-vehicle motion planning problem is formulated as a centralized optimization problem that computes conflict-free trajectories for all the vehicles in the platoon simultaneously. The proposed optimization scheme uses a receding horizon fashion. At each time step it solves an optimization problem and obtains the control input based on dynamic model predictions over a time horizon and applies the first control input solution. At the next time step, the horizon is shifted forward and the procedure is repeated. The maneuvers are computed by closed-loop simulation of optimization (4.10) with dynamic model (4.1).

The objective function penalizes the deviation of each individual vehicle from the reference trajectory generated at the high level and the collision avoidance constraint is incorporated as hard constraint to guarantee safety. The optimization problem is formulated as follows

$$\min_{\mathbf{u}^i(\cdot|t)} \quad \sum_{i=1}^{N_V} \Big(\sum_{k=t}^{t+N} ||\mathbf{Q}_z(\mathbf{z}^i(k|t) - \mathbf{z}^i_{\text{Ref}}(k|t))||_2^2 + \sum_{k=t}^{t+N-1} ||\mathbf{Q}_u(\mathbf{u}^i(k|t))||_2^2 + ||\mathbf{Q}_{\Delta u}(\Delta \mathbf{u}^i(k|t))||_2^2 \Big) \tag{4.10a}$$

$$\text{subject to} \quad \mathbf{z}^i(k+1|t) = f(\mathbf{z}^i(k|t), \mathbf{u}^i(k|t)), \tag{4.10b}$$

$$\mathbf{z}^i(0|t) = \mathbf{z}^i(t), \tag{4.10c}$$

$$\mathbf{z}_{\min} \leq \mathbf{z}^i(k|t) \leq \mathbf{z}_{\max}, \tag{4.10d}$$

$$\mathbf{u}_{\min} \leq \mathbf{u}^i(k|t) \leq \mathbf{u}_{\max}, \tag{4.10e}$$

$$\Delta \mathbf{u}_{\min} \leq \mathbf{u}^i(k|t) - \mathbf{u}^i(k-1|t) \leq \Delta \mathbf{u}_{\max}, \tag{4.10f}$$

$$\mathcal{P}(\mathbf{z}^i(k|t)) \cap \mathcal{P}(\mathbf{z}^j(k|t)) = \emptyset, \quad i \neq j \tag{4.10g}$$

$$\text{for all } i \in \mathcal{V}, \ j \in \mathcal{N}_i,$$

where $\mathbf{u}^i(\cdot|t) = \{\mathbf{u}^i(t|t), ..., \mathbf{u}^i(t+N-1|t)\}$ denotes the sequence of control inputs over the planning horizon N for ith vehicle. The optimal solution is $U^*(t) = \{\mathbf{u}^*(t|t), ..., \mathbf{u}^*(t+N-1|t)\}$, and the receding horizon control law is obtained by applying the first control input $\mathbf{u}^*(t|t)$.

Superscript i denotes the ith vehicle, N_V is the total number of vehicles in the platoon, $\mathbf{z}^i(k|t)$ and $\mathbf{u}^i(k|t)$ are the state variable and control input of ith vehicle at step k predicted at time t, respectively. The above problem is a multi-objective optimization in which, the first term penalizes deviation of the states $\mathbf{z}$ from the reference state $\mathbf{z}_{\text{Ref}}$, the second term penalizes control input effort $\mathbf{u}$ and the third term penalizes the input rate (change of control input in two consecutive time steps) $\Delta\mathbf{u}$. The weight factors $\mathbf{Q}_z$, $\mathbf{Q}_u$ and $\mathbf{Q}_{\Delta u}$ are positive semidefinite matrices. The function $f(\cdot)$ in (4.10b) represents the vehicle kinematic bicycle model (4.2), which is discretized using Euler discretization. The reference trajectory obtained from the high level planner is denoted as $\mathbf{z}^i_{\text{Ref}}$ and $\mathbf{z}_{\min}$ and $\mathbf{z}_{\max}$ are the state limits and $\mathbf{u}_{\min}$ and $\mathbf{u}_{\max}$ are the input limits. The input rate is lower bounded by $\Delta\mathbf{u}_{\min}$ and upper bounded by $\Delta\mathbf{u}_{\max}$. Therefore, (4.10f) avoids heavy braking/acceleration as well as aggressive steering and enhances energy efficiency and comfort. $\mathcal{P}(\mathbf{z}^i(k|t))$ represents ith vehicle polytope as the road area occupied by the vehicle and $\mathcal{P}(\mathbf{z}^j(k|t))$ represents the other vehicle polytopes as moving obstacles for ith vehicle. The set of neighbors $\mathcal{N}_i$ is the set of all the vehicles within the platoon except ith vehicle and is defined as $\mathcal{N}_i = \mathcal{V} \setminus i$. In order to guarantee collision avoidance, the vehicles are modeled as polytopic sets that not only each set has empty intersection with all the other sets, but also each set keeps a minimum distance from the other sets. The collision avoidance between the ith vehicle and all the other vehicles

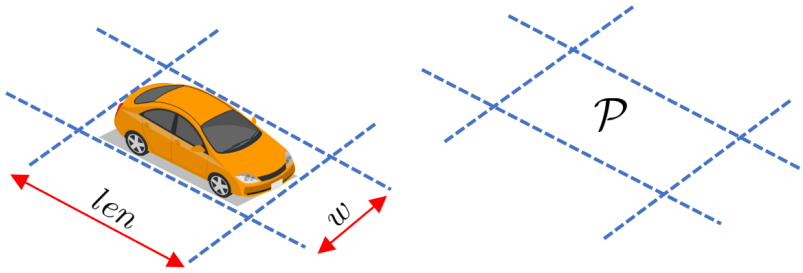


Figure 4.7: The occupied road region is modeled as a polytopic set that undergoes affine transformations.

(neighbors) is formulated in (4.10g), where $\mathcal{P}(\mathbf{z}^j)$ are the polytopic sets that represent all neighbor vehicles. The remainder of this section is devoted to detailed description and reformulation of the constraint (4.10g). The approach presented in [77] is used and applied to multi-lane platoon in the next section. The underlying technical reasoning is similar and repeated here for the sake of completeness. The computed trajectories from closed simulation of optimization (4.10) with dynamic model (4.1) are stored in a look-up table and will be executed in real-time by a path-follower which is a feedback controller.

Representation of the Road Area Occupied by the Vehicle

As discussed platooning is maintaining close inter-vehicular distance within a group of vehicles. In tight platooning, both road geometry (lane width) and platoon geometry (longitudinal and lateral inter-vehicle spacing) restrict the motion of the vehicles within the platoon and results in creating a tight environment. To allow navigation at tight spaces, it is essential to model the road structure and the vehicles dimensions as exact sizes with no approximation or enlargement. The vehicle pose or the corresponding road region occupied by the vehicle is defined by a two-dimensional convex polytope $\mathcal{P}$, as seen in Fig. 4.7. The initial pose of the vehicle is represented as $\mathcal{P}_o$. As the vehicle travels along the road, $\mathcal{P}_o$ undergoes affine transformations including rotation and translation. Hence $\mathcal{P}(\mathbf{z}(k)) = \mathbf{R}(\mathbf{z}(k))\mathcal{P}_o + \mathbf{t}_r(\mathbf{z}(k))$, where $\mathbf{z}(k)$ represents the vehicle state at kth time step, $\mathcal{P}(\mathbf{z}(k))$ is the vehicle occupied region as a function of the state $\mathbf{z}(k)$, and dimensions including length h and width w and is defined as a set of linear inequalities. $\mathbf{R} : \mathbb{R}^{n_z} \to \mathbb{R}^{n \times n}$ is an orthogonal rotation matrix and $\mathbf{t}_r : \mathbb{R}^{n_z} \to \mathbb{R}^n$ is the translation vector. n_z is the dimension of $\mathbf{z}$ and n is two, since the transformation is occurring in two-dimensional space $\mathbb{R}^2$. The rotation matrix $\mathbf{R}(\cdot)$ is a function of the vehicle heading angle $\psi(k)$ and the translation vector $\mathbf{t}_r(\cdot)$ is a function of the longitudinal $x(k)$ and lateral $y(k)$ positions of the vehicle. So the transformed polytope is defined as $\mathcal{P}(\mathbf{z}(k)) = \{[p_x, p_y]^\top \in \mathbb{R}^2 | \mathbf{A}(\mathbf{z}(k))[p_x, p_y]^\top \leq \mathbf{b}(\mathbf{z}(k))\}$, where p_x and p_y are the coordinates of points in two-dimensional space which are representation of the polytope.

The matrix $\mathbf{A}(\mathbf{z}(k))$ and the vector $\mathbf{b}(\mathbf{z}(k))$ are defined as

$$\begin{aligned}\mathbf{A}(\mathbf{z}(k)) &= \begin{bmatrix}\mathbf{R}(\psi(k))^\top \\ -\mathbf{R}(\psi(k))^\top\end{bmatrix}, \\ \mathbf{b}(\mathbf{z}(k)) &= [len/2, w/2, len/2, w/2]^\top \\ &+ \mathbf{A}(\mathbf{z}(k))[x(k), y(k)]^\top,\end{aligned} \tag{4.11}$$

where $\mathbf{R}(\psi(k)) = \begin{bmatrix}\cos(\psi(k)) & -\sin(\psi(k)) \\ \sin(\psi(k)) & \cos(\psi(k))\end{bmatrix}$. The length and width of the vehicle are denoted as len and w , respectively, as shown in Fig. 4.7. For coordination of multiple vehicles, each vehicle's occupied area is modeled as a time-varying polytope and at each time step, re-planning is performed such that no intersection occurs between the polytopic sets.

Collision Avoidance Reformulation

The distance between two polytopic sets $\mathcal{P}_1$ and $\mathcal{P}_2$ is defined as

$$\text{dist}(\mathcal{P}_1, \mathcal{P}_2) = \min_{\mathbf{x},\mathbf{y}} \|\mathbf{x} - \mathbf{y}\|_2 \,|\mathbf{A}_1 x \leq \mathbf{b}_1, \mathbf{A}_2 \mathbf{y} \leq b_2, \tag{4.12}$$

where $\mathcal{P}_1$ and $\mathcal{P}_2$ are described as $\mathbf{A}_1 x \leq \mathbf{b}_1$ and $\mathbf{A}_2 \mathbf{y} \leq \mathbf{b}_2$, respectively. The two sets do not intersect if $\text{dist}(\mathcal{P}_1, \mathcal{P}_2) > 0$. However, for autonomous driving applications, since the vehicles must keep a minimum safe distance d_{min} from each other and from the obstacles, the distance between their polytopic sets should be larger than a predefined minimum distance, $\text{dist}(\mathcal{P}_1, \mathcal{P}_2) \geq d_{min}$.

In the motion planning optimization problem (4.10), the collision avoidance is imposed as constraint. However, the collision avoidance formulated in (4.12) is itself an optimization problem. Hence, an optimization problem has to be solved as the constraint of another optimization problem. To deal with this issue, as explained in [77], the dual problem can be solved instead of the primal problem (4.12), based on strong duality theory. The dual problem is expressed as $\max_{\boldsymbol{\lambda},\, \boldsymbol{\mu},\mathbf{s}}\{-\mathbf{b}_1^\top \boldsymbol{\lambda} - b_2^\top \ \boldsymbol{\mu} : \mathbf{A}_1^\top \boldsymbol{\lambda} + \mathbf{s} = 0, \mathbf{A}_2^\top \ \boldsymbol{\mu} - \mathbf{s} = 0, \|\mathbf{s}\| \leq 1, \boldsymbol{\lambda} \succeq 0, \quad \boldsymbol{\mu} \succeq 0\}$, where $\boldsymbol{\lambda}$, $\boldsymbol{\mu}$ and $\mathbf{s}$ are dual variables. The optimal value of the dual problem is the distance between the two polytopes $\mathcal{P}_1$ and $\mathcal{P}_2$ and is constrained to be larger than minimum distance. Hence the constraint on dual problem optimal value is equivalent to the following feasibility problem $\{\exists \boldsymbol{\lambda} \succeq 0, \ \boldsymbol{\mu} \succeq 0, s : -\mathbf{b}_1^\top \boldsymbol{\lambda} - \mathbf{b}_2^\top \ \boldsymbol{\mu} \geq d_{\min}, \mathbf{A}_1^\top \boldsymbol{\lambda} + \mathbf{s} = 0, \mathbf{A}_2^\top \ \boldsymbol{\mu} - \mathbf{s} = 0, \|\mathbf{s}\| \leq 1\}$. This reformulation can be substituted instead of collision avoidance constraint (4.10g) in the motion planning optimization problem (4.10).

Therefore, problem (4.10) can be rewritten as

$$
\begin{aligned}
\min_{\substack{\mathbf{u}^i(\cdot|t),\ \boldsymbol{\lambda}_{ij}(\cdot|t),\\ \boldsymbol{\mu}_{ij}(\cdot|t),\ \mathbf{s}_{ij}(\cdot|t)}} \quad & (4.10a)\\
\text{subject to} \quad & (4.10b), (4.10c), (4.10d), (4.10e),\\
& \big(- \mathbf{b}_i(\mathbf{z}^i(k|t))^\top \boldsymbol{\lambda}_{ij}(k|t)\\
& - \mathbf{b}_j(\mathbf{z}^j(k|t))^\top\ \boldsymbol{\mu}_{ij}(k|t)\big) \geq d_{\min}, \qquad (4.13)\\
& \mathbf{A}_i(\mathbf{z}^i(k|t))^\top \boldsymbol{\lambda}_{ij}(k|t) + \mathbf{s}_{ij}(k|t) = 0,\\
& \mathbf{A}_j(\mathbf{z}^j(k|t))^\top\ \boldsymbol{\mu}_{ij}(k|t) - \mathbf{s}_{ij}(k|t) = 0,\\
& \|\mathbf{s}_{ij}(k|t)\| \leq 1, -\boldsymbol{\lambda}_{ij}(k|t) \leq 0,\\
& -\ \boldsymbol{\mu}_{ij}(k|t) \leq 0, \text{for all } i \in \mathcal{V},\ j \in \mathcal{N}_i,
\end{aligned}
$$

where $\mathbf{A}_i$ and $\mathbf{b}_i$ are functions of $\mathbf{z}^i(k|t)$ and represent the polytopic set of ith vehicle at step k predicted at time t. Similarly $\mathbf{A}_j$ and $\mathbf{b}_j$ denote the polytopic set of jth vehicle which belongs to neighbor set $\mathcal{N}_i$. The dual variables $\boldsymbol{\lambda}_{ij}$, $\boldsymbol{\mu}_{ij}$ and $\mathbf{s}_{ij}$ are coupled through the collision avoidance constraint among vehicle i and vehicle j. $\boldsymbol{\lambda}_{ij}(\cdot|t)$, $\boldsymbol{\mu}_{ij}(\cdot|t)$ and $\mathbf{s}_{ij}(\cdot|t)$ represent the sequence of dual variables over the optimization horizon N. So $\boldsymbol{\lambda}_{ij}(\cdot|t) = \{\boldsymbol{\lambda}_{ij}(t|t), ..., \boldsymbol{\lambda}_{ij}(t+N|t)\}$, $\boldsymbol{\mu}_{ij}(\cdot|t) = \{\boldsymbol{\mu}_{ij}(t|t), ..., \boldsymbol{\mu}_{ij}(t+N|t)\}$ and $\{\mathbf{s}_{ij}(\cdot|t) = \{\mathbf{s}_{ij}(t|t), ..., \mathbf{s}_{ij}(t+N|t)\}$.

One main advantage of the proposed planning method is that the required minimum distance between the vehicles $d_{\min}$, which can be chosen as a design parameter, is always enforced during the lane change maneuvers. In theory, the trajectories can be obtained for zero $d_{\min}$, which means the polytopic sets (cars) can move on each other boundaries. In practice, $d_{\min}$ should be determined based on the quantification of uncertainty of physical models and stochastic measurement errors, which is one future extension of this work.

The optimal solution of (4.13) is $U^*(t) = \{\mathbf{u}^*(t|t), ..., \mathbf{u}^*(t+N-1|t)\}$, and the first control input $\mathbf{u}^*(t|t)$ is applied to the vehicle nonlinear dynamic model (4.2). Then, the initial condition is updated with the current states and the optimization (4.13) is solved again. By running forward simulations of system (4.2) in closed loop with $\mathbf{u}^*(t|t)$ from the initial time 0 to the final maneuver time T, one can obtain collision-free closed-loop trajectories. Such closed-loop trajectories are represented by the state $\boldsymbol{\tau}^i_{\mathbf{z}_{\text{Target}}}$ trajectories. These trajectories are stored in a look-up table. The output of the motion planning system is this look-up table that captures different configurations and possible reconfigurations/transition maneuvers among them. Table 4.2 shows the structure of the look-up table. In the look-up table a set of trajectories are associated with ($\mathcal{C}_i$,$\mathcal{C}_f$) pair.

Note that for a specified pair of ($\mathcal{C}_i$,$\mathcal{C}_f$), once the high-level reference trajectory is computed, there is one optimal reconfiguration maneuver (the solution of optimization (4.13)) that transforms $\mathcal{C}_i$ to $\mathcal{C}_f$. However, the high-level reference trajectory computed by (4.6) and (4.7) is parameterized by $\rho \in (0, 1)$ and different choices of ρ result in different reference trajectories and consequently various reconfiguration maneuvers. In practice, several different values of ρ can be chosen, for example $\rho = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$ and the look-up table can be computed with these ρ values. So the family of reconfiguration maneuvers

Table 4.2: Look-up table structure (Parameterized by ρ).

Reconfiguration	**Trajectories** $\forall i \in \mathcal{V}$
$(\mathcal{C}_i, \mathcal{C}_f)$	$\boldsymbol{\tau}^i_{z_{\text{Target}}}(\rho^i)$ $\rho^i \in (0,1)$

Table 4.3: Look-up table structure (for specified ρ values).

Reconfiguration	**Index**	**Trajectories** $\forall i \in \mathcal{V}$
$(\mathcal{C}_i, \mathcal{C}_f)$	1	$\boldsymbol{\tau}^i_{z_{\text{Target}}}(\rho^i = 0.1)$
	2	$\boldsymbol{\tau}^i_{z_{\text{Target}}}(\rho^i = 0.2)$
	$\vdots$	$\vdots$
	M	$\boldsymbol{\tau}^i_{z_{\text{Target}}}(\rho^i = 0.9)$

from $\mathcal{C}_i$ to $\mathcal{C}_f$ is computed for these specified ρ values. The number of reconfiguration trajectories for a specific ($\mathcal{C}_i$,$\mathcal{C}_f$) pair is restricted to M numbers due to limited memory storage. The look-up table for specified ρ values is shown in Table 4.3. The nonlinear optimization (4.13) is not persistently feasible. The infeasible solutions are discarded and not included in the look-up table. Note that the motion-planner avoids collisions among the vehicles within the platoon ($i \in \mathcal{V}$). However, collision avoidance with surrounding traffic (vehicles outside the platoon), should be considered by decision-maker, as explained in Section 4.5.

Traffic Operation System

The traffic operation system (TOS) is the topmost level of online hierarchical control system and operates in the cloud. TOS determines the desired initial $\mathcal{C}_i$ and final $\mathcal{C}_f$ configurations based on the road traffic information. This level selects the desired platoon configurations among all the pre-identified configurations within the set $\mathbb{C}$, in such a way to improve traffic mobility and to reduce traffic congestion. The vehicles communicate with this level via V2C communication and receive $\mathcal{C}_i$ and $\mathcal{C}_f$. Determining the optimal desired configuration can be done with rule-based method and its discussion is out of the scope of this paper. Therefore, in this paper, based on assumption (A3), it is assumed that the initial $\mathcal{C}_i$ and final $\mathcal{C}_f$ configurations are already determined by TOS and are given to the vehicles.

Decision-Making

The decision-making system runs on an individual vehicle in the platoon. The front-most vehicle in the reference lane, is chosen as platoon leader on which the decision-maker operates. The decision-maker receives three types of information: 1) the desired initial $\mathcal{C}_i$ and final $\mathcal{C}_f$ configurations from the TOS obtained via cloud; 2) look-up table computed by motion-planner pre-stored on the vehicle; 3) the surrounding (outside of platoon) vehicles real-time

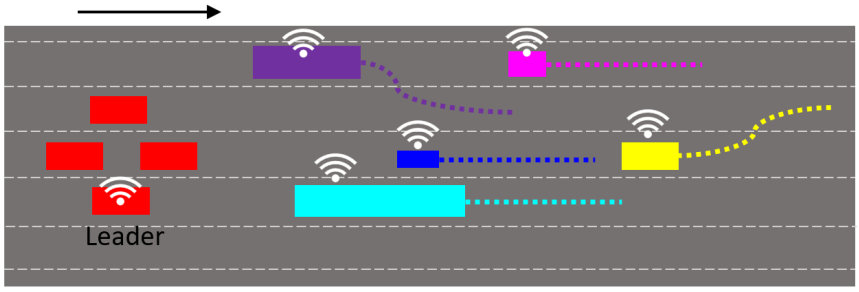


Figure 4.8: The red vehicles are within the platoon $i \in \mathcal{V}$ and the rest of vehicles with different colors are the surrounding traffic (outside platoon) vehicles $q \in \mathcal{S}$. The outside platoon vehicles share their future planned trajectories $\boldsymbol{\tau}_{\mathbf{z}}^{q}$, with the platoon leader (decision-maker) via V2V communication. The dotted lines illustrate the planned trajectories of surrounding traffic vehicles.

information obtained via V2V communication. This information includes the vehicles current states and future plans over the horizon that is equal or longer than the reconfiguration maneuver duration T. The decision-maker makes use of this information to check whether the desired reconfiguration maneuver planned by TOS is feasible or not.

Based on assumption (A1), all the vehicles are assumed to be autonomous and equipped with V2V communication, so the surrounding vehicles (outside of platoon) can communicate with decision-maker and share their future planned trajectories with it, as shown in Fig. 4.8. The surrounding traffic vehicles are defined as the set $\mathcal{S}$ and their future planned trajectories are denoted as $\boldsymbol{\tau}_{\mathbf{z}}^{q}$. Superscript q is index of the outside platoon vehicle. Decision-maker is responsible to ensure the planned desired reconfiguration from $\mathcal{C}_i$ to $\mathcal{C}_f$ is collision-free with respect to outside platoon traffic. To do a collision check between the platoon vehicles $i \in \mathcal{V}$ and surrounding traffic (outside platoon vehicles) $q \in \mathcal{S}$, their trajectories must be compared. Therefore, platoon reconfiguration trajectories $\boldsymbol{\tau}_{z\text{Target}}^{i}$ and the surrounding vehicles (not in platoon) future planned trajectories $\boldsymbol{\tau}_{\mathbf{z}}^{q}$ should be checked for collision at each time instant t. The positions $x(t)$, $y(t)$ and heading $\psi(t)$ are included in these trajectories. In addition the shared information of the surrounding vehicles include the vehicles dimensions including length len and width w.

Given two trajectories $\boldsymbol{\tau}_{\mathbf{z}}^{1} = \{\mathbf{z}^1(0), \ldots, \mathbf{z}^1(T)\}$ and $\boldsymbol{\tau}_{\mathbf{z}}^{2} = \{\mathbf{z}^2(0), \ldots, \mathbf{z}^2(T)\}$ associated with vehicles 1 and 2, respectively, the vehicles polytopic representations (4.11) at each time instant can be used to check whether the vehicles collide or not. The Algorithm 1 explains the collision-check procedure. Step ④ computes the polytopic representation of the vehicle 1, $\mathbf{A}_1(t)$, $\mathbf{b}_1(t)$ and the polytopic representation of the vehicle 2, $\mathbf{A}_2(t)$, $\mathbf{b}_2(t)$, for all the time steps $t \in 0, 1, \ldots, T$ in parallel. To do so, the algorithm uses (4.11). Since $\boldsymbol{\tau}_{z}^{1}$ includes the information $\mathbf{z}^1(t) = [x^1(t), y^1(t), \psi^1(t), v^1(t)]$, by substituting $x^1(t), y^1(t), \psi^1(t)$ and by including the vehicle dimensions, length len^1 and width w^1 in (4.11), the polytopic representation $\mathbf{A}_1(t)$, $\mathbf{b}_1(t)$ can be computed. The same procedure is repeated to find $\mathbf{A}_2(t)$, $\mathbf{b}_2(t)$. Step ⑤ uses (4.12) to compute the distance, $dist(t)$ between the two polytopic repre-

sentations (vehicles). When the distance is less than an acceptable safe minimum distance $d_{\min}$, the collision has occurred and the algorithm outputs true collision flag, otherwise it outputs false collision flag. Note that the problem (4.12) is a simple convex problem which is computationally cheap and suitable for real-time implementation.

Algorithm 1 Collision Check Algorithm

Inputs: $\boldsymbol{\tau}_z^1, \boldsymbol{\tau}_z^2$, and vehicles dimensions: len^1, w^1 and len^2, w^2
Output: collision flag **($\boldsymbol{\tau}_z^1$ and $\boldsymbol{\tau}_z^2$ collide if flag=True, and $\boldsymbol{\tau}_z^1$ and $\boldsymbol{\tau}_z^2$ do not collide if flag=False.)**
for all $t \in \{0, 1, \ldots, T\}$ **do** in parallel
    compute $\mathbf{A}_1(t)$, $\mathbf{b}_1(t)$ and $\mathbf{A}_2(t)$, $\mathbf{b}_2(t)$, the polytopic representation of vehicle 1 and vehicle 2, by using equation (4.11).
    compute the distance $dist(t)$ between the two vehicles (polytopes), by solving problem (4.12).
    if $dist(t) \geq d_{\min}$ **then**
        collision flag = False
    else
        collision flag = True, and go to step 12
    end if
end for
return collision flag

The Algorithm 2 explains the decision making process. At step ③, the decision-maker queries the pre-stored look-up table to get the family of trajectories associated with the pair $(\mathcal{C}_i, \mathcal{C}_f)$. Step ⑤ uses Algorithm 1 to check the collision between the selected platoon reconfiguration maneuver $\boldsymbol{\tau}_{\mathbf{z}_{\text{Target}}}$ and the future planned trajectories $\boldsymbol{\tau}_{\mathbf{z}}^q$ shared by surrounding (not in platoon) vehicles. The search over the family of trajectories in the look-up table is continued until finding a feasible platoon reconfiguration maneuver which has no conflict with the surrounding (not in platoon) vehicles. If the search finishes and no conflict-free reconfiguration maneuver is found, the decision-maker informs TOS that the current reconfiguration plan is infeasible. So, the plan is canceled and the vehicles will move forward with the current configuration. The decision-maker waits for the TOS to plan a new reconfiguration for the future time. If a feasible maneuver is found, it will be broadcasted through V2V network to all the vehicles and each vehicle executes its own trajectory in real-time via low-level path-following controller.

Path-Following

The path-following controller on each vehicle i executes ith vehicle corresponding maneuver and operates in real time. The desired maneuver (communicated by decision-maker) is represented by the state trajectory $\boldsymbol{\tau}_{\mathbf{z}_{\text{Target}}} = \{\mathbf{z}_{\text{Target}}(0), \ldots, \mathbf{z}_{\text{Target}}(T)\}$. The path-follower

Algorithm 2 Decision-Making Algorithm

Inputs: $\mathcal{C}_i, \mathcal{C}_f$, look-up table, $\boldsymbol{\tau}_{\mathbf{z}}^{q} \quad \forall q \in \mathcal{S}$.
Output: $\boldsymbol{\tau}_{\mathbf{z}_{\text{Target}}}(j)$ or infeasible flag.
query the look-up table to find the family of trajectories $\boldsymbol{\tau}_{\mathbf{z}_{\text{Target}}}$ associated with the pair $(\mathcal{C}_i, \mathcal{C}_f)$.
for j = 1 to M **do**
    check the collision between $\boldsymbol{\tau}_{\mathbf{z}_{\text{Target}}}(j)$ and $\boldsymbol{\tau}_{\mathbf{z}}^{q}$ by running the Algorithm 1.
    if collision flag = False **then**
        return $\boldsymbol{\tau}_{\mathbf{z}_{\text{Target}}}(j)$
    end if
end for
return infeasible flag

is designed using model predictive control (MPC) as follows

$$
\begin{aligned}
\min_{\mathbf{u}(\cdot|t)} \quad & \Bigg(\sum_{k=t}^{t+N} ||\mathbf{Q}_z^{pf}(\mathbf{z}(k|t) - \mathbf{z}_{\text{Target}}(k|t))||_2^2 \\
& + \sum_{k=t}^{t+N-1} \left(||\mathbf{Q}_{u2}^{pf}(\mathbf{u}(k|t))||_2^2 \right. \\
& \left. + ||\mathbf{Q}_{\Delta u}^{pf}(\Delta\mathbf{u}(k|t))||_2^2 \right) \Bigg) \\
\text{subject to} \quad & \mathbf{z}(k+1|t) = f(\mathbf{z}(k|t), \mathbf{u}(k|t)), && \text{(4.14a)} \\
& \mathbf{z}(0|t) = \mathbf{z}(t), && \text{(4.14b)} \\
& \mathbf{z}_{\min} \leq \mathbf{z}(k|t) \leq \mathbf{z}_{\max}, && \text{(4.14c)} \\
& \mathbf{u}_{\min} \leq \mathbf{u}(k|t) \leq \mathbf{u}_{\max}, && \text{(4.14d)} \\
& \Delta\mathbf{u}_{\min} \leq \mathbf{u}(k|t) - \mathbf{u}(k-1|t) \leq \Delta\mathbf{u}_{\max}, && \text{(4.14e)}
\end{aligned}
$$

where the notations are similar to the notations in problem (4.10). The superscript i is removed, because each car independently runs the path-following controller. The first term of the objective penalizes the state deviation from the target state trajectory $\boldsymbol{\tau}_{\mathbf{z}_{\text{Target}}}$, the second and third terms penalize control input effort and input rate, respectively. The weight factors, $\mathbf{Q}_z^{pf}$, $\mathbf{Q}_{u1}^{pf}$, $\mathbf{Q}_{u2}^{pf}$ and $\mathbf{Q}_{\Delta u}^{pf}$ are super-scripted by pf to be distinguished from the weight factors in problem (4.10). These weight factors should be tuned to achieve high tracking performance. The constraints (4.14a)-(4.14e) are the same as the constraints (4.10b)-(4.10f) in problem (4.10).

Configuration Design Heuristics

The two main factors that should be considered in configuration design are 1) inter-vehicle longitudinal spacing in one lane d and 2) the shifting distance in two adjacent lanes d_{shift}.

Figure 4.9: (a) The configuration without shifting distance in adjacent lanes. (b) The configuration with shifting distance.

The small inter-vehicle longitudinal gap reduces air drag, contributes to energy saving and improves traffic throughput, as discussed earlier. In addition, the small gap prevents the surrounding traffic (outside platoon vehicles) to cut-in between the platoon vehicles. On the other hand the longitudinal spacing should be large enough to ensure safety and robustness to uncertainties. Furthermore, a shifting distance between two adjacent lane facilitates lane-change maneuvers. Fig. 4.9(a) shows a platoon configuration with no shifting distance in adjacent lanes. Fig. 4.9(b) shows a configuration with shifting distance in adjacent lanes. The platoon configuration (b) is more flexible for reconfiguration compared to the platoon configuration (a). Choosing the optimal values of d and d_{shift} should be done using experimental data and is another extension of this work.

4.6 Formation as Sequence of Motion Primitives

An alternative approach for the proposed optimization-based motion planning is behavior-based planning. In this section, a behavior-based planning using sequence of motion primitives [6] is reviewed. This behavior-based approach is used to benchmark the proposed optimization-based planning against. As described in Section 4.3, among all the existing methods, the behavior-based approach which uses a sequence of motion primitives is more suitable for formation of multi-lane platoons. In robotics applications, a complex dynamical task is achieved by synthesizing a sequence of motion primitives. In a similar way, achieving the desired platoon formation requires that a sequence of motion primitives to be performed by each single vehicle in the platoon. This method is considered as a baseline and the proposed optimization-based motion planning approach is compared with this behavior-based method using a simple example scenario in Section 4.7 and the advantages of the proposed approach are discussed.

For each motion primitive a number of parameters have to chosen. The examples of parameterized motion primitives for a single car in multi-lane formation are

- slow down: parameterized by desired speed and desired deceleration),
- cruise control (CC): parameterized by desired speed
- lane change: parameterized by lane index, desired acceleration or deceleration),

- adaptive cruise control (ACC): parameterized by the front's car velocity and the desired inter-vehicle distance.

Planning sequence of motion primitives for each vehicle in the platoon to achieve a certain formation is hard to formulate and analyze mathematically. In this method, the system of vehicles is modeled as a hybrid system with various motion primitives as discrete modes and the transition maneuvers between them as continuous dynamics. To plan a sequence of motion primitives a mixed-integer program (MIP) has to be solved, where different types of motion primitives are integer decision variables and the vehicles' states are the continuous decision variables. However, MIPs are in general difficult to solve. An alternative common approach is to obtain the sequence of motion primitives according to a rule-based approach and then execute each motion primitives using the individual controllers for each primitives. Since the study of behavior-based approach is not the focus of this paper, the problem is simplified and the sequence of motion primitives for each vehicle are assumed to be already determined based on some rules. Given the sequence of primitives, the controllers are designed to execute them. All the controllers are designed, using MPC scheme such that the reference tracking cost is minimized while respecting vehicle dynamics and input and state limits. To keep the brevity of the paper, the controllers' mathematical formulations are not discussed here, but detailed description can be found in the authors' previous works. For example, an MPC cruise controller (CC) discussed in [26] is designed to execute following a desired velocity. Also, an adaptive cruise control (ACC) is designed to maintain a proper distance from the front car and follow the front car's velocity, using the MPC formulation described in [25]. "Lane change" is achieved by changing the center of lanes as reference. In Section 4.7, these controllers are used to execute the given motion primitives for a simple example scenario for multi-vehicle formation.

4.7 Numerical Results

Three simulation scenarios are conducted to verify the effectiveness of the proposed motion planning algorithm. The simulations are conducted in MATLAB, the optimization problem is modeled using YALMIP and the nonlinear optimization is solved using IPOPT. The results are reported for three cases: a) platoon formation and re-configuration, b) obstacle avoidance, and c) comparison with behavior-based approach. The vehicle dimensions are chosen as 4.5m length and 1.8m width. The road width is chosen as 3.7m, which is the highway lane width standard at the United States. The control input limits are chosen as realistic physical limits of actual passenger vehicle. The acceleration input lower and upper bounds are chosen as -4m/s^2 and 4m/s^2, respectively and its change is limited to -1m/s^2 and 1m/s^2. The steering input lower and upper bounds are chosen as -0.3rad and 0.3rad and its change is limited to 0.2rad/s. At each iteration the optimization problem (4.13) is solved and the first control input is applied to the vehicle kinematic model (4.1) for all the vehicles. Then the horizon is shifted and same procedure is repeated for the next step. For

all the three scenarios the simulation results are presented as top view snapshots, as well as a series of state and action plots. The vehicles colors of the snapshots and plots are matched. The video for formation reconfiguration and obstacle avoidance scenarios is available online at this link https://github.com/RoyaFiroozi/Centralized-Planning.

Platoon Re-Configuration

In this scenario the platoon formation is alternating between two different configurations, as seen in Fig. 4.10. The platoon of four vehicles is moving in a two-dimensional configuration. The vehicles are moving in three different lanes and the platoon reshape into one-dimensional configuration and all the vehicles merge into one lane. The initial configuration is $\mathcal{C}_i(2, [1, 1, 1], p_i)$, with

$$P_i = \begin{bmatrix} 0 & 5.5 \\ 6 & 0 \\ -4.5 & 0 \end{bmatrix}.$$

The final configuration is $\mathcal{C}_f(4, [0, 1, 0], p_f)$, with

$$p_f = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0.3 & 0.3 & 0.3 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

The initial longitudinal coordinates for all the four vehicles are

$$\begin{bmatrix} x^1(0) & x^2(0) & x^3(0) & x^4(0) \end{bmatrix} = \begin{bmatrix} 10.5 & 4.5 & 0.5 & 15 \end{bmatrix},$$

and the initial lateral coordinates are

$$\begin{bmatrix} y^1(0) & y^2(0) & y^3(0) & y^4(0) \end{bmatrix} = \begin{bmatrix} 1.85 & 5.55 & 1.85 & 9.25 \end{bmatrix}.$$

$d_{\min}$ is chosen as 0.3 m, the horizon N is 5, sampling time Δt is 0.2s, simulation time T is 120, ρ is 0.25 and $v_{\max}$ is 20m/s. Fig. 4.10 represents the vehicles' states and actions. The plots show the transient behavior between the two modes or configurations. The longitudinal and lateral coordinates x and y, as well as heading angle ψ and velocity v for all the vehicles are shown in different colors which are matched with the colors in Fig. 4.10. The control actions a and δ are also illustrated for all the vehicles. As seen the platoon reaches its steady state at final configuration after about 25 seconds.

Obstacle Avoidance

In obstacle avoidance scenario multiple vehicles are traveling together in a multi-lane platoon formation and once an obstacle is detected in the left Lane, the TOS selects reconfiguration to a single-lane configuration in the right lane. The vehicles in the other lane make enough gap to facilitate safe and smooth lane changing and merging for the vehicles in

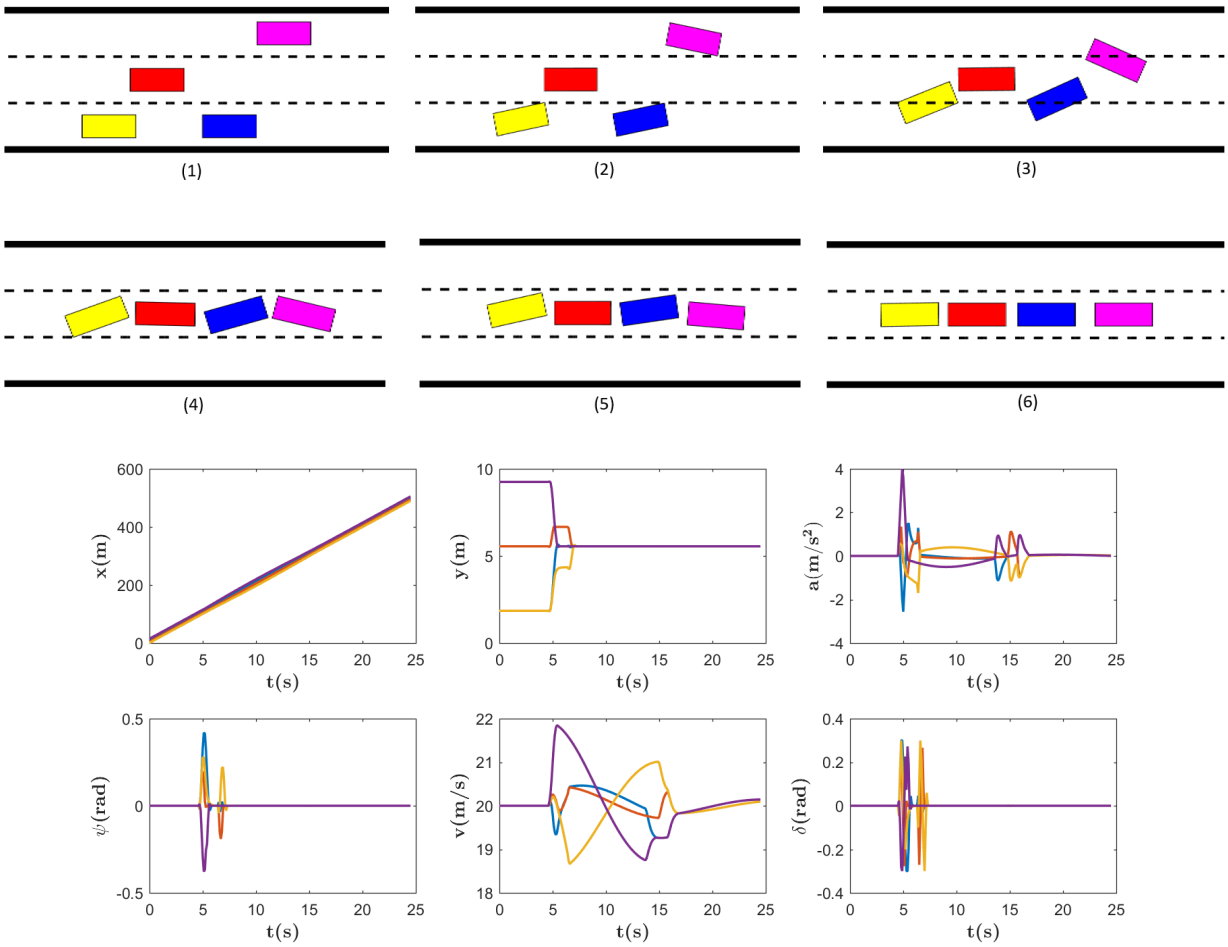
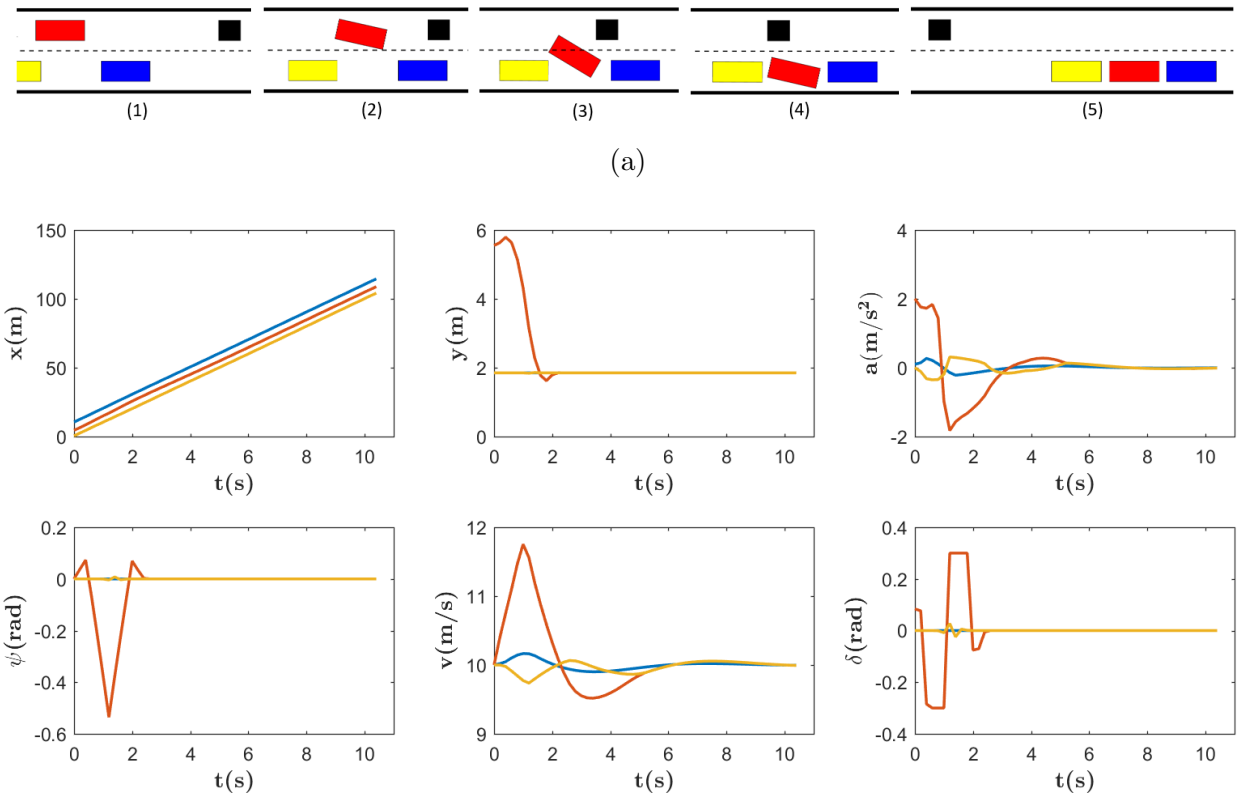


Figure 4.10: **Top:** Platoon reshapes from multi-lane configuration into single-lane configuration. Four vehicles moving in three different lanes merge in one lane. Step (1) shows four vehicles moving to the right in three different lanes at steady state. Steps (2) to (5) show the merging maneuver and finally step (6) demonstrates single-lane platoon configuration as another steady state of the platoon. **Bottom:** The vehicles' states and actions in a merging maneuver are presented. The colors of all the plots are matched with the color of the vehicles in top view snapshots.



(a)

(b)

Figure 4.11: (a) A static obstacle (black object) is detected in the red vehicle lane, the yellow and blue vehicles make gap for the red to merge into their lane. Vehicles are travelling in 2D configuration are flexible and are able to reshape in case of presence of obstacle in one lane. (b) The plots correspond to the snapshots and represents the vehicles' states and actions during the simulation. As seen, the steady state is achieved and 1D configuration is formed.

the lane with obstacle. Fig. 4.11a shows the top view snapshots for obstacle avoidance simulation. The red vehicle has to change lane because a static obstacle (black object) has been detected on its lane. The yellow and blue vehicles make gap for the red vehicle to merge into their lane. The obstacle is modeled as a polytopic set and the obstacle avoidance constraints are introduced. The initial longitudinal coordinates for all the three vehicles are $[x^1(0), x^2(0), x^3(0)] = [10.5, 4.5, 0.5]$ and the initial lateral coordinates are $[y^1(0), y^2(0), y^3(0)] = [1.85, 5.55, 1.85]$. $d_{\min}$ is chosen as 0.2m, the horizon N is 8, sampling time Δt is 0.1s, simulation time T is 100, ρ is 0.25 and $v_{\max}$ is 10m/s. The vehicles' states and actions are shown for in Fig. 4.11b. As seen, the steady state is achieved and 1D platoon is formed after about 10 seconds.

Comparison with Behavior-Based Approach

To compare the proposed approach with the behavior-based approach discussed in Section 4.6, a simple example scenario is considered. Two vehicles, which are moving together in the same lane, make enough gap for the third vehicle to allow it to merge into their lane. This simple scenario is chosen to be able to determine the sequence of motion primitives for each vehicle intuitively without any mathematical analysis. However, sequence of motion primitives should be obtained using mathematical analysis such as MIP for more complicated scenarios. The simulation results for behavior-based approach is shown in Fig. 4.12a. The sequence of motion primitives for this simulation are:

1. Red car follows a constant desired velocity (CC).
2. Yellow car slows down.
3. Blue car performs lane-change.
4. Blue car follows the red car (ACC).
5. Yellow car follows the blue car (ACC).

As seen in Fig. 4.12a, at step (1), the cars are moving to the right in two-dimensional platoon and the yellow car slows down to make a proper gap to allow the blue car to merge into the lane, while the red car is moving with constant speed. Step (2) shows the lane change of the blue car. Step (3) illustrates the reconfiguration of one-dimensional platoon. In this simulation, the collision avoidance constraints among the cars are not imposed, so the blue car changes its lane only after a large enough gap is created between the red and yellow cars. Even for this simple scenario obtaining maneuvers with larger velocity and closer inter-vehicle distance was impossible after running extensive simulations. The same scenario is replicated with optimization-based planning. The same initial conditions and parameters are used for both methods. The initial longitudinal coordinates for the three vehicles are $[x^1(0), x^2(0), x^3(0)] = [6, 12, 0.5]$ and the initial lateral coordinates are $[y^1(0), y^2(0), y^3(0)] = [1.85, 5.55, 5.55]$. $d_{\min}$ is chosen as 0.2m, the horizon N is 8, the simulation sampling time Δt is 0.1s, simulation time T is 150, ρ is 0.25 and $v_{\max}$ is 10.5m/s. The resulting maneuvers obtained by motion primitive and optimization-based approaches are presented in Fig. 4.12b and Fig. 4.12c, respectively. As seen in Fig. 4.12b, the x plot, the yellow car longitudinal position is far behind the other two. Also in v plot, the yellow car reduces its speed dramatically and the blue car is changing its speed. However in Fig. 4.12c, that shows the obtained trajectories using optimization-based approach the cars maintain a tight inter vehicle distance as seen in the x plot and the velocities and accelerations are changing smoothly.

In addition, for this example, despite extensive tuning efforts, it was not possible to obtain trajectories at highway speed and tight inter-vehicle distance, using motion primitive approach. The reason is that this approach requires proper tuning of many parameters and

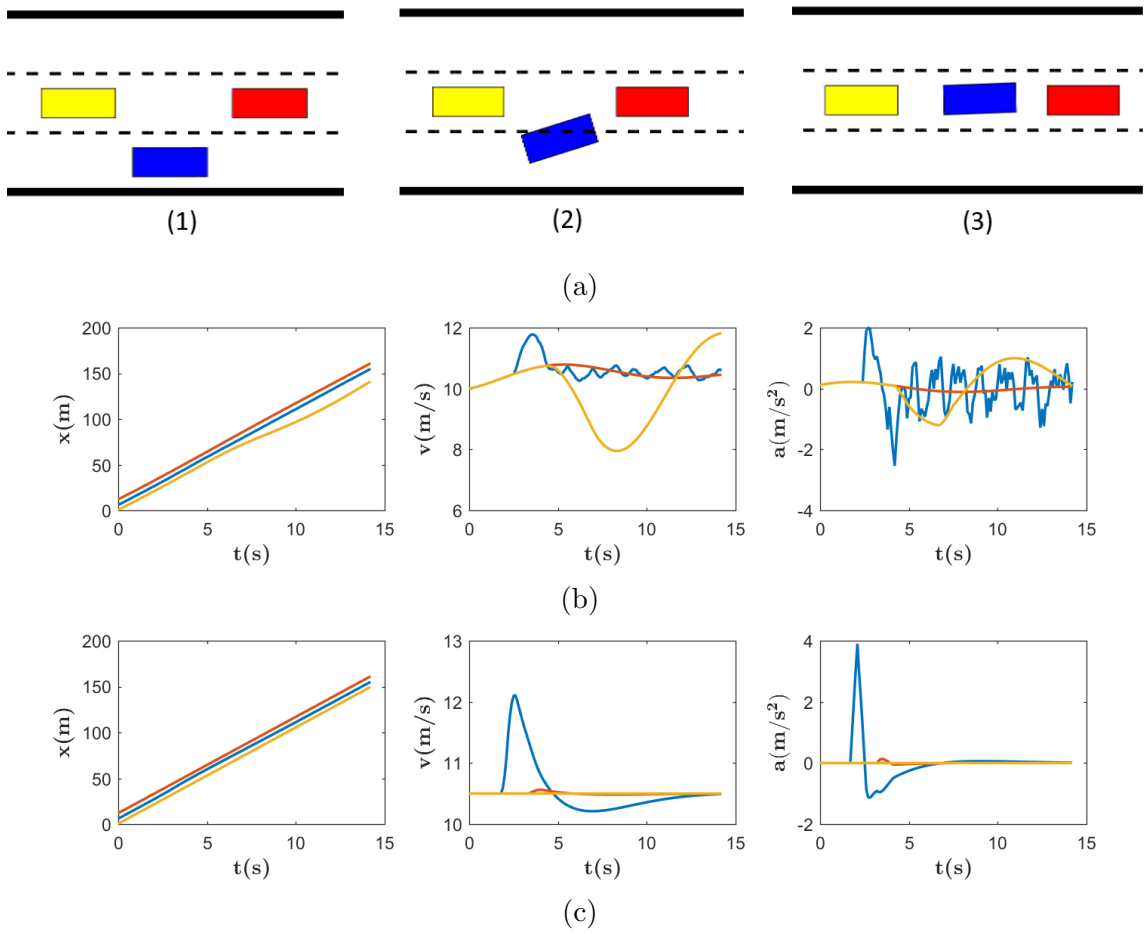


Figure 4.12: (a) Formation using sequence of motion primitives is demonstrated. At step (1), the cars are moving to the right in two-dimensional formation and the yellow car starts slowing down to make enough gap for the blue car to merge, while the red car doesn't change its speed. At step (2), the blue car changes lane to merge the platoon. At step (3) blue follows the red car and yellow follows the blue car and 1D platoon is formed. (b) Planning using sequence of motion primitives (c) Optimization-based planning.

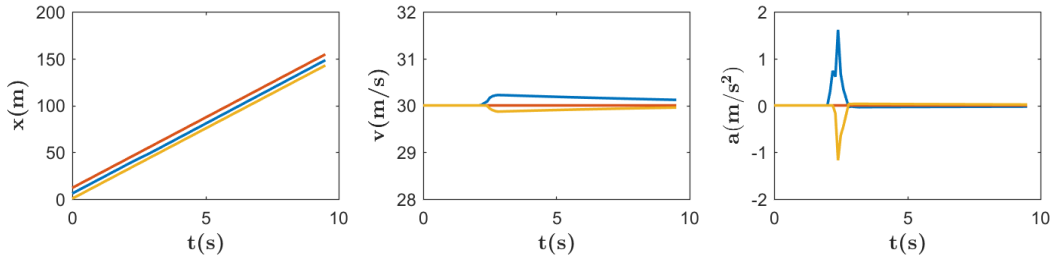


Figure 4.13: The trajectories obtained by optimization-based approach with highway speed and tight inter-vehicle distance are shown.

Table 4.4: Computation time in seconds for various sampling rates

Sampling Rate (Hz)	**Average** (s)	**Max.** (s)
50	0.30	0.9
100	0.42	1.91
200	1.04	5.24

switches as discussed earlier. However, the optimization-based approach yields trajectories with highway speed 30m/s and tight inter-vehicle distance 0.2m. The results are shown in Fig. 4.13. In summary, the motion primitive approach does not provably enforce the collision avoidance constraints. Furthermore, to design tight mini platoons at highway speed, the proposed optimization-based approach is simplified compared to motion primitives approach in which extensive tuning is required for all the switches and all the possible parameters.

Path Following

In this section, the MPC path-follower controller (4.14) is simulated in closed loop with dynamic model (4.1). The results are shown for a lane-change maneuver selected from the look-up table. The lane-change maneuver is planned by the motion-planner and path-follower follows the pre-computed motion. The results are reported in Fig. 4.14 for various sampling rates including $50Hz$, $100Hz$ and $200Hz$. The top plot shows the path in xy plane. The target trajectory (obtained by motion-planner) is shown with red dashed line. The gray, blue and pink plots are the results of path-follower controller with different sampling rates. The second plot shows the velocity tracking, in which the red dashed line is the target trajectory obtained by motion planner. The third and forth plots are acceleration and steering angle, respectively, which are obtained by the MPC path-follower. The results show that path tracking and velocity tracking performance are not affected by changing the sampling rate. However, 4.4 compares the average and maximum of the computation time for different sampling rates. As seen, the average of computation time is reduced for lower sampling rate.

In addition, six different simulations have been run (with sampling rate of 50 Hz) and the average and maximum of computation time of the controller is reported (in seconds) at

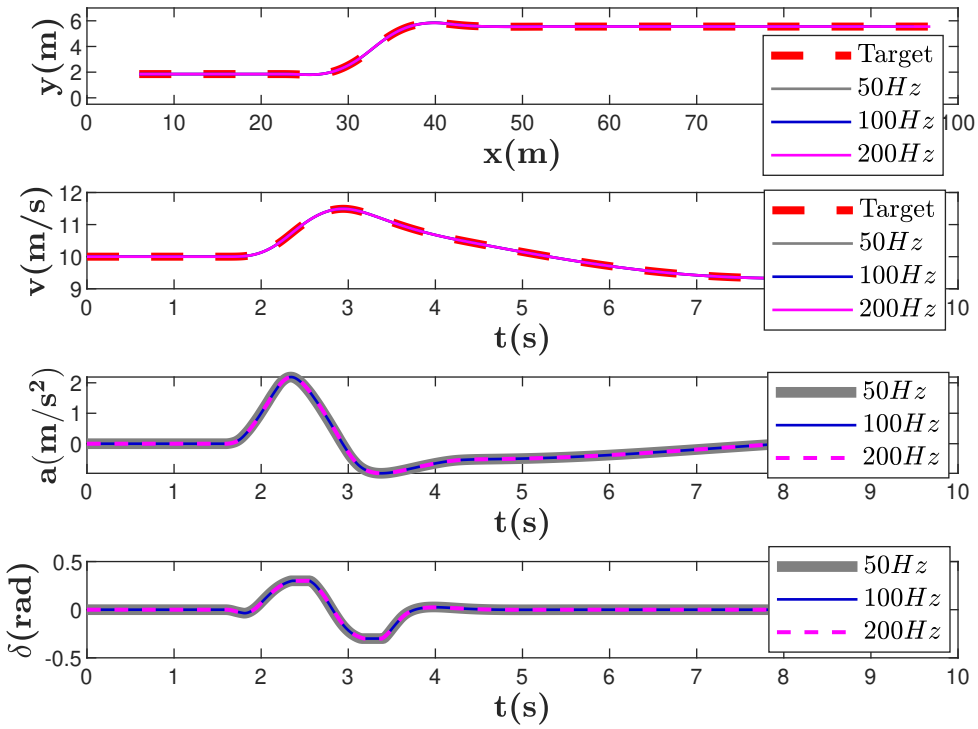


Figure 4.14: The closed-loop simulation of MPC path-following controller is shown for various sampling times.

Table 4.5. These results are reported by running the simulation on a Surface Book laptop with Intel(R) Core(TM) i7-6600U CPU @2.81 GHz and 16.0GB RAM in MATLAB. The total average of the computation time of the MPC controller is $0.32(s)$ and the maximum is $0.61(s)$. Note that these values can be reduced dramatically (an order of magnitude) if the controller's dynamic model (4.14a) is linearized around the given target trajectory. The linearized version of (4.14) can be solved in real-time.

4.8 Conclusion

An architecture for autonomous navigation of multi-lane platoons on public roads is presented. The architecture is composed of an offline motion-planning system and an online hierarchical control system, which consists of TOS, decision-maker and path-follower. The motion-planner avoids collisions among the vehicles within the platoon, but does not consider the collisions with surrounding vehicles outside the platoon. However, decision-maker checks the possible collisions between the planned reconfiguration maneuver and the future

Table 4.5: Computation time in seconds for sampling rate of $50Hz$

Run #	**Average** (s)	**Max.** (s)
1	0.31	0.71
2	0.21	0.55
3	0.45	0.68
4	0.28	0.52
5	0.33	0.61
6	0.35	0.59
Total Avg.	0.32	0.61

planned trajectories of the surrounding vehicles shared via V2V communication. Once a feasible reconfiguration maneuver is selected by the decision-maker, it will be executed by the path-follower controller in real time. The simulation results demonstrate that a platoon of vehicles can form geometrically flexible and reconfigurable shapes in tight environment while moving at highway speed. It is shown that in the case of sudden change in the environment, like appearing an obstacle or slow traffic in one lane, the multi-lane platoon of vehicles can perform collaborative maneuvers and change their configuration to merge into faster lanes. The presented approach is compared with behavior-based planning, in which the formation and reconfiguration is achieved by a sequence of motion primitives. The results show that to design tight maneuvers for mini-platoons at highway speed the presented optimization-based method is simplified compared to the motion primitive approach, which requires extensive tuning for the switches and parameters. The future work will be robustification of the planning scheme by handling the uncertainty caused by model mismatch, sensor measurements and communication delays and using closed-loop policies instead of open-loop ones.

Chapter 5

Distributed Coordination

5.1 INTRODUCTION

This chapter focuses on distributed coordination between multiple robots, especially in tight environments (e.g., highway lanes, parking lots, warehouses, canals, etc.), where the robots have reduced maneuver space. The centralized coordination scheme introduced in the previous chapter solves a large optimization to simultaneously optimize the actions for all the robots. However, the centralized coordination is computationally intensive and not suitable for real-time implementations. In this chapter, starting from the centralized coordination problem, it is shown how to exploit the dual structure of the problem to split the optimization into smaller sub-problems well-suited for online applications.

To ensure safe navigation of connected and autonomous robots, robots have to closely interact with their neighbors to avoid collisions and reach their goal. They have to efficiently generate a safe coordination strategy. In particular, this chapter is focused on efficient, safe and coordinated multi-robot trajectory planning (the interested reader can refer to [78] for an overview of the state of the art)

Contributions

A nonlinear model predictive control (NMPC) to plan collision-free trajectories to coordinate the robots. A polytopic representation of the individual robot is used and the collision avoidance problem as the problem of finding the minimum distance between two polytopes is formulated. To incorporate this collision avoidance strategy in the NMPC formulation, the proposed method relies on duality theory [80]. The minimum distance collision-avoidance constraints between each pair of robots is reformulated as a feasibility test (with associated collision-avoidance variables) that can be included within the constraints of the NMPC problem. Solving the NMPC problem in a centralized way (Fig. 5.1.a) is computationally intensive, so a distributed algorithm is introduced to solve the NMPC problem in a distributed way (Fig. 5.1.b), which is computationally efficient compared to centralized formulation and is suitable for real time applications.

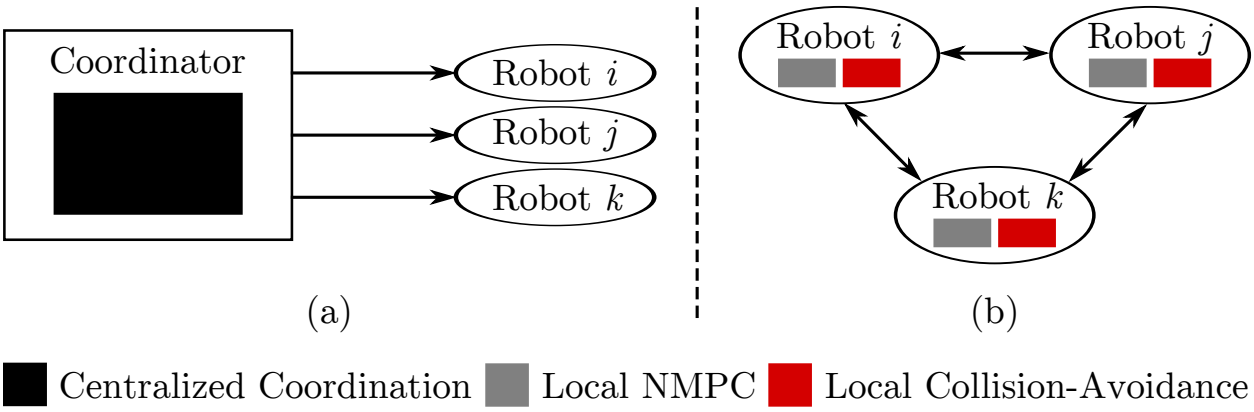


Figure 5.1: Centralized design (a) vs. the proposed distributed design (b) for multi-robot coordination.

In order to split the centralized problem into distributed sub-problems, the centralized formulation must be separable. Although the dynamic models of the robots are decoupled, but the collision avoidance constraints are coupled among them. To break the coupling, an alternating optimization approach is used to decompose the centralized problem as local minimization problems performed by alternating between two different optimizations (Fig. 5.1.b): *(i)* a collision avoidance optimization (red boxes in Fig. 5.1.b) that computes the predicted collision-avoidance variables, given the latest predicted intention of each pair of robots, and *(ii)* local NMPC optimizations (grey boxes in Fig. 5.1.b) that update the robot states, given the latest predicted collision-avoidance variables. The advantage of this decomposition is that each collision avoidance optimization solves efficiently (in milliseconds) convex problems of fixed dimension and the local NMPC problems have always a fixed number of decision variables (the local robot states), compared to the centralized problem. Also the error caused by relying on open-loop predicted trajectories of neighbor robots is quantified in distributed approach. It is shown that this error is bounded (and small) and a strategy is proposed to account for this error in the local NMPC problem formulation. Finally, the proposed method is validated for the autonomous navigation of a platoon of connected vehicles on a highway setting comparing its performance with a centralized implementation. In platooning both road geometry and platoon geometry restrict the motion of the vehicles within the platoon. Hence, the vehicles must coordinate in a *tight* environment. To allow navigation at tight spaces, the proposed approach models the road structure and the vehicles dimensions, as exact sizes with no approximation or enlargement. Also the results are demonstrated for a coordination scenario of a heterogeneous team of robots with different polytopic shapes.

Related Work

Classical methods for multi-robot coordination either use reactive strategies (such as potential fields [61, 68, 30], dynamic window [29], and velocity obstacles [24, 71]), assume a priority order [13], or rely on a scheduling [10] for the robots. These methods, however, do not explicitly consider the interaction among the robots. Learning-based methods [5, 12, 33, 41, 17] and constrained-optimization approaches can be used to take these interactions into account. This work fits in this last category and relies on tools from control and optimization to model the interactions among the robots to avoid collisions. Distributed constrained-optimization designs have been proposed for example in [39, 72, 4, 16, 55, 23]. The authors in [39] present a decentralized model predictive control (MPC) formulation for multi-robot coordination that relies on invariant-set theory and mix-integer linear programming (MILP). The authors in [72] propose a distributed MPC design for formation control using the alternating direction method of multipliers (ADMM) and separating hyperplanes for collision avoidance. The authors in [4] use a potential cost function and collision-avoidance constraints to formulate a distributed MPC problem, in which the collision avoidance constraints can be either linearized or formulated using integer variables. In addition, the authors rely on motion primitives to account for robot kinematic and dynamic constraints. The authors in [16, 55] present distributed MPC approaches that rely on ADMM to decompose the (linearized) coordination problem. The authors in [23] propose a distributed nonlinear MPC formulation with non-convex collision avoidance constraints.

Compared to [39], the proposed approach does not require the solution of a MILP problem that can be computationally expensive to solve. In addition, compared to [4, 16, 55] the proposed method does not require any linearization (which could reduce the solution space of the problem) of the collision-avoidance constraints. Also compared to [4], the proposed approach does not require the use of motion primitives (the robot dynamics are directly included in the NMPC formulation). Compared to [72], the proposed strategy allows to specify a desired distance between the robots, instead of using separating hyperplanes. Inspired by [80, 79], the proposed method uses dual optimization to formulate the collision avoidance constraints. Compared to [80, 79], however, the proposed method exploits the structure of the coordination problem to solve it in a distributed fashion.

5.2 PRELIMINARIES

The needed definitions and notations are provided below.
Robots and Neighbor Robots: The set of M cooperative robots is defined as $\mathcal{V} := \{1, 2, ..., M\}$. Each robot is identified through its index $i \in \mathcal{V}$. Throughout this paper, the superscript i denotes the ith robot. The neighbor set of robot i is denoted as $\mathcal{N}_i$ and represents all the robots that are in the communication range of Robot i.
Polytopic Description of Robot Pose: Robot pose or the region occupied by the robot can be described as a convex set defined by a polytope $\mathcal{P}$. Polytopes are described as the intersection

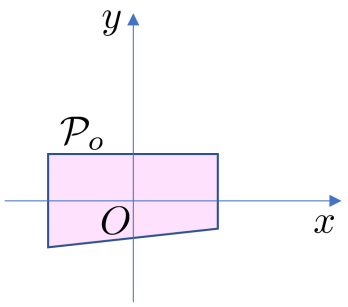


Figure 5.2: Each robot is represented as a polytopic set.

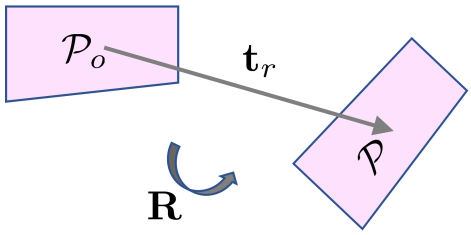


Figure 5.3: The polytopic set that represents the robot undergoes rotation and translation.

of a set of half-spaces and are defined as a set of linear inequalities. The initial pose of the robot is represented as $\mathcal{P}_o$, depicted in Fig. 5.2. As the robot travels, $\mathcal{P}_o$ undergoes affine transformations including rotation and translation shown in Fig. 5.3. Hence $\mathcal{P} = \mathbf{R}\mathcal{P}_o + \mathbf{t}_r$, where $\mathbf{R} : \mathbb{R}^{n_z} \to \mathbb{R}^{n \times n}$ is an orthogonal rotation matrix, $\mathbf{t}_r : \mathbb{R}^{n_z} \to \mathbb{R}^n$ is the translation vector, n_z is the dimension of the robot state $\mathbf{z}$, and n is dimension of the space which is 2 for 2D- and 3 for 3D planning.

Static and Dynamic Obstacles: $\mathcal{S} = \{1, ..., n_\mathcal{S}\}$ is the set of $n_\mathcal{S}$ static obstacles. $\mathcal{O}^r$ is the r-th static obstacle, $r \in \mathcal{S}$. Each static obstacle is modeled as a polytopic set. The collision avoidance between robot i and static obstacle r is defined as $\mathcal{P}^i \cap \mathcal{O}^r = \emptyset$. $\mathcal{N}_i$ is the set of dynamic obstacles for robot i. To avoid collision between Robots i and j, the intersection of their polytopic sets must be empty, $\mathcal{P}^i \cap \mathcal{P}^j = \emptyset$.
MPC Scheme: MPC is useful for online local motion planing in uncertain and dynamic environment because it is able to re-plan according to the new available information. MPC relies on the receding-horizon principle. At each time step it solves a constrained optimization problem and obtains a sequence of optimal control inputs that minimize a desired cost function J, while considering dynamic, state, and input constraints, over a fixed time horizon. Then, the controller applies, in closed-loop, the first control-input solution. At the next time step, the procedure is repeated. Throughout this work, $(\cdot|t)$ indicates the values along the entire planning horizon N, predicted based on the measurements at time t. For example $\mathbf{z}(\cdot|t)$ represents the entire state trajectory along the horizon $[\mathbf{z}(1), \mathbf{z}(2), ..., \mathbf{z}(N)]$ predicted

at time t. The bar notation $\bar{(\cdot)}$ represents constant known values.
MPC Cost Function J: The MPC cost is $J := \sum_{i\in\mathcal{V}} J^i$, where J^i is the local objectives of each robot. Each J^i can be designed according to the local-robot planning and control objectives. For example, the local costs can be specified to reach a goal set or to reduce the deviation from a global reference path (which is not collision-free) computed using high-level planning methods (e.g., A* or RRT*) as proposed in [58, 62, 9, 27] for single-robot local motion planning.

5.3 CENTRALIZED COORDINATION

The multi-robot coordination can be considered as a motion-planning problem and formulated as a centralized MPC optimization problem that computes collision-free trajectories for all the robots, simultaneously. The optimization problem is formulated in the NMPC framework as follows

$$
\begin{aligned}
\min_{\mathbf{u}^i(\cdot|t)} \quad & \sum_{i=1}^{M} J^i(\mathbf{z}^i, \mathbf{u}^i) && \text{(5.1a)}\\
\text{subject to} \quad & \mathbf{z}^i(k+1|t) = f(\mathbf{z}^i(k|t), \mathbf{u}^i(k|t)), && \text{(5.1b)}\\
& \mathbf{z}^i(0|t) = \mathbf{z}^i(t), && \text{(5.1c)}\\
& \mathbf{z}^i(k|t) \in \mathcal{Z},\ \mathbf{u}^i(k|t) \in \mathcal{U}, && \text{(5.1d)}\\
& \mathcal{P}(\mathbf{z}^i(k|t)) \cap \mathcal{O}^r = \emptyset,\ r \in \mathcal{S}, && \text{(5.1e)}\\
& \mathcal{P}(\mathbf{z}^i(k|t)) \cap \mathcal{P}(\mathbf{z}^j(k|t)) = \emptyset, \quad i \neq j && \text{(5.1f)}\\
& \forall i \in \mathcal{V},\ j \in \mathcal{N}_i, \text{and } k \in \{1, 2, .., N\}.
\end{aligned}
$$

In the formulation above, $\mathbf{u}^i(\cdot|t) = [u^i(k|t), ..., u^i(k+N-1|t)]$ denotes the sequence of control inputs over the MPC planning horizon N for ith robot. $\mathbf{z}^i(k|t)$ and $\mathbf{u}^i(k|t)$ variables of ith robot at step k are predicted at time t. The function $f(\cdot)$ in (5.1b) represents the nonlinear (dynamic or kinematic) model of the robot, which is discretized using Euler discretization. $\mathcal{Z}$, $\mathcal{U}$ are the state and input feasible sets, respectively. These sets represent state and actuator limitations. Constraints (5.1f) represent the collision-avoidance constraints between the ith robot and all the neighboring robots within the communication radius. This representation is time-varying and is a function of the robot state at each time step. The remainder of this section details the derivation of constraints (5.1e) and (5.1f). Note that the centralized NMPC problem (5.1), might get infeasible. However, persistent feasibility of (5.1) can be guaranteed by computing the reachable set. The focus of this chapter is to reformulate the centralized problem (5.1) into a distributed one, but the techniques to guarantee persistent feasibility can be incorporated into the proposed approach.

Collision Avoidance Reformulation

Consider two polytopic sets $\mathcal{P}_1$ and $\mathcal{P}_2$. The distance between these sets is given by the following primal problem

$$\text{dist}(\mathcal{P}_1, \mathcal{P}_2) = \min_{\mathbf{x},\mathbf{y}} \{\|\mathbf{x}-\mathbf{y}\|_2 | \mathbf{A}_1\mathbf{x} \leq \mathbf{b}_1, \mathbf{A}_2\mathbf{y} \leq \mathbf{b}_2\}, \tag{5.2}$$

where $\mathcal{P}_1 = \{\mathbf{x} \in \mathbb{R}^n | \mathbf{A}_1\mathbf{x} \leq \mathbf{b}_1\}$ and $\mathcal{P}_2 = \{\mathbf{y} \in \mathbb{R}^n | \mathbf{A}_2\mathbf{y} \leq \mathbf{b}_2\}$. The two sets do not intersect if $\text{dist}(\mathcal{P}_1, \mathcal{P}_2) > 0$. For motion-planning applications, however, the robots must keep a minimum safe distance $d_{\min}$ from each other and from the obstacles. Hence, the distance between their polytopic sets should be larger than a predefined minimum distance, $\text{dist}(\mathcal{P}_1, \mathcal{P}_2) \geq d_{\min}$.

Problem (5.2) is itself an optimization problem that cannot directly be used in Problem (5.1), because that would an optimization problem as the constraint of another optimization problem. To deal with this issue, one can rely on strong-duality theory. Building on [80], the dual problem can be solved instead of the primal problem (5.2). The dual problem is expressed as follows:

$$\begin{aligned} \text{dist}(\mathcal{P}_1, \mathcal{P}_2) := \max_{\boldsymbol{\lambda}_{12}, \boldsymbol{\lambda}_{21}, \mathbf{s}} \; & -\mathbf{b}_1^\top \boldsymbol{\lambda}_{12} - \mathbf{b}_2^\top \boldsymbol{\lambda}_{21} \\ \text{s.t. } & \mathbf{A}_1^\top \boldsymbol{\lambda}_{12} + \mathbf{s} = 0, \; \mathbf{A}_2^\top \boldsymbol{\lambda}_{21} - \mathbf{s} = 0, \\ & ||\mathbf{s}||_2 \leq 1, -\boldsymbol{\lambda}_{12} \leq 0, \; -\boldsymbol{\lambda}_{21} \leq 0, \end{aligned} \tag{5.3}$$

where $\boldsymbol{\lambda}_{12}$, $\boldsymbol{\lambda}_{21}$ and $\mathbf{s}$ are dual variables (the derivation of (5.3) from (5.2) is provided in the Appendix). The optimal value of the dual problem is the distance between $\mathcal{P}_1$ and $\mathcal{P}_2$ and is constrained to be larger than a desired minimum distance. Consequently, based on this insight the dual problem can be reformulated as the following feasibility problem: $\{\exists \boldsymbol{\lambda}_{12} \succeq 0, \boldsymbol{\lambda}_{21} \succeq 0, \mathbf{s} : -\mathbf{b}_1^\top \boldsymbol{\lambda}_{12} - \mathbf{b}_2^\top \boldsymbol{\lambda}_{21} \geq d_{\min}, \mathbf{A}_1^\top \boldsymbol{\lambda}_{12} + \mathbf{s} = 0, \mathbf{A}_2^\top \boldsymbol{\lambda}_{21} - \mathbf{s} = 0, \|\mathbf{s}\|_2 \leq 1\}$. This reformulation can be substituted to the collision-avoidance constraint (5.1f) in Problem (5.1). A similar reformulation can be derived for static obstacles (5.1e). Therefore, problem (5.1) can be rewritten as

$$\begin{aligned} \min_{\substack{\mathbf{u}^i(\cdot|t),\; \boldsymbol{\lambda}_{ij}(\cdot|t),\\ \boldsymbol{\lambda}_{ji}(\cdot|t),\; \mathbf{s}_{ij}(\cdot|t)}} \quad & \sum_{i=1}^{M} J^i(\mathbf{z}^i, \mathbf{u}^i) \\ \text{subject to} \quad & (5.1b), (5.1c), (5.1d), \\ & \big(-\mathbf{b}^i(\mathbf{z}^i(k|t))^\top \boldsymbol{\lambda}_{ij}(k|t) \\ & - \mathbf{b}^j(\mathbf{z}^j(k|t))^\top \boldsymbol{\lambda}_{ji}(k|t)\big) \geq d_{\min}, && (5.4a) \\ & \mathbf{A}^i(\mathbf{z}^i(k|t))^\top \boldsymbol{\lambda}_{ij}(k|t) + \mathbf{s}_{ij}(k|t) = 0, && (5.4b) \\ & \mathbf{A}^j(\mathbf{z}^j(k|t))^\top \boldsymbol{\lambda}_{ji}(k|t) - \mathbf{s}_{ij}(k|t) = 0, && (5.4c) \\ & \boldsymbol{\lambda}_{ij}(k|t), \boldsymbol{\lambda}_{ji}(k|t) \geq 0, \|\mathbf{s}_{ij}(k|t)\|_2 \leq 1, \\ & \forall i \in \mathcal{V}, \; j \in \mathcal{N}_i, \text{ and } k \in \{1, 2, .., N\}, \end{aligned}$$

where $\mathbf{A}^i$ and $\mathbf{b}^i$ are functions of $\mathbf{z}^i(k|t)$ and represent the polytopic set of ith vehicle at step k predicted at time t. Similarly $\mathbf{A}^j$ and $\mathbf{b}^j$ denote the polytopic set of jth robot which belongs to neighbor set $\mathcal{N}_i$. The dual variables $\boldsymbol{\lambda}_{ij}$, $\boldsymbol{\lambda}_{ji}$ and $\mathbf{s}_{ij}$ are coupled through the collision avoidance constraint between robot i and robot j. For space limitation the static obstacle avoidance constraint (5.1e) is removed in the above formulation and only the collision avoidance among robots are formulated. However, it can be added using dual reformulation. Note that the dual variable $\mathbf{s}_{ij}$ is equivalent to the variable $\mathbf{s}$ in (5.3). The new variable $\mathbf{s}_{ij}$ is introduced in (5.4) to distinguish for example, $\mathbf{s}_{12}$ from $\mathbf{s}_{13}$, but $\mathbf{s}_{12}$ is identical to $\mathbf{s}_{21}$ according to (5.3). Therefore, the variables $\mathbf{s}_{ij}$, $\mathbf{s}_{ji}$ and $\mathbf{s}$ are all identical vectors ($\mathbf{s}_{ij} = \mathbf{s}_{ji} = \mathbf{s} \in \mathbb{R}^n$) with dimension of n that is the dimension of the space which is 2 for 2D- and 3 for 3D planning. The geometric interpretation of the vector $\mathbf{s}$ is discussed in the next section.

Remark 2 *The required minimum distance between the robots d_{min}, which can be chosen as a design parameter. In practice, d_{min} should be determined based on the quantification of uncertainty of physical models and stochastic measurement errors.*

5.4 DISTRIBUTED COORDINATION

Problem (5.4) simultaneously optimizes over all the robots' states $\mathbf{z}^i$ and the collision avoidance variables $\boldsymbol{\lambda}_{ij}, \boldsymbol{\lambda}_{ji}, \mathbf{s}_{ij}$ (for all $i = 1, ..., M$, $j \neq i$), that is, the number of variables to optimize is proportional to the number of robots. This is computationally expensive when M is large, making the centralized formulation not scalable with the number of robots. The goal is to remove the need of a central coordinator and make the problem scalable with the number of robots (allowing the robots to coordinate and locally solve smaller sub-problems in parallel).

By looking at the structure of Problem (5.4), one can notice that the collision avoidance constraints (5.4a)-(5.4c) create a coupling among the robots. In addition, Constraint (5.4a) creates a nonlinear coupling between the state variables $\mathbf{z}^i$,$\mathbf{z}^j$ and the collision avoidance variables $\boldsymbol{\lambda}_{ij}$,$\boldsymbol{\lambda}_{ji}$. To break-up these couplings and devise the proposed distributed algorithm, one can rely on the dual structure originally used to formulate Problem 5.4 and on the ability of MPC to generate predictions. In particular, the key idea is to solve Problem (5.4) by using an alternating optimization scheme, in which the central coordinator is replaced by two independent optimizations that perform alternating optimization of the dual variables (associated with the collision avoidance constraints) and of primal state variables, respectively, as Algorithm 3 details.

In Algorithm 3, first the dual variables over the NMPC horizon are initialized, then the first optimization step (NMPC optimization) optimizes the state variables $\mathbf{z}^i$,$\mathbf{z}^j$ over the horizon, while keeping the dual variables $\boldsymbol{\lambda}_{ij}$,$\boldsymbol{\lambda}_{ji}, \mathbf{s}_{ij}$ fixed. The second optimization step (CA optimization) optimizes the dual variable while keeping the state variables fixed. These two optimizations are detailed below.

NMPC optimization

At time t, each robot i independently computes its own state $\mathbf{z}^i$ trajectory, given the dual variables over the horizon $[\boldsymbol{\lambda}_{ij}(1), \ldots, [\boldsymbol{\lambda}_{ij}(N)]$, $[\boldsymbol{\lambda}_{ji}(1), \ldots, \boldsymbol{\lambda}_{ji}(N)]$ and $[\mathbf{s}_{ij}(1), \ldots, \mathbf{s}_{ij}(N)]$. For each robot $i \in \mathcal{V},\ j \in \mathcal{N}_i$, the NMPC optimization is given by:

$$
\begin{aligned}
\min_{\mathbf{u}^i(\cdot|t)} \quad & J^i(\mathbf{z}^i, \mathbf{u}^i) \\
\text{subject to} \quad & (5.1b), (5.1c), (5.1d), \\
& \big(- \mathbf{b}^i(\mathbf{z}^i(k|t))^\top \bar{\boldsymbol{\lambda}}_{ij}(k|t) \\
& - \bar{\mathbf{b}}^j(\bar{\mathbf{z}}^j(k|t))^\top \bar{\boldsymbol{\lambda}}_{ji}(k|t)\big) \geq d_{\min}, \qquad (5.5a) \\
& \mathbf{A}^i(\mathbf{z}^i(k|t))^\top \bar{\boldsymbol{\lambda}}_{ij}(k|t) + \bar{\mathbf{s}}_{ij}(k|t) = 0, \qquad (5.5b) \\
& \text{for all } k \in \{1, 2, .., N\},
\end{aligned}
$$

where the bar notation $\bar{(\cdot)}$ represents constant known values and $\mathbf{A}^i$, $\mathbf{b}^i$ are the polytopic representation of the ith robot and are functions of $\mathbf{z}^i$. The optimized trajectory is then shared with the collision avoidance optimization (shifted in time according to step ⑤ of Algorithm 3 to account for the 1-step delay in the calculation of the collision-avoidance strategies). Problem (5.5) can be solved in parallel by each robot. In this optimization, the collision-avoidance variables $\boldsymbol{\lambda}_{ij}, \boldsymbol{\lambda}_{ji}, \mathbf{s}_{ij}$ are considered as known values along the planning horizon. Note that compared to the centralized formulation, the only decision variable to optimize in the NMPC optimization is the ith robot state $\mathbf{z}^i$ (i.e., the number of decision variables in the local problem formulations is constant).

Collision Avoidance (CA) optimization

Each robot pair of $i, j \in \mathcal{N},\ i \neq j$, computes the collision avoidance variables $\boldsymbol{\lambda}_{ij}, \boldsymbol{\lambda}_{ji}, \mathbf{s}_{ij}$. The CA optimization is given by

$$
\begin{aligned}
\max_{\substack{\boldsymbol{\lambda}_{ij}(\cdot|t),\\ \boldsymbol{\lambda}_{ji}(\cdot|t),\\ \mathbf{s_{ij}}(\cdot|t)}} \quad & -\bar{\mathbf{b}}^i(\bar{\mathbf{z}}^i(k|t))^\top \boldsymbol{\lambda}_{ij}(k|t) - \bar{\mathbf{b}}^j(\bar{\mathbf{z}}^j(k|t))^\top \boldsymbol{\lambda}_{ji}(k|t) \\
\text{subject to} \quad & \bar{\mathbf{A}}^i(\bar{\mathbf{z}}^i(k|t))^\top \boldsymbol{\lambda}_{ij}(k|t) + \mathbf{s}_{ij}(k|t) = 0, \qquad (5.6a) \\
& \bar{\mathbf{A}}^j(\bar{\mathbf{z}}^j(k|t))^\top \boldsymbol{\lambda}_{ji}(k|t) - \mathbf{s}_{ij}(k|t) = 0, \qquad (5.6b) \\
& \big(- \bar{\mathbf{b}}^i(\bar{\mathbf{z}}^i(k|t))^\top \boldsymbol{\lambda}_{ij}(k|t) \\
& - \bar{\mathbf{b}}^j(\bar{\mathbf{z}}^j(k|t))^\top \boldsymbol{\lambda}_{ji}(k|t)\big) \geq d_{\min}, \\
& \|\mathbf{s}_{ij}(k|t)\|_2 \leq 1, -\boldsymbol{\lambda}_{ij}(k|t) \leq 0, \qquad (5.6c) \\
& -\boldsymbol{\lambda}_{ji}(k|t) \leq 0, \text{ for all } i \in \mathcal{V},\ j \in \mathcal{N}_i, \\
& \text{for all } k \in \{1, 2, .., N\}.
\end{aligned}
$$

Each robot solves Problem (5.6) in parallel. This optimization assumes the state trajectories of the robots $\mathbf{z}^i$ to be fixed (obtained by the NMPC optimization and from the neighboring

Figure 5.4: In NMPC optimization, the dual variables are kept fixed and in CA optimization, the primal variables are kept fixed.

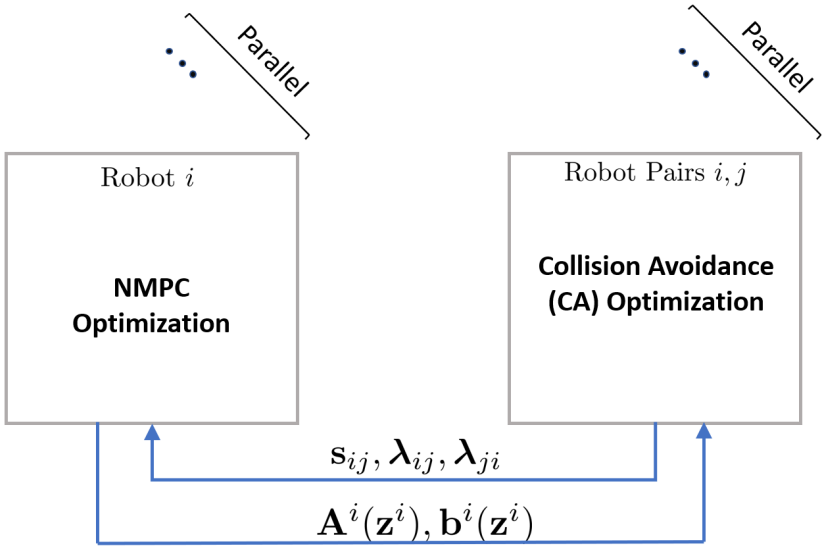


Figure 5.5: The robots independently solves their own NMPC optimization in parallel. The CA optimization is solved in parallel for each pair of robots.

robots according to step ⑦ of Algorithm 3). This problem can be solved efficiently (in the order of milliseconds). The proposed scheme is shown in Fig. 5.4 and Fig. 5.5.

Algorithm 3 is an alternating optimization scheme with the NMPC optimization (5.5) and the CA optimization (5.6). On one hand this alternating optimization scheme allows us to improve computation time of coordination strategy. On the other hand, due to this optimization scheme, the distributed NMPC (5.5) returns less tight trajectories (larger margins in the coordination) compared to the centralized NMPC (5.4). Solving CA optimization and substituting its solution in the NMPC optimization further restricts the constraints (5.5a)-(5.5b), since the dual variables are kept fixed (i.e., the NMPC optimizer has less degree of freedom in the computation of the local trajectories). In contrast, in the centralized NMPC

Algorithm 3 Distributed Coordination Algorithm

Initialize $[\mathbf{s}_{ij}(1),...,\mathbf{s}_{ij}(N)]$, $[\boldsymbol{\lambda}_{ij}(1),...,\boldsymbol{\lambda}_{ij}(N)]$, $[\boldsymbol{\lambda}_{ji}(1),...,\boldsymbol{\lambda}_{ji}(N)]$, $\forall i,j \in \mathcal{N}$ and $i \neq j$.
for $t = 0, 1, ..., \infty$ **do**
  for all Robot i, $i \in \mathcal{V}$ **do in parallel**
    Solve Problem (5.5)
    Compute the shifted state $[\mathbf{z}^i(2),...,\mathbf{z}^i(N),\mathbf{z}^i(N)]$.
    Compute the associated polytopic sets: $[\mathbf{A}^i(2),...,\mathbf{A}^i(N),\ \mathbf{A}^i(N)]$, $[\mathbf{b}^i(2),...,\ \mathbf{b}^i(N),\ \mathbf{b}^i(N)]$.
    Communicate to Robot j $(\forall j \in \mathcal{N}_i)$ the polytopic sets.
    Solve Problem (5.6) for each $j \in \mathcal{N}_i$
    Apply $\mathbf{u}^i_{\text{MPC}}$ to move forward.
  end for
end for

(5.4), the equivalent constraints (5.4a)-(5.4c) can be interpreted as the relaxed version of the constraints (5.5a)-(5.5b), since the dual variables are decision variables of the centralized problem.

Geometric Interpretation of Primal and Dual Variables

The dual variables have an interesting geometric interpretation. All these geometric meanings are obtained from the Karush–Kuhn–Tucker (KKT) conditions for problem (5.2). As the authors in [21] presented the following theorem can be proved from the KKT conditions.

Theorem 1 *The dual problem (5.3) representing the distance between two convex polytopes* $\mathbf{A}_1\mathbf{x} \preceq \mathbf{b}_1$ *and* $\mathbf{A}_2\mathbf{y} \preceq \mathbf{b}_2$*, can be interpreted geometrically as two parallel supporting hyperplanes with normal vector* $\|\mathbf{s}^*\| = 1$*, and affine terms* $-\mathbf{b}_1\boldsymbol{\lambda}^*_{12} > 0$ *and* $\mathbf{b}_2\boldsymbol{\lambda}^*_{21} < 0$*. The difference between affine terms as expressed in the dual objective represents the distance between two polytopes.*

As seen in Fig. 5.6, the top plots show the geometric representation of the primal formulation (5.2) in which the optimal solutions are $\mathbf{x}^*$ and $\mathbf{y}^*$ and the distance is defined as the *classical* Euclidean distance $\|\mathbf{x}^* - \mathbf{y}^*\|_2$ between the two sets. The bottom plots show the equivalent dual formulation (5.3) in which the optimal solutions are $\mathbf{s}^*$, $\boldsymbol{\lambda}^*_{12}$ and $\boldsymbol{\lambda}^*_{21}$ and the same distance between the two polytopic sets is defined as $-\mathbf{b}_1{}^\top\boldsymbol{\lambda}^*_{12} - \mathbf{b}_2{}^\top\boldsymbol{\lambda}^*_{21}$. As Fig. 5.6(e) depicts, the separating hyperplane between the two polytopic sets is always perpendicular to the minimum distance. Therefore $\mathbf{s}^*$, which is the normal vector of the separating hyperplane, is always parallel to the minimum distance. The normal vector $\mathbf{s}^*$ plays the role of the consensus variable between the robots. As discussed earlier, the vector $\mathbf{s}^*$ is shared between each pair of robots according to (5.3), so $\mathbf{s}_{ij} = \mathbf{s}_{ji}$. Furthermore, in Fig. 5.6(f), the green lines show the two supporting hyperplanes which are parallel to the separating hyperplane. The hyperplane $\mathbf{s}^{*\top}\mathbf{x} = -\mathbf{b}_1^\top\boldsymbol{\lambda}^*_{12}$ supports the set $\mathbf{x}$ or $(\mathcal{P}_1)$ at the point $\mathbf{x}^*$. Similarly the

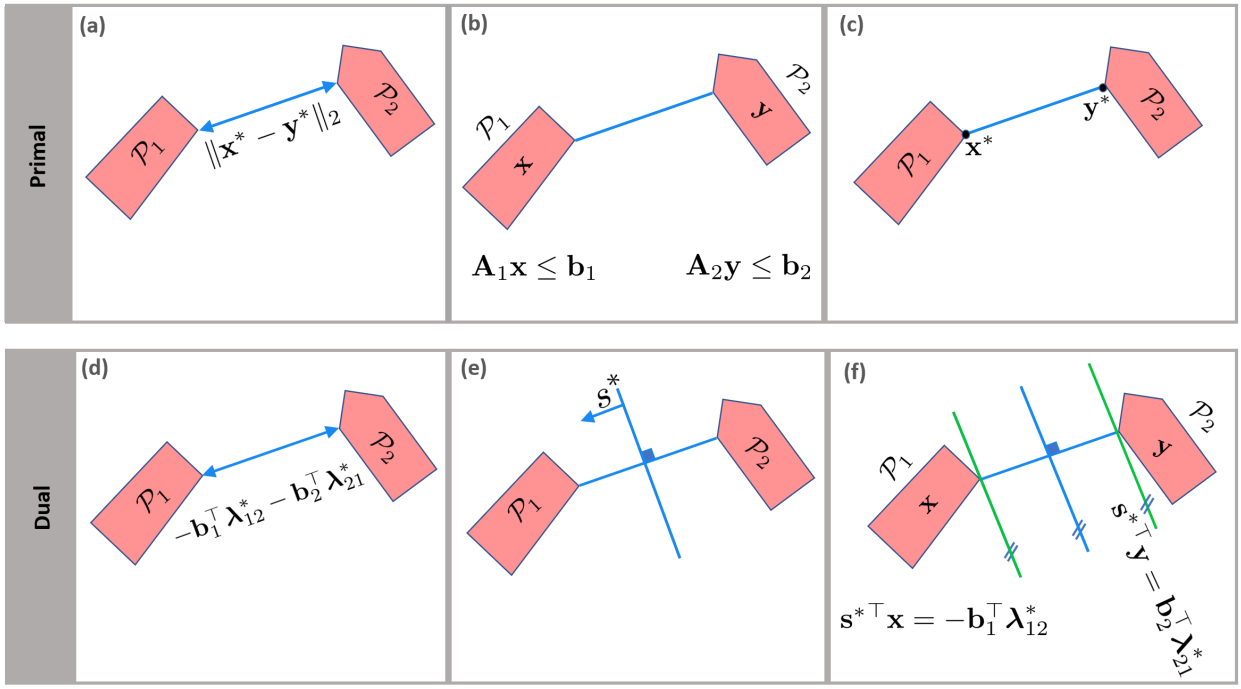


Figure 5.6: **Top: Primal** (a) minimum distance (b) polytope representation (c) optimal solutions. **Bottom: Dual**: (d) minimum distance (e) separating hyperplane (f) supporting hyperplanes.

hyperplane $\mathbf{s}^{*\top}\mathbf{y} = \mathbf{b}_2^\top \boldsymbol{\lambda}_{21}^*$ supports the set $\mathbf{y}$ or ($\mathcal{P}_2$) at the point $\mathbf{y}^*$. On the other hand, the primal (5.2) and dual (5.3) problems are convex and the Slater's condition is satisfied, so the strong duality holds [8]. Therefore, finding the shortest distance between two polytopic sets (primal problem (5.2)) is equivalent to finding the maximal separation, which is the maximum distance between a pair of parallel hyperplanes that supports the two sets (dual problem (5.3)) as shown in Fig. 5.6(f). The derivation of the KKT conditions upon which the geometric interpretations are obtained are provided in the Appendix (C).

5.5 NUMERICAL RESULTS FOR AUTONOMOUS DRIVING APPLICATION

In this section the centralized and distributed approaches are compared in terms of computation time (and cost) as the number of robots increase. It will be demonstrated that the proposed approach scales well with the number of robots. Section 5.5 describes the simulation setup. Sections 5.5 and 5.5 presents two different simulation scenarios, that are, *a)* a platoon formation and re-configuration and *b)* a heterogeneous team of robots with different polytopic shapes, respectively.

Simulation Setup

The proposed design is tested on a quad-core CPU Intel Core i7-7700HQ @ 2.80 GHz in MATLAB using the MATLAB Parallel Computing Toolbox to simulate the individual robots and communication exchanges among the robots. The optimization problems are modeled in YALMIP [46]. The Problems (5.4), (5.5) are solved using IPOPT [73], a state-of-the-art interior-point solver for non-convex optimization, and the Problem (5.6) is solved using Gurobi [34], an efficient quadratic programming solver. For all the scenarios the simulation results are presented as top view snapshots, as well as a series of state and action plots. The robots colors of the snapshots and plots are matched.

Platoon Formation

In this scenario, autonomous and connected vehicles merge into a platoon (train-like formation) and maintain a close inter-vehicular distance within the group. In platooning on public roads, both road geometry (lane width) and platoon geometry (longitudinal and lateral inter-vehicle spacing) restrict the motion of the vehicles within the platoon. Hence, the vehicles must coordinate in a *tight* environment. To allow navigation at tight spaces, it is essential to model the road structure and the vehicles dimensions, including length h and width w, as exact sizes with no approximation or enlargement.

Relevant NMPC Quantities

Each vehicle i is modeled within the platoon by using a nonlinear kinematic bicycle model [40] (a common modeling approach in path planning) described by the following equations (the superscript i is omitted when it is clear from the context):

$$\begin{aligned} &\dot{x} = v\cos(\psi+\beta),\ \ \dot{y} = v\sin(\psi+\beta),\\ &\dot{\psi} = \frac{v\cos\beta}{l_f+l_r}(\tan\delta),\ \ \dot{v} = a, \end{aligned} \tag{5.7}$$

where the ith vehicle state vector is $\mathbf{z} = [x, y, \psi, v]^\top$ (x, y, ψ, and v are the longitudinal position, the lateral position, the heading angle, and the velocity, respectively), the control input vector is $\mathbf{u} = [a, \delta]^\top$ (a and δ are the acceleration and the steering angle, respectively), $\beta := \arctan\left(\tan\delta(\frac{l_r}{l_f+l_r})\right)$ is the side slip angle, and l_f, l_r are the distance from the center of gravity to the front and rear axles, respectively. Using Euler discretization, the model (5.7) is discretized with sampling time Δt as

$$\begin{aligned} x(t+1) &= x(t) + \Delta t\ v(t)\cos(\psi(t)+\beta(t)),\\ y(t+1) &= y(t) + \Delta t\ v(t)\sin(\psi(t)+\beta(t)),\\ \psi(t+1) &= \psi(t) + \Delta t\ \frac{v(t)\cos\beta(t)}{l_f+l_r}(\tan\delta(t)),\\ v(t+1) &= v(t) + \Delta t\ a(t). \end{aligned} \tag{5.8}$$

The local costs are defined as

$$J(\mathbf{z},\mathbf{u}) = \sum_{k=t}^{t+N} \|(\mathbf{z}(k|t) - \mathbf{z}_{\text{Ref}}(k|t))\|^2_{Q_\mathbf{z}} + \sum_{k=t}^{t+N-1} (\|(\mathbf{u}(k|t))\|^2_{Q_\mathbf{u}} + \|(\Delta\mathbf{u}(k|t))\|^2_{Q_{\boldsymbol{\Delta u}}}), \tag{5.9}$$

where $\Delta\mathbf{u}$ penalizes changes in the input rate. $Q_\mathbf{z} \succeq 0$, $Q_\mathbf{u}, Q_{\boldsymbol{\Delta u}} \succ 0$ are weighting matrices, $\mathbf{z}_{\text{Ref}}$ is the reference trajectory generated by a high-level planner (and it is not obstacle-free). The ith vehicle dimensions are chosen as length $h = 4.5$m and width $w = 1.8$m. The road width is chosen as 3.7m (i.e., the standard highway-lane width in the United States). The speed is lower bounded by zero. The acceleration of each vehicle is bounded within $\pm\,4\text{m/s}^2$ and its rate change is bounded within $\pm\,1\text{m/s}^2$. The steering input is bounded within $\pm\,0.3$rad and its change within ± 0.2rad/s. The corresponding road region occupied by the ith vehicle is defined by a two-dimensional convex polytope $\mathcal{P}$. For each vehicle, the transformed polytope is defined as $\mathcal{P}(\mathbf{z}(t)) = \{p \in \mathbb{R}^2 | \mathbf{A}(\mathbf{z}(t))p \leq \mathbf{b}(\mathbf{z}(t))\}$, where $\mathbf{A}(\mathbf{z}(t))$ and $\mathbf{b}(\mathbf{z}(t))$ are defined as

$$\mathbf{A}(\mathbf{z}(t)) = \begin{bmatrix} \mathbf{R}(\mathbf{z}(t))^\top \\ -\mathbf{R}(\mathbf{z}(t))^\top \end{bmatrix}, \ \mathbf{R}(\mathbf{z}(t)) = \begin{bmatrix} \cos(\psi(t)) & -\sin(\psi(t)) \\ \sin(\psi(t)) & \cos(\psi(t)) \end{bmatrix}, \tag{5.10}$$

$$\mathbf{b}(\mathbf{z}(t)) = [h/2, w/2, h/2, w/2]^\top + \mathbf{A}(\mathbf{z}(t))[x(t), y(t)]^\top, \tag{5.11}$$

where $[x(t), y(t)]^\top$ is the center of gravity of the vehicle.

Platoon Formation Results

In this scenario, as seen in Fig. 5.7a, the vehicles are initially traveling in three different lanes and need to merge in one lane to form a train-like platoon, while maintaining the safe distance $d_{min} = 0.5$m from each other at all times (i.e., during the lane change maneuvers and afterwards). The formation scenario is tested for different configurations and initial conditions. Fig. 5.7b shows an example with four vehicles. The initial longitudinal coordinates for all the four vehicles are $[x^1(0), x^2(0), x^3(0), x^4(0)] = [11.5, 5.5, 0.5, 20]$ and the initial lateral coordinates are $[y^1(0), y^2(0), y^3(0), y^4(0)] = [1.85, 5.55, 1.85, 9.25]$. The planning horizon N is 0.75s, the sampling time Δt is 0.05s, and v_{Ref} is 15m/s. Fig. 5.7b represents the vehicles' states and actions. The longitudinal and lateral coordinates x and y, as well as heading angle ψ and velocity v for all the vehicles are shown in different colors which are matched with the colors in Fig. 5.7a.

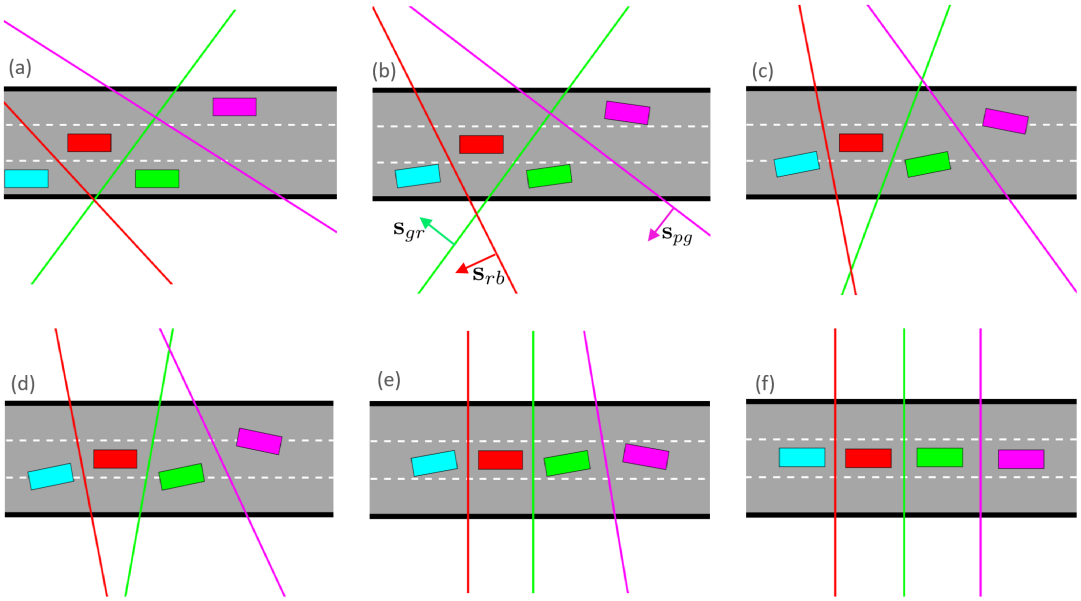
The control actions a and δ are also illustrated for all the vehicles. The acceleration and velocity plots highlights the collaborative behavior between the vehicles. In contrast to reactive approaches, such as velocity obstacles, the speed of each vehicle is not assumed constant and varies based on the interactions with the neighbors. For example, as seen in the acceleration plot, the blue vehicle brakes and the magenta vehicle accelerates to make enough gap for other vehicles to merge into the lane. This behavior is obtained by the solving the optimization problem and is not enforced explicitly. This collaborative behavior is fundamental

	Centralized		**Distributed**			
			NMPC		**CA**	
Num. of Vehicles	Avg.	Max.	Avg.	Max.	Avg.	Max.
2	1.3851	2.7482	0.1457	0.3621	0.0022	0.0024
			0.1102	0.3313	0.0021	0.0022
			Total Avg. = 0.1301			
3	2.7680	4.6817	0.1848	0.3933	0.0025	0.0028
			0.1206	0.3169	0.0024	0.0026
			0.1416	0.3470	0.0022	0.0023
			Total Avg. = 0.1514			
4	12.8331	28.7763	0.1919	0.4211	0.0030	0.0024
			0.1125	0.2602	0.0027	0.0029
			0.1842	0.3497	0.0024	0.0025
			0.2041	0.4195	0.0025	0.0023
			Total Avg. = 0.1757			

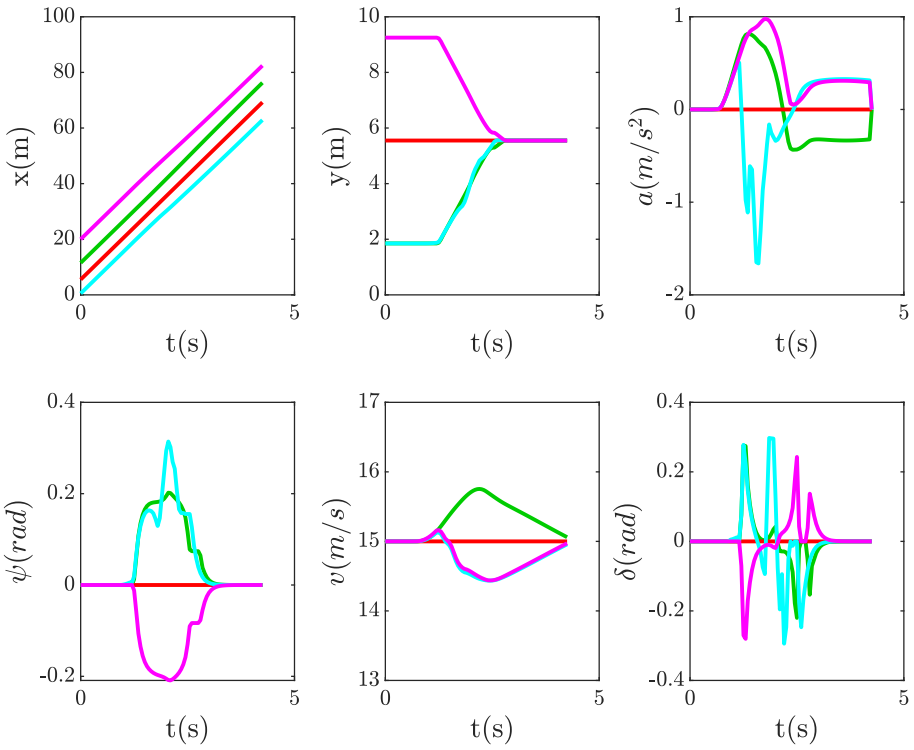
Table 5.1: Computation time (in second) for both centralized and distributed approaches are reported for two, three and four numbers of coordinated vehicles.

to keep the desired minimum distance and form the platoon. Ellipsoidal representations of the vehicles would have required a larger minimum distance (to fit each vehicle the axes of each ellipses would have been 6.36m and 2.54m, respectively, leading to a minimum distance, in the longitudinal direction for example, equal to $(2(6.36 - 4.5)\text{m} > d_{\min})$ leading to more conservative behaviors. Similar considerations hold for implementations based on potential fields. A potential field can be always included in the NMPC problem formulation to enforce clearance with respect to the other vehicles, but it would lead to more conservative behaviors (e.g., larger minimum distance between the vehicles) and additional tuning parameters. In Fig. 5.7a(a), in addition to snapshots, the separating hyperplanes between pink/green, green/red and red/blue pairs are shown and the normal of these separating hyperplanes are denoted as $\mathbf{s}_{pg}$, $\mathbf{s}_{gr}$ and $\mathbf{s}_{rb}$, respectively.

To evaluate the performance in terms of computation time of Algorithm 3, the formation of two, three and four vehicles in highway are simulated (varying the initial configuration of the vehicles to test the robustness of the tuning). The average and maximum computation time in second is compared for each step along the trajectory with the centralized implementation. Table 5.1 shows the results. While both centralized and distributed approaches ensure safe robot coordination, the maximum and average of computation time for centralized approach increases dramatically by adding a vehicle ($>> N$), compromising the safety of the approach in real-time applications (i.e., delays in the calculation of the planning strategy can lead to collisions). The distributed approach outperforms the centralized design by more than 2 orders of magnitude (by looking at the worst case scenario with 4 vehicles) and the computation time is reasonable for online implementation ($<< N$).



(a) Simulation snapshots with the separating hyperplanes between the vehicles.



(b) The state and input vehicles trajectories.

Figure 5.7: Four vehicles merge into a platoon in the center lane.

	Centralized	**Distributed**
Num. of Vehicles	Sum of Costs	Sum of Costs
2	0.0443	1.8957 1.6845
		Total Avg. = 1.7901
3	0.0985	6.0267 0.0000 5.6856
		Total Avg. = 3.9041
4	0.0321	6.8064 0.0000 12.1444 10.3252
		Total Avg. = 7.3190

Table 5.2: The total cost (sum of the closed-loop costs for the entire maneuver) is reported for centralized and distributed approaches for two, three and four numbers of coordinated vehicles.

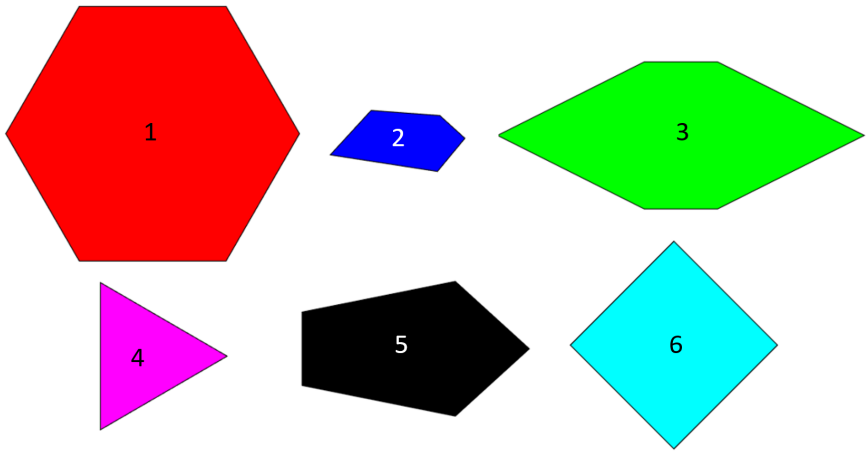


Figure 5.8: Polytopic Shapes

Heterogeneous Robots Reconfiguration and Tight Collision Avoidance

A team of robots with different polytopic shapes is considered as seen in Fig. 5.8. The robots are initially in a circular formation. A goal region is assigned to each robot. The robots should plan their motion based on the shared information communicated from other robots to reach their goals while avoiding collision with each other.

Relevant Problem Parameters

Each robot i is modeled by a nonlinear kinematic unicycle model which is similar to the previously described model (5.7)

$$\begin{aligned} \dot{x} &= v\cos(\psi), \\ \dot{y} &= v\sin(\psi), \\ \dot{\psi} &= \delta, \end{aligned} \tag{5.12}$$

where the ith robot state vector is $\mathbf{z} = [x, y, \psi]^\top$ and the control input vector is $\mathbf{u} = [v, \delta]^\top$ (the notations are the same as model (5.7). Using Euler discretization, the model (5.12) is discretized with sampling time Δt as

$$\begin{aligned} x(t+1) &= x(t) + \Delta t\ v(t)\cos(\psi(t)), \\ y(t+1) &= y(t) + \Delta t\ v(t)\sin(\psi(t)), \\ \psi(t+1) &= \psi(t) + \Delta t\ \delta(t), \end{aligned} \tag{5.13}$$

The local costs are defined as

$$J(\mathbf{z}, \mathbf{u}) = \sum_{k=t}^{t+N} \|(\mathbf{z}(k|t) - \mathbf{z}_{\text{Goal}})\|^2_{Q_\mathbf{z}} + \sum_{k=t}^{t+N-1} (\|(\mathbf{u}(k|t))\|^2_{Q_\mathbf{u}} + \|(\Delta\mathbf{u}(k|t))\|^2_{Q_{\boldsymbol{\Delta u}}}), \tag{5.14}$$

where $\Delta\mathbf{u}$ penalizes changes in the input rate. $Q_\mathbf{z} \succeq 0$, $Q_\mathbf{u}, Q_{\boldsymbol{\Delta u}} \succ 0$ are weighting matrices, $\mathbf{z}_{\text{Goal}}$ is the goal location that robot should reach.

The sampling time is 0.05s, minimum allowable distance $d_{\min}$ is 10cm, the velocity input is bounded within $\pm$ 4m/s and its rate change is bounded within $\pm$ 0.5m/s. The rate change of steering input is bounded within $\pm$ 0.5rad/s. The ith robot shape is defined by a two-dimensional convex polytope centered at the origin $\mathcal{P}_O = \{p \in \mathbb{R}^2 | \mathbf{A}_O\ p \leq \mathbf{b}_O\}$. As the robot moves, the polytopic shape undergoes rotation and translation. The transformed polytope is defined as $\mathcal{P}(\mathbf{z}(t)) = \{p \in \mathbb{R}^2 | \mathbf{A}(\mathbf{z}(t))p \leq \mathbf{b}(\mathbf{z}(t))\}$, where $\mathbf{A}(\mathbf{z}(t))$ and $\mathbf{b}(\mathbf{z}(t))$ are defined as $\mathbf{A}(\mathbf{z}(t)) = \mathbf{A}_O\mathbf{R}(\mathbf{z}(t))^\top$, $\mathbf{R}(\mathbf{z}(t)) = \begin{bmatrix} \cos(\psi(t)) & -\sin(\psi(t)) \\ \sin(\psi(t)) & \cos(\psi(t)) \end{bmatrix}$, $\mathbf{b}(\mathbf{z}(t)) = \mathbf{b}_O$ $+\mathbf{A}(\mathbf{z}(t))[x(t), y(t)]^\top$, where $[x(t), y(t)]^\top$ is the center of gravity of the vehicle.

Heterogeneous Robots Reconfiguration Results

Six robots with different polytopic shapes are considered as shown in Fig. 5.11 (a). A square (light blue), a hexagon (red), an irregular pentagon (dark blue), a hexagon (green), a triangle (pink), a pentagon (black) are representation of the robots and are located on a circle. The goal assigned to each robot is the other end of the diameter of the circle (i.e., each robot needs to swap its position with the one of the robot on the opposite side of the circle). For example in Fig. 5.11 (a), the goal for the red robot is the location of pink robot on the circle. So the pairs on the same diameter swap their locations. The pairs red/pink, dark blue/black and green/light blue swap, respectively, as shown in Fig. 5.9. The Fig. 5.9

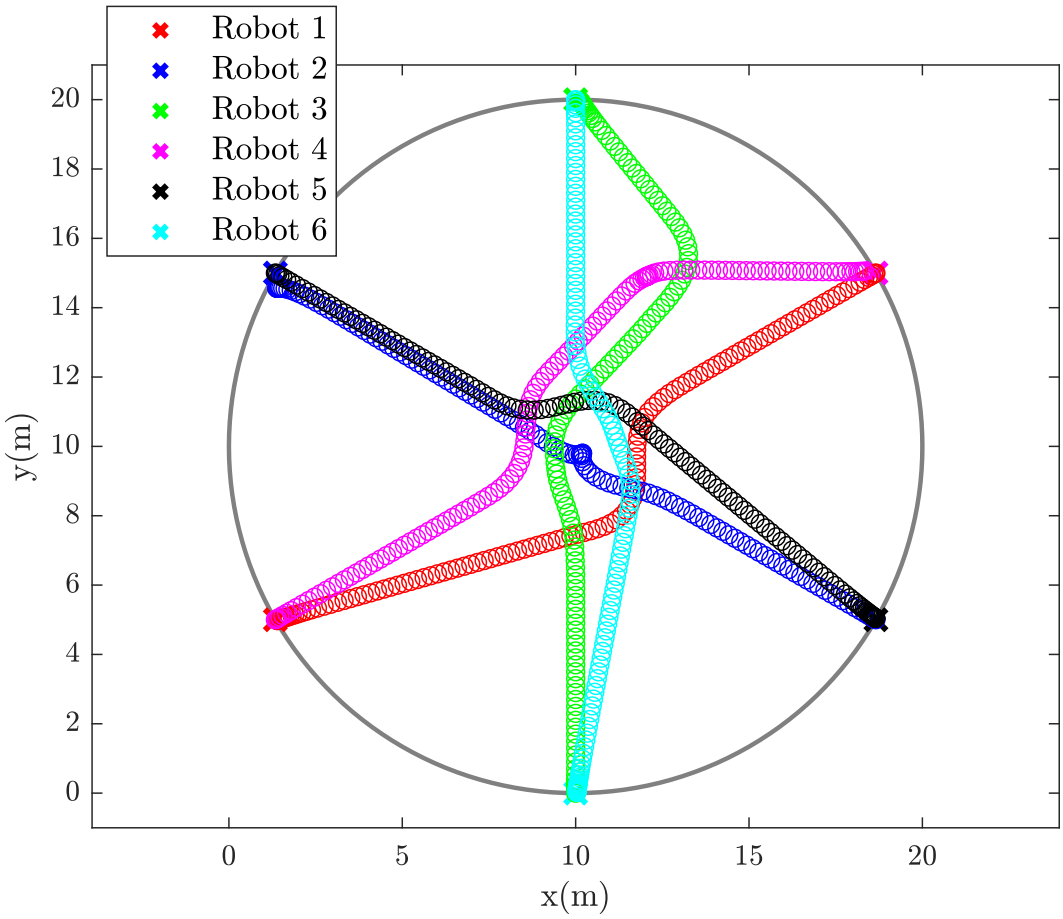


Figure 5.9: Trajectories of all the robots in x-y plane.

shows the position of each robot along the simulation in x-y plane. The Fig. 5.10 shows the control input trajectories and the Fig. 5.11 shows the snapshots of the simulation. The initial formation of robots is shown in snapshot (a), then (b)-(e) snapshots shown how they resolve the conflict among each other and (f) shows the final configuration, in which the pairs have swapped their locations. As seen, the robots reduce their speed to avoid collisions at the center of the circle. The open-loop predicted trajectories are shown with small circles for each robot.

Remark 3 *Note that the proposed coordination strategy is a local one and the performance can be affected by choosing different tuning parameters. For example, choosing a short planning horizon (which would help reduce the computation time even further) can cause deadlock situations, which can be preventable by increasing the horizon length. In addition, the proposed design requires communication among the robots and can tolerate delays up to the prediction horizon of the local NMPC formulations. If the delay is exceeded, fallback strategies are required (e.g., rely on local perception to understand the intentions of the neighboring*

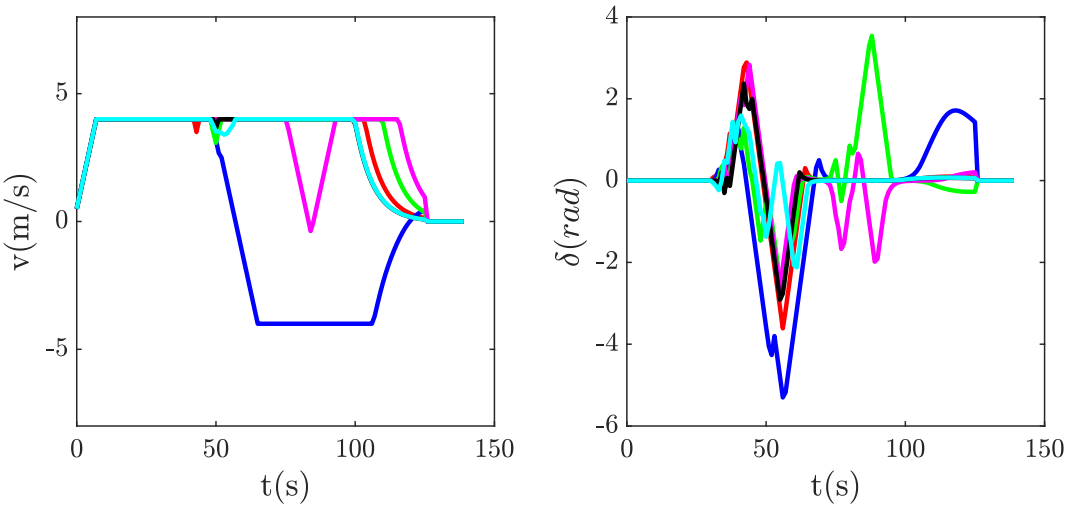


Figure 5.10: Control input trajectories of the six robots.

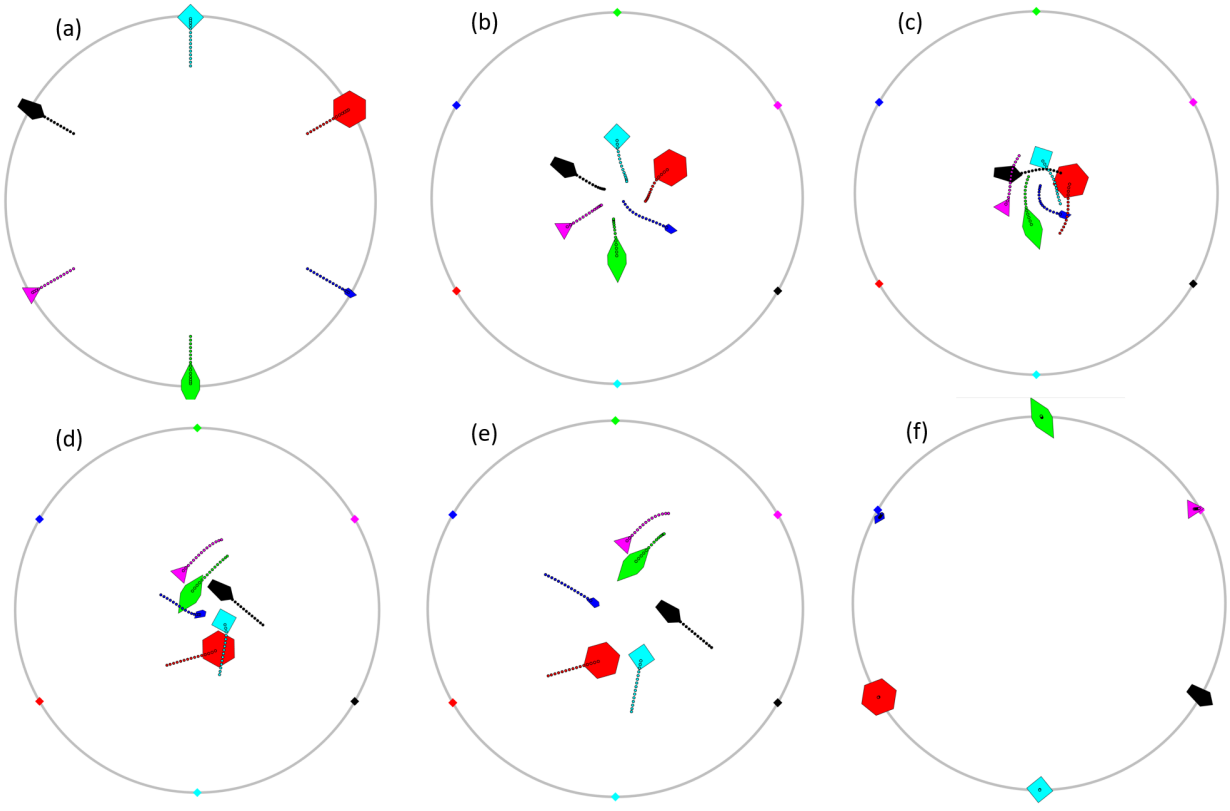


Figure 5.11: Snapshots of the reconfiguration of a team of 6 heterogeneous robots with different polytopic shapes are shown. The predicted open-loop trajectories are shown with small dots.

robots or detect possible faults). These strategies, however, are outside the scope of this work.

5.6 Prediction Error Bound

This section focuses on the discrepancy between distributed and centralized approaches caused by trajectory prediction error. In the centralized approach (5.1), since the coordinator chooses the optimal action for all the robots simultaneously, the approach does not rely on the open-loop trajectories of the neighbor robots (no communication is involved). However, in the distributed problem (5.5), each robot optimizes its own action based on the communicated information about the predicted (open-loop) trajectories of other robots. In this section, the trajectory prediction error is quantified and an upper bound on this error is established. To help the discussion, a simple simulation scenario is considered and all the results are presented for this scenario. In this scenario two cars, namely, car i and car j, are moving in the same direction using two different lanes when car i decides to overtake and merge into car j's lane, as shown in Fig. 5.12, (for more details on the model and formulation refer to Sec. 5.5).

Consider the open-loop trajectory of robot i predicted by robot i as

$$\boldsymbol{\tau}^i_{pi}(t) = \left[\ \mathbf{z}^i(0|t),\quad \mathbf{z}^i(1|t),\quad \ldots,\quad \mathbf{z}^i(N|t)\ \right]$$

and the shifted and augmented open-loop trajectory of robot i predicted by robot j as

$$\bar{\boldsymbol{\tau}}^i_{pj}(t-1) = \left[\ \bar{\mathbf{z}}^i(1|t-1),\quad \bar{\mathbf{z}}^i(2|t-1),\quad \ldots, \bar{\mathbf{z}}^i(N|t-1),\quad \bar{\mathbf{z}}^i(N|t-1),\ \right]$$

(the trajectory is shifted one step forward in time and augmented with the same last value). The subscript pi means predicted by robot i and similarly pj means predicted by robot j. Also $\boldsymbol{\tau}^j_{pj}(t)$ represents trajectory of robot j predicted by j at current time step t and $\bar{\boldsymbol{\tau}}^i_{pj}(t-1)$ denotes the trajectory of robot i predicted by robot j at the previous time step $t-1$.

The prediction error at each time step t, $e_{\text{predict}}(t)$ is defined as

$$e_{\text{predict}}(t) = \|\text{dist}_{pi}(t) - \text{dist}_{pj}(t)\|_2, \tag{5.15}$$

where dist_{pi} is the distance between robot i and robot j predicted by robot i (distance from the point of view of robot i) and, similarly, dist_{pj} is the distance between robot i and robot j predicted by robot j (distance from the point of view of robot j). The predicted distances are equivalent to

$$\begin{aligned} \text{dist}_{pi}(t) &\simeq \|\boldsymbol{\tau}^i_{pi}(t) - \bar{\boldsymbol{\tau}}^j_{pi}(t-1)\|_2, \\ \text{dist}_{pj}(t) &\simeq \|\boldsymbol{\tau}^j_{pj}(t) - \bar{\boldsymbol{\tau}}^i_{pj}(t-1)\|_2, \end{aligned} \tag{5.16}$$

and by substituting (5.16) in (5.15), the predicted error is equivalent to

$$e_{\text{predict}}(t) \simeq \| \left(\|\boldsymbol{\tau}^i_{pi}(t) - \bar{\boldsymbol{\tau}}^j_{pi}(t-1)\|_2 - \|\boldsymbol{\tau}^j_{pj}(t) - \bar{\boldsymbol{\tau}}^i_{pj}(t-1)\|_2 \right) \|_2. \tag{5.17}$$

Since the cars are rigid bodies (modeled as polytopic sets) and not just point masses, the exact minimum distance between the two polytopic sets is considered and dist_{pi} and dist_{pj} are defined using the notion of distance between the sets. As discussed earlier, the distance between the two polytopic sets i and j is defined as $-\mathbf{b}^{i\top}\boldsymbol{\lambda}_{ij} - \mathbf{b}^{j\top}\boldsymbol{\lambda}_{ji}$. However this value is different from the view of robot i and robot j, in distributed approach ($\text{dist}_{pi} \neq \text{dist}_{pj}$). According to distance constraint (5.5a), the distances from point of view of robot i and j can be written as

$$\text{dist}_{pi}(t) = -\mathbf{b}^{i}(t)^{\top}\bar{\boldsymbol{\lambda}}_{ij}(t-1) - \bar{\mathbf{b}}^{j}(t-1)^{\top}\bar{\boldsymbol{\lambda}}_{ji}(t-1), \tag{5.18}$$

$$\text{dist}_{pj}(t) = -\bar{\mathbf{b}}^{i}(t-1)^{\top}\bar{\boldsymbol{\lambda}}_{ij}(t-1) - \mathbf{b}^{j}(t)^{\top}\bar{\boldsymbol{\lambda}}_{ji}(t-1) \tag{5.19}$$

By substituting (5.18) and (5.19) in (5.15), the prediction error e_{predict} can be computed. After this quantification, an upper-bound on the prediction error is established. Note that in the centralized approach, the prediction error e_{predict} is zero since no communication is involved (the approach does not rely on the open-loop trajectory predictions of other robots) and $\text{dist}_{pi} = \text{dist}_{pj}$.

Theorem 2 *Let the minimum distance between robot i and j from the perspective of robot i and j be defined according to (5.18) and (5.19), respectively. By using Algorithm 1 to solve Problem (5.5), the prediction error is bounded, that is,*

$$\begin{aligned} e_{predict}(t) &= \|dist_{pi}(t) - dist_{pj}(t)\|_2 \\ &\leq c_i||\mathbf{b}^{i}(\mathbf{z}^{i}(t)) - \bar{\mathbf{b}}^{i}(\bar{\mathbf{z}}^{i}(t-1))||_2 \\ &+ c_j||\mathbf{b}^{j}(\mathbf{z}^{j}(t)) - \bar{\mathbf{b}}^{j}(\bar{\mathbf{z}}^{j}(t-1))||_2, \end{aligned} \tag{5.20}$$

where c_i and c_j are constant scalars. For vehicle applications with rectangular polytopic shapes c_i and c_j are equal to one. For general polytopic shapes

$$c_i = \sqrt{2}\|({\mathbf{A}_O}^{i\top})^{\dagger}\|_{\mathbf{F}} = \sqrt{2\sum_{i=1}^{r}\sigma_i^2},$$

where σ_i is the ith singular value of the matrix and r is its rank. Similarly, c_j has the same definition associated with matrix $(\mathbf{A}_O^{j\top})^{\dagger}$.

Proof 1 *See Appendix.*

To clarify the meaning of this time-varying upper-bound (right-hand side of (5.20)), the overtaking scenario in Fig. 5.12 is simulated and the minimum distances between the cars at each time step along the simulation is shown in Fig. 5.13 top plot. As seen in the top plot, dist_{pi}, the distance predicted by car i (light blue line) is different from dist_{pj}, the distance predicted by car j (pink line). The true distance is the actual distance between the cars

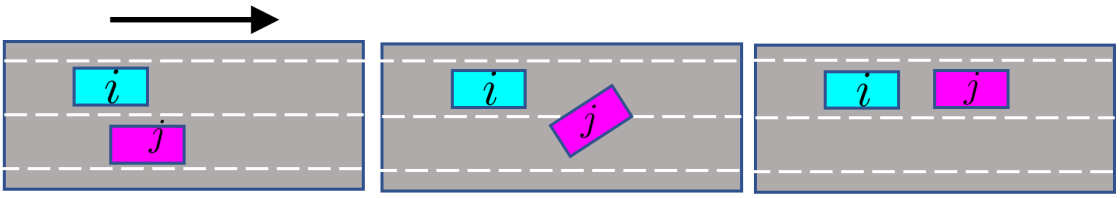


Figure 5.12: Overtaking maneuver: the top arrow indicates direction of motion.

(gray line). This difference in predicted distances is the prediction error which is bounded according to (5.20). The bottom plot in Fig. 5.13, shows that for overtaking scenario the prediction error e_{predict} (blue line) is bounded by the computed proposed upper bound (right-hand side of (5.20)) (red line). This upper bound means the prediction error in computing the distance between two cars is bounded by the deviation of the $\mathbf{b}^i$ and $\mathbf{b}^j$ vectors, between two iterations of the distributed algorithm. The distance in dual formulation is proportional to $\mathbf{b}^i$ and $\mathbf{b}^j$, which are right-hand side of polytopic set representation, as shown Fig. 5.6. The error is bounded and it is small, since the vector $\mathbf{b}$ does not change dramatically between two problem iterations.

In the remainder of this section, the following questions are addressed:

1. How tight is this bound compared to a trivial upper bound (computed from boundaries of the feasible set)?

2. For a given acceptable error on the states, how one can rewrite the upper-bound as a function of acceptable error on the states and enforce it as a constraint in NMPC problem (5.5)?

3. How can one normalize the upper-bound values?

1) To study how tight the upper-bound (5.20) is, it is compared with a trivial upper-bound. The trivial upper-bound is defined by maximizing the right-hand-side of (5.20). This maximization is done by substituting maximum boundary values of the feasible set $\mathcal{Z}$ in (5.18) and (5.19). The results for the same overtaking simulation is shown in Fig. 5.14. The top plot shows that prediction error is bounded by the proposed upper-bound. The bottom plot shows that the proposed upper-bound is considerably tighter than the trivial upper-bound.

2) For a given acceptable error on the states, a threshold for the upper-bound can be defined and the upper-bound (5.20) can be included in the NMPC formulation (5.5) to bound the prediction error in the distributed problem. For this application, the states are x,y and ψ. The maximum error on x, y and ψ are denoted as e_x, e_y and e_ψ, respectively and defined as

$$\begin{aligned} e_x &= |x(t) - \bar{x}(t-1)| \\ e_y &= |y(t) - \bar{y}(t-1)| \\ e_\psi &= |\psi(t) - \bar{\psi}(t-1)| \end{aligned} \tag{5.21}$$

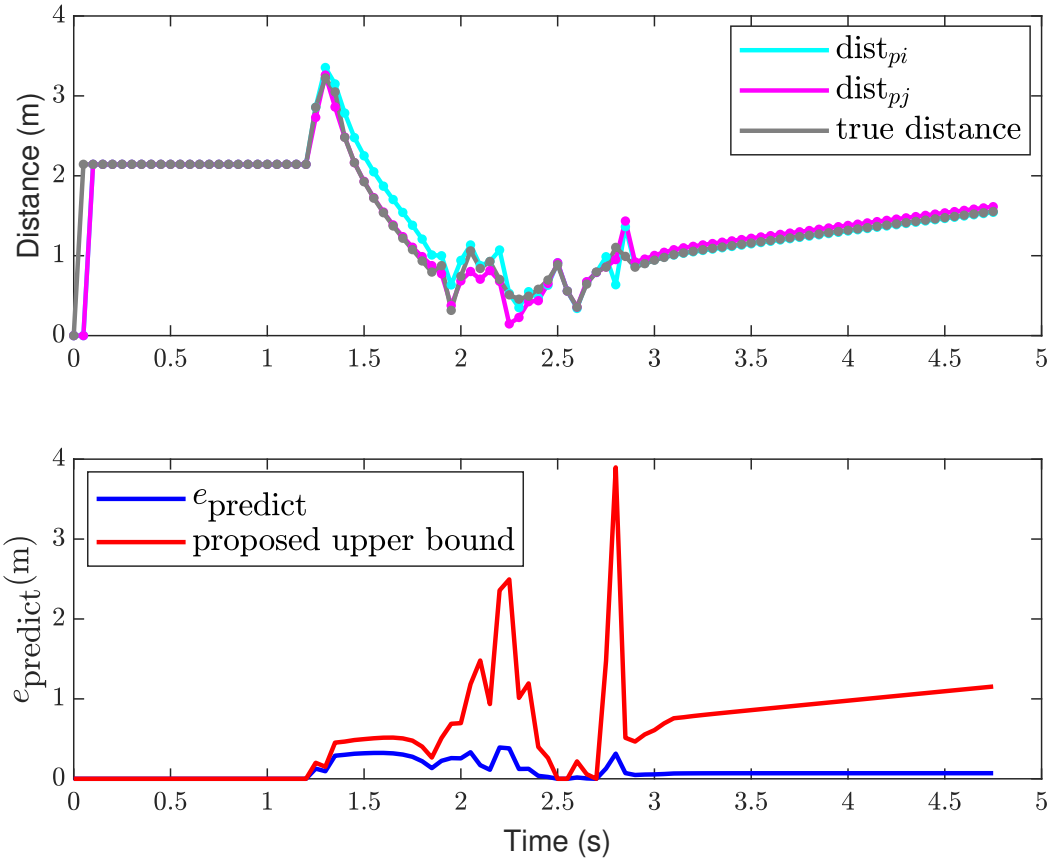


Figure 5.13: **Top** In distributed approach, the distance between robot i and robot j from the view of both robots is not the same. **Bottom** The proposed upper-bound of (5.20) is illustrated in red for the overtaking simulation and the prediction error on distance is bounded by the proposed upper-bound.

Corollary 2.1 *Given the acceptable error values on the states e_x, e_y and e_ψ, let e_x, e_y and e_ψ be defined as (5.21), one can compute $\alpha_{min} \geq 0$, such that the following holds for all $t \geq 0$:*

$$||\mathbf{b}^i(\mathbf{z}^i(t)) - \bar{\mathbf{b}}^i(\bar{\mathbf{z}}^i(t-1))||_2 \leq \alpha_{min}(t). \tag{5.22}$$

Then by enforcing (5.22) as a constraint in the NMPC problem (5.5), if the problem (5.5) is feasible, the prediction error for each state will not violate the acceptable specified error.

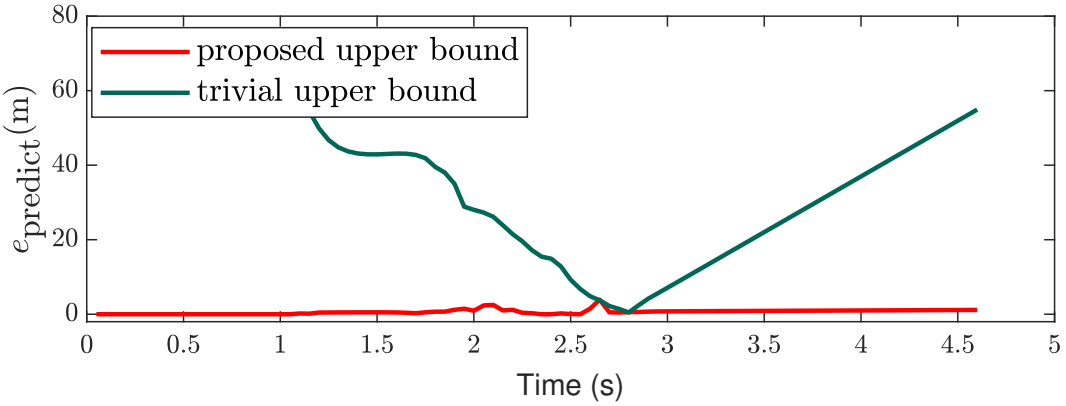


Figure 5.14: The trivial upper-bound is compared versus the proposed upper-bound on distance error.

Proof 2 *The equation (5.20) is quantified based on acceptable error in the following*

$$\begin{aligned} &||\mathbf{b}^i(\mathbf{z}^i(t)) - \bar{\mathbf{b}}^i(\bar{\mathbf{z}}^i(t-1))||_2 \\ =&||\mathbf{A}(\psi(t)) \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} - \mathbf{A}(\bar{\psi}(t-1)) \begin{bmatrix} \bar{x}(t-1) \\ \bar{y}(t-1) \end{bmatrix} ||_2 \\ =&||\mathbf{A}(\bar{\psi}(t-1) \pm e_\psi) \begin{bmatrix} \bar{x}(t-1) \pm e_x \\ \bar{y}(t-1) \pm e_y \end{bmatrix} \\ &- \mathbf{A}(\bar{\psi}(t-1)) \begin{bmatrix} \bar{x}(t-1) \\ \bar{y}(t-1) \end{bmatrix} ||_2 = \alpha(t). \end{aligned} \tag{5.23}$$

The right-hand side of (5.23) is minimized (all the values are known but different combinations of signs must be considered to find the minimum $\alpha(t)$), then the upper-bound is compared with the threshold value $\alpha_{min}(t)$

By specifying the acceptable error and minimizing the right-hand side of (5.23), the value of $\alpha_{\min}(t)$ can be calculated and (5.22) can be imposed as a constraint in the NMPC problem (5.5) to ensure the solution never violates acceptable error values on the prediction error.

5.7 CONCLUSIONS

A distributed algorithm for multi-robot coordination in tight spaces is introduced using nonlinear MPC and strong duality theory. The collision avoidance constraints are formulated in dual formulation and dual decomposition is used to split the large centralized optimization problem into smaller sub-problems. The proposed distributed approach consists of NMPC optimization and collision avoidance optimization and the algorithm iterate between theses two optimizations in an alternating optimization scheme.

The effectiveness of the algorithm is shown for coordination of connected and automated vehicles on public roads through platoon merging. It is shown the distributed approach outperforms the centralized design by more than 2 orders of magnitude. The results show that the distributed algorithm is computationally efficient for online implementation and is scalable to larger networks of robots. In addition, it is shown that the method is generalizable to heterogeneous team of robots with different polytopic shapes. It is shown that compared to a centralized design, the proposed decomposition introduces a local prediction error that could lead to more conservative local trajectories. Nevertheless, it is proved that this error is bounded and can be accounted for in the local NMPC problems.

As part of the future work, testing with real hardware (e.g., drones and ground vehicles) and in more complex scenarios (e.g., urban driving) can be performed. From the algorithm-design perspective, strategies to deal with real sensor data, communication delays, and random faults can be investigated.

Bibliography

[1] A. Alam et al. "Heavy-Duty Vehicle Platooning for Sustainable Freight Transportation: A Cooperative Method to Enhance Safety and Efficiency". In: *IEEE Control Systems Magazine* 35.6 (Dec. 2015), pp. 34–56. ISSN: 1066-033X. DOI: 10.1109/MCS.2015.2471046.

[2] Assad Alam et al. "Guaranteeing safety for heavy duty vehicle platooning: Safe set computations and experimental evaluations". In: *Control Engineering Practice* 24 (2014), pp. 33–41. ISSN: 0967-0661.

[3] Assad Al Alam, Ather Gattami, and Karl Henrik Johansson. "An experimental study on the fuel reduction potential of heavy duty vehicle platooning". In: *13th International IEEE Conference on Intelligent Transportation Systems* (2010), pp. 306–311.

[4] Javier Alonso-Mora, Paul Beardsley, and Roland Siegwart. "Cooperative collision avoidance for nonholonomic robots". In: *IEEE Transactions on Robotics* 34.2 (2018), pp. 404–420.

[5] Yoshikazu Arai et al. "Collision avoidance in multi-robot systems based on multi-layered reinforcement learning". In: *Robotics and Autonomous Systems* 29.1 (1999), pp. 21–32. DOI: https://doi.org/10.1016/S0921-8890(99)00035-4.

[6] T. Balch and R. C. Arkin. "Behavior-based formation control for multirobot teams". In: *IEEE Transactions on Robotics and Automation* 14.6 (Dec. 1998), pp. 926–939. ISSN: 1042-296X.

[7] Francesco Borrelli, Alberto Bemporad, and Manfred Morari. *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[8] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, Mar. 2004. ISBN: 0521833787. URL: http://www.amazon.com/exec/obidos/redirect?tag=citeulike-20%5C&path=ASIN/0521833787.

[9] B. Brito et al. "Model Predictive Contouring Control for Collision Avoidance in Unstructured Dynamic Environments". In: *IEEE Robotics and Automation Letters* (2019), pp. 1–1. DOI: 10.1109/LRA.2019.2929976.

[10] Leonardo Bruni, Alessandro Colombo, and Domitilla Del Vecchio. "Robust multi-agent collision avoidance through scheduling". In: *52nd IEEE Conference on Decision and Control*. IEEE. 2013, pp. 3944–3950.

[11] Wolfram Burgard, Oliver Brock, and Cyrill Stachniss. "Map-Based Precision Vehicle Localization in Urban Environments". In: *Robotics:Science and Systems III*. MIT Press, 2008, pp. 352–. ISBN: 9780262255868.

[12] Lucian Busoniu, Robert Babuška, and Bart De Schutter. "A comprehensive survey of multiagent reinforcement learning". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 38.2 (2008), pp. 156–172.

[13] Michal Čáp et al. "Prioritized planning algorithms for trajectory coordination of multiple mobile robots". In: *IEEE transactions on automation science and engineering* 12.3 (2015), pp. 835–849.

[14] Francois Caron et al. "GPS/IMU data fusion using multisensor Kalman filtering: introduction of contextual aspects". In: *Information Fusion* 7.2 (2006), pp. 221–230. ISSN: 1566-2535.

[15] Tim Caselitz et al. "Monocular camera localization in 3D LiDAR maps". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2016), pp. 1926–1931.

[16] Linying Chen, Hans Hopman, and Rudy R Negenborn. "Distributed model predictive control for vessel train formations of cooperative multi-vessel systems". In: *Transportation Research Part C: Emerging Technologies* 92 (2018), pp. 101–118.

[17] Y. F. Chen et al. "Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. May 2017, pp. 285–292. DOI: 10.1109/ICRA.2017.7989037.

[18] Jiangmin Chunyu et al. "A New Multi-objective Control Design for Autonomous Vehicles". In: vol. 381. inbook, Oct. 2008, pp. 81–102. DOI: 10.1007/978-3-540-88063-9_5.

[19] D. Corona and B. De Schutter. "Adaptive Cruise Control for a SMART Car: A Comparison Benchmark for MPC-PWA Control Methods". In: *IEEE Transactions on Control Systems Technology* 16.2 (Mar. 2008), pp. 365–372. ISSN: 1063-6536.

[20] R. Dang et al. "Coordinated Adaptive Cruise Control System With Lane-Change Assistance". In: *IEEE Transactions on Intelligent Transportation Systems* 16.5 (Oct. 2015), pp. 2373–2383. ISSN: 1524-9050.

[21] Achiya Dax. "The distance between two convex sets". In: *Linear Algebra and its Applications* 416.1 (2006). Special Issue devoted to the Haifa 2005 conference on matrix theory, pp. 184–213. ISSN: 0024-3795. DOI: https://doi.org/10.1016/j.laa.2006.03.022. URL: https://www.sciencedirect.com/science/article/pii/S0024379506001789.

[22] Daniel J Fagnant and Kara Kockelman. "Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations". In: *Transportation Research Part A: Policy and Practice* 77 (2015), pp. 167–181.

[23] Laura Ferranti et al. "Coordination of Multiple Vessels Via Distributed Nonlinear Model Predictive Control". In: *2018 European Control Conference (ECC)*. 2018, pp. 2523–2528.

[24] Paolo Fiorini and Zvi Shiller. "Motion planning in dynamic environments using velocity obstacles". In: *The International Journal of Robotics Research* 17.7 (1998), pp. 760–772.

[25] R. Firoozi et al. "Safe Adaptive Cruise Control with Road Grade Preview and **V2V** Communication". In: *2019 American Control Conference (ACC)*. July 2019, pp. 4448–4453. DOI: 10.23919/ACC.2019.8814928.

[26] R. Firoozi et al. "Vehicle Localization and Control on Roads with Prior Grade Map". In: *2018 IEEE Conference on Decision and Control (CDC)*. Dec. 2018, pp. 6982–6987. DOI: 10.1109/CDC.2018.8619538.

[27] Roya Firoozi, Xiaojing Zhang, and Francesco Borrelli. "Formation and reconfiguration of tight multi-lane platoons". In: *Control Engineering Practice* 108 (2021), p. 104714. ISSN: 0967-0661. DOI: https://doi.org/10.1016/j.conengprac.2020.104714. URL: https://www.sciencedirect.com/science/article/pii/S0967066120302847.

[28] Roya Firoozi et al. "Vehicle localization and control on roads with prior grade map". In: (). arXiv: 1809.04167.

[29] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. "The dynamic window approach to collision avoidance". In: *IEEE Robotics & Automation Magazine* 4.1 (1997), pp. 23–33.

[30] R. Gayle et al. "Multi-robot coordination using generalized social potential fields". In: *2009 IEEE International Conference on Robotics and Automation*. May 2009, pp. 106–113.

[31] Jacopo Guanetti, Yeojun Kim, and Francesco Borrelli. "Control of connected and automated vehicles: State of the art and future challenges". In: *Annual Reviews in Control* 45 (2018), pp. 18–40. ISSN: 1367-5788.

[32] Maxime Guériau et al. "How to assess the benefits of connected vehicles? A simulation framework for the design of cooperative traffic management strategies". In: *Transportation Research Part C Emerging Technologies* 67 (Apr. 2016). DOI: 10.1016/j.trc.2016.01.020.

[33] Hongliang Guo and Yan Meng. "Distributed Reinforcement Learning for Coordinate Multi-Robot Foraging". In: *Journal of Intelligent & Robotic Systems* 60.3 (Dec. 2010), pp. 531–551.

[34] LLC Gurobi Optimization. *Gurobi Optimizer Reference Manual*. 2019. URL: http://www.gurobi.com.

[35] Han-Shue Tan, R. Rajamani, and Wei-Bin Zhang. "Demonstration of an automated highway platoon system". In: *Proceedings of the 1998 American Control Conference. ACC (IEEE Cat. No.98CH36207)*. Vol. 3. June 1998, 1823–1827 vol.3. DOI: 10.1109/ACC.1998.707332.

[36] Y. Hayashi and T. Namerikawa. "Flocking algorithm for multiple nonholonomic cars". In: *2016 55th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*. Sept. 2016, pp. 1660–1665. DOI: 10.1109/SICE.2016.7749193.

[37] J. K. Hedrick et al. "Longitudinal Vehicle Controller Design for IVHS Systems". In: *1991 American Control Conference*. June 1991, pp. 3107–3112. DOI: 10.23919/ACC.1991.4791980.

[38] IHS Markit. *Autonomous Vehicle Sales to Surpass 33 Million Annually in 2040*. 2018. URL: https://tinyurl.com/IHSMarkit2018.

[39] Tamás Keviczky et al. "Decentralized receding horizon control and coordination of autonomous vehicle formations". In: *IEEE Transactions on Control Systems Technology* 16.1 (2007), pp. 19–33.

[40] Jason Kong et al. "Kinematic and dynamic vehicle models for autonomous driving control design". In: *2015 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2015, pp. 1094–1099.

[41] H. M. La, R. Lim, and W. Sheng. "Multirobot Cooperative Learning for Predator Avoidance". In: *IEEE Transactions on Control Systems Technology* 23.1 (Jan. 2015), pp. 52–63. ISSN: 1063-6536. DOI: 10.1109/TCST.2014.2312392.

[42] S. Lefevre, A. Carvalho, and F. Borrelli. "A Learning-Based Framework for Velocity Control in Autonomous Driving". In: *IEEE Transactions on Automation Science and Engineering* 13.1 (Jan. 2016), pp. 32–42. ISSN: 1545-5955.

[43] N. E. Leonard and E. Fiorelli. "Virtual leaders, artificial potentials and coordinated control of groups". In: *Proceedings of the 40th IEEE Conference on Decision and Control (Cat. No.01CH37228)*. Vol. 3. Dec. 2001, 2968–2973 vol.3. DOI: 10.1109/CDC.2001.980728.

[44] S. E. Li et al. "Performance Enhanced Predictive Control for Adaptive Cruise Control System Considering Road Elevation Information". In: *IEEE Transactions on Intelligent Vehicles* 2.3 (Sept. 2017), pp. 150–160. ISSN: 2379-8904.

[45] J. Löfberg. "YALMIP : A Toolbox for Modeling and Optimization in MATLAB". In: *Proceedings of the CACSD Conference*. Taipei, Taiwan, 2004.

[46] Johan Löfberg. "YALMIP: A toolbox for modeling and optimization in MATLAB". In: *Proceedings of the CACSD Conference*. Vol. 3. Taipei, Taiwan. 2004.

[47] A. Loria, J. Dasdemir, and N. Alvarez Jarquin. "Leader-Follower Formation and Tracking Control of Mobile Robots Along Straight Paths". In: *IEEE Transactions on Control Systems Technology* 24.2 (Mar. 2016), pp. 727–732. ISSN: 1063-6536. DOI: 10.1109/TCST.2015.2437328.

[48] Li-hua Luo et al. "Model predictive control for adaptive cruise control with multi-objectives: comfort, fuel-economy, safety and car-following". In: *Journal of Zhejiang University SCIENCE A* 11.3 (Mar. 2010), pp. 191–201. ISSN: 1862-1775.

[49] Javad Marzbanrad and Iman Tahbaz-zadeh Moghaddam. "Self-tuning control algorithm design for vehicle adaptive cruise control system through real-time estimation of vehicle parameters and road grade". In: *Vehicle System Dynamics* 54.9 (2016), pp. 1291–1316.

[50] MI News Network. *7 Major Developments in Autonomous Shipping in 2018*. 2018. URL: https://www.marineinsight.com/know-more/7-major-developments-in-autonomous-shipping-in-2018/.

[51] Isaac Miller, Mark Campbell, and Dan Huttenlocher. "Map-aided localization in sparse global positioning system environments using vision and particle filtering". In: *Journal of Field Robotics* 28.5 (2011), pp. 619–643. ISSN: 1556-4967. DOI: 10.1002/rob.20395.

[52] R. Olfati-Saber. "Flocking for multi-agent dynamic systems: algorithms and theory". In: *IEEE Transactions on Automatic Control* 51.3 (Mar. 2006), pp. 401–420. ISSN: 0018-9286. DOI: 10.1109/TAC.2005.864190.

[53] Xiangjun Qian, Arnaud de La Fortelle, and Fabien Moutarde. "A Hierarchical Model Predictive Control Framework for On-road Formation Control of Autonomous Vehicles". In: *10.1109/IVS.2016.7535413*. June 2016.

[54] Rajesh Rajamani et al. "Design and Experimental Implementation of Longitudinal Control for a Platoon of Automated Vehicles". In: *Rajmanian*. 2000.

[55] Felix Rey et al. "Fully decentralized ADMM for coordination and collision avoidance". In: *2018 European Control Conference (ECC)*. 2018, pp. 825–830.

[56] Craig W. Reynolds. "Flocks, Herds and Schools: A Distributed Behavioral Model". In: *SIGGRAPH Comput. Graph.* 21.4 (Aug. 1987), pp. 25–34. ISSN: 0097-8930. DOI: 10.1145/37402.37406.

[57] Jackeline Rios-Torres and Andreas A. Malikopoulos. "A Survey on the Coordination of Connected and Automated Vehicles at Intersections and Merging at Highway On-Ramps". In: *IEEE Transactions on Intelligent Transportation Systems* 18.5 (2017), pp. 1066–1077. ISSN: 15249050.

[58] Ugo Rosolia, Stijn De Bruyne, and Andrew G Alleyne. "Autonomous vehicle control: A nonconvex approach for obstacle avoidance". In: *IEEE Transactions on Control Systems Technology* 25.2 (2016), pp. 469–484.

[59] L. R. Sahawneh et al. "Real-Time Implementation of GPS Aided Low-Cost Strapdown Inertial Navigation System". In: *Journal of Intelligent & Robotic Systems* 61.1 (Jan. 2011), pp. 527–544. ISSN: 1573-0409. DOI: 10.1007/s10846-010-9501-0.

[60] Roman Schmied, Harald Waschl, and Luigi del Re. "Comfort Oriented Robust Adaptive Cruise Control in Multi-Lane Traffic Conditions". In: 49 (Dec. 2016), pp. 196–201.

[61] F. E. Schneider and D. Wildermuth. "A potential field based approach to multi robot formation navigation". In: *IEEE International Conference on Robotics, Intelligent Systems and Signal Processing*. Vol. 1. Oct. 2003, pp. 680–685.

[62] Wilko Schwarting et al. "Safe nonlinear trajectory generation for parallel autonomy with a dynamic vehicle model". In: *IEEE Transactions on Intelligent Transportation Systems* 19.9 (2017), pp. 2994–3008.

[63] P. Shakouri et al. "Adaptive Cruise Control System: Comparing Gain-Scheduling PI and LQ Controllers". In: *IFAC Proceedings Volumes* 44.1 (2011). 18th IFAC World Congress, pp. 12964–12969. ISSN: 1474-6670.

[64] S. E. Shladover et al. "Automated vehicle control developments in the PATH program". In: *IEEE Transactions on Vehicular Technology* 40.1 (Feb. 1991), pp. 114–130. ISSN: 1939-9359. DOI: 10.1109/25.69979.

[65] Matt Simon. *Inside the Amazon Warehouse Where Humans and Machines Become One*. 2019. URL: https://www.wired.com/story/amazon-warehouse-robots/.

[66] Xiaotong Sun and Yafeng Yin. "Behaviorally stable vehicle platooning for energy savings". In: *Transportation Research Part C: Emerging Technologies* 99 (2019), pp. 37–52. ISSN: 0968-090X. DOI: https://doi.org/10.1016/j.trc.2018.12.017. URL: http://www.sciencedirect.com/science/article/pii/S0968090X18312245.

[67] Olle Sundstr and Lino Guzzella. "A generic dynamic programming Matlab function". In: *2009 IEEE Control Applications, (CCA) & Intelligent Control, (ISIC)* (2009), pp. 1625–1630.

[68] H. G. Tanner and A. Kumar. "Towards Decentralization of Multi-robot Navigation Functions". In: *Proc. of the 2005 IEEE International Conference on Robotics and Automation*. Apr. 2005, pp. 4132–4137.

[69] V. Turri et al. "A model predictive controller for non-cooperative eco-platooning". In: *2017 American Control Conference (ACC)*. May 2017, pp. 2309–2314.

[70] Valerio Turri, Bart Besselink, and Karl H Johansson. "Cooperative look-ahead control for fuel-efficient and safe heavy-duty vehicle platooning". In: *IEEE Transactions on Control Systems Technology* 25.1 (2017), pp. 12–28.

[71] Jur Van Den Berg et al. "Reciprocal n-body collision avoidance". In: *Robotics research*. Springer, 2011, pp. 3–19.

[72] R. Van Parys and G. Pipeleers. "Distributed model predictive formation control with inter-vehicle collision avoidance". In: *2017 11th Asian Control Conference (ASCC)*. Dec. 2017, pp. 2399–2404. DOI: 10.1109/ASCC.2017.8287550.

[73] Andreas Wächter and Lorenz T Biegler. "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming". In: *Mathematical programming* 106.1 (2006), pp. 25–57.

[74] Jon Walker. *The Self-Driving Car Timeline - Predictions from the Top 11 Global Automakers*. 2019. URL: https://emerj.com/ai-adoption-timelines/self-driving-car-timeline-themselves-top-11-automakers/.

[75] Liang Wang, Yihuan Zhang, and Jun Wang. "Map-Based Localization Method for Autonomous Vehicles Using 3D-LIDAR." In: *IFAC-PapersOnLine* 50.1 (2017). 20th IFAC World Congress, pp. 276–281. ISSN: 2405-8963.

[76] Yinsong Wang et al. "Google Earth elevation data extraction and accuracy assessment for transportation applications". In: *PLOS ONE* 12.4 (Apr. 2017), pp. 1–17. DOI: 10.1371/journal.pone.0175756.

[77] Alexander Liniger Xiaojing Zhang and Francesco Borrelli. "Optimization-based collision avoidance". In: *arXiv* (2017). arXiv: 1711.03449.

[78] Zhi Yan, Nicolas Jouandeau, and Arab Ali Cherif. "A survey and analysis of multi-robot coordination". In: *International Journal of Advanced Robotic Systems* 10.12 (2013), p. 399.

[79] X. Zhang et al. "Autonomous Parking Using Optimization-Based Collision Avoidance". In: *IEEE Conference on Decision and Control*. Dec. 2018, pp. 4327–4332. DOI: 10.1109/CDC.2018.8619433.

[80] Xiaojing Zhang, Alexander Liniger, and Francesco Borrelli. "Optimization-based collision avoidance". In: *arXiv preprint arXiv:1711.03449* (2017).

[81] Gao Zhenhai et al. "Multi-argument Control Mode Switching Strategy for Adaptive Cruise Control System". In: *Procedia Engineering* 137 (2016). Green Intelligent Transportation System and Safety, pp. 581–589. ISSN: 1877-7058.

Appendix A

Proof of Theorem I

Proof 3 *Proof of Theorem 1*
For robot i and robot j, since $\mathbf{s}_{ij}$ represents the consensus variable between the two robots (normal of a separating hyperplane between the robots), therefore $\mathbf{s}_{ij} = \mathbf{s}_{ji}$, in CA optimization (5.6). Also, based on (5.6a) and (5.6b) constraints in CA optimization (5.6), we have

$$\boldsymbol{\lambda}_{ij} = -(\bar{\mathbf{A}}^i(\bar{\mathbf{z}}^i)^\top)^\dagger \mathbf{s}_{ij}, \tag{A.1}$$

$$\boldsymbol{\lambda}_{ji} = (\bar{\mathbf{A}}^j(\bar{\mathbf{z}}^j)^\top)^\dagger \mathbf{s}_{ij}, \tag{A.2}$$

where $\mathbf{A}^\dagger$ is $(\mathbf{A}^\top\mathbf{A})^{-1}\mathbf{A}^\top$.

The minimum distance $dist_{pi}$ from the view of robot i, is computed by substituting (A.1) and (A.2) into (5.5a)

$$dist_{pi} = \left(\mathbf{b}^i(\mathbf{z}^i(t))^\top(\bar{\mathbf{A}}^i(\bar{\mathbf{z}}^i(t-1))^\top)^\dagger - \bar{\mathbf{b}}^j(\bar{\mathbf{z}}^j(t-1))^\top(\bar{\mathbf{A}}^j(\bar{\mathbf{z}}^j(t-1))^\top)^\dagger\right)\bar{\mathbf{s}}_{ij}. \tag{A.3}$$

Similarly, from the view of robot j, by substituting (A.1) and (A.2) into (5.5a) we have

$$dist_{pj} = \left(\bar{\mathbf{b}}^i(\bar{\mathbf{z}}^i(t-1))^\top(\bar{\mathbf{A}}^i(\bar{\mathbf{z}}^i(t-1))^\top)^\dagger - \mathbf{b}^j(\mathbf{z}^j(t))^\top(\bar{\mathbf{A}}^j(\bar{\mathbf{z}}^j(t-1))^\top)^\dagger\right)\bar{\mathbf{s}}_{ij}. \tag{A.4}$$

The difference of (A.3) and (A.4)) is the prediction error $e_{predict}$,

$$\begin{aligned} e_{predict}(t) = ||\Big(&\mathbf{b}^i(\mathbf{z}^i(t))^\top(\bar{\mathbf{A}}^i(\bar{\mathbf{z}}^i(t-1))^\top)^\dagger \\ &- \bar{\mathbf{b}}^j(\bar{\mathbf{z}}^j(t-1))^\top(\bar{\mathbf{A}}^j(\bar{\mathbf{z}}^j(t-1))^\top)^\dagger \\ &- \bar{\mathbf{b}}^i(\bar{\mathbf{z}}^i(t-1))^\top(\bar{\mathbf{A}}^i(\bar{\mathbf{z}}^i(t-1))^\top)^\dagger \\ &+ \mathbf{b}^j(\mathbf{z}^j(t))^\top(\bar{\mathbf{A}}^j(\bar{\mathbf{z}}^j(t-1))^\top)^\dagger\Big)\bar{\mathbf{s}}_{ij}||_2. \end{aligned} \tag{A.5}$$

On the other hand,

$$||(\mathbf{A}^\top)^\dagger||_\mathbf{F} = \sqrt{\sum_{i=1}^{r} \sigma_i^2},$$

where r is the matrix rank. For the application of vehicles (which all the polytopes are rectangle or square) The matrix $\mathbf{A}(\mathbf{z})$ is defined as

$$\mathbf{A}(\mathbf{z}) = \begin{bmatrix} \mathbf{R}(\mathbf{z})^\top \\ -\mathbf{R}(\mathbf{z})^\top \end{bmatrix},$$

its norm remains constant as $\mathbf{z}$ changes. Performing Singular Value Decomposition (SVD) on the rotation matrix $\mathbf{R}(\mathbf{z})$, returns identity matrix and singular values are independent from $\mathbf{z}$ values. With the same analogy, SVD on $(\mathbf{A}(\mathbf{z})^\top)^\dagger$ always results in the same decomposition with no regards to $\mathbf{z}$ values, $(\mathbf{A}^\top)^\dagger = U\Sigma V^\top$, where the matrix Σ is

$$\Sigma = \begin{bmatrix} \frac{\sqrt{2}}{2} & 0 \\ 0 & \frac{\sqrt{2}}{2} \end{bmatrix}. \tag{A.6}$$

Therefore,

$$||(\mathbf{A}(\mathbf{z})^\top)^\dagger||_\mathbf{F} = \sqrt{\sum_{i=1}^{r} \sigma_i^2} = \sqrt{\left(\frac{\sqrt{2}}{2}\right)^2 + \left(\frac{\sqrt{2}}{2}\right)^2} = 1. \tag{A.7}$$

By rearranging the terms, using triangular inequality, Cauchy-Schwarz inequality and based on the constraint $||\mathbf{s}_{ij}||_2 \leq 1$ and $||(\mathbf{A}(\mathbf{z})^\top)^\dagger||_\mathbf{F} = 1$ obtained from (A.7) we have

$$\begin{aligned}
&e_{predict} \\
&= ||\Big(\big(\mathbf{b}^i(\mathbf{z}^i(t)) - \bar{\mathbf{b}}^i(\bar{\mathbf{z}}^i(t-1))\big)^\top (\bar{\mathbf{A}}^i(\bar{\mathbf{z}}^i(t-1))^\top)^\dagger \\
&+ \big(\mathbf{b}^j(\mathbf{z}^j(t)) - \bar{\mathbf{b}}^j(\bar{\mathbf{z}}^j(t-1))\big)^\top (\bar{\mathbf{A}}^j(\bar{\mathbf{z}}^j(t-1))^\top)^\dagger \Big) \bar{\mathbf{s}}_{ij}||_2 \\
&\leq ||\Big(\big(\mathbf{b}^i(\mathbf{z}^i(t)) - \bar{\mathbf{b}}^i(\bar{\mathbf{z}}^i(t-1))\big)^\top (\bar{\mathbf{A}}^i(\bar{\mathbf{z}}^i(t-1))^\top)^\dagger \\
&+ \big(\mathbf{b}^j(\mathbf{z}^j(t)) - \bar{\mathbf{b}}^j(\bar{\mathbf{z}}^j(t-1))\big)^\top \\
&(\bar{\mathbf{A}}^j(\bar{\mathbf{z}}^j(t-1))^\top)^\dagger \Big) ||_2 \; ||\bar{\mathbf{s}}_{ij}||_2 \\
&\leq ||\mathbf{b}^i(\mathbf{z}^i(t)) - \bar{\mathbf{b}}^i(\bar{\mathbf{z}}^i(t-1))||_2 \; ||(\bar{\mathbf{A}}^i(\bar{\mathbf{z}}^i(t-1))^\top)^\dagger||_\mathbf{F} \\
&+ ||\mathbf{b}^j(\mathbf{z}^j(t)) - \bar{\mathbf{b}}^j(\bar{\mathbf{z}}^j(t-1))||_2 \; ||(\bar{\mathbf{A}}^j(\bar{\mathbf{z}}^j(t-1))^\top)^\dagger||_\mathbf{F} \\
&\leq ||\mathbf{b}^i(\mathbf{z}^i(t)) - \bar{\mathbf{b}}^i(\bar{\mathbf{z}}^i(t-1))||_2 \\
&+ ||\mathbf{b}^j(\mathbf{z}^j(t)) - \bar{\mathbf{b}}^j(\bar{\mathbf{z}}^j(t-1))||_2
\end{aligned} \tag{A.8}$$

For the cases with general polytopic shapes we have

$$\begin{aligned} ||(\bar{\mathbf{A}}^i(\bar{\mathbf{z}}^i(t-1))^\top)^\dagger||_{\mathbf{F}} &= ||\mathbf{A}_O^i \ \mathbf{R}(\mathbf{z}^i(t-1))^\top||_{\mathbf{F}} \\ &\leq \|\mathbf{A}_O^i\|_{\mathbf{F}} \ \|\mathbf{R}(\mathbf{z}^i(t-1))^\top\|_{\mathbf{F}}, \end{aligned} \tag{A.9}$$

where $\mathbf{A}_O^i$ *is the polytopic representation at the origin. The SVD decomposition for rotation matrix is independent of* $\mathbf{z}$*, and* $||\mathbf{R}||_{\mathbf{F}} = \sqrt{2}$*. On the other hand, the constant matrix* $\mathbf{A}_O^i$ *is not a function of time and*

$$\|(\mathbf{A}_O{}^{i\top})^\dagger\|_{\mathbf{F}} = \sqrt{\sum_{i=1}^{r_1} \sigma_i^2}.$$

so

$$||(\bar{\mathbf{A}}^i(\bar{\mathbf{z}}^i(t-1))^\top)^\dagger||_{\mathbf{F}} \leq \sqrt{2\sum_{i=1}^{r_1} \sigma_i^2}. \tag{A.10}$$

With the same analogy

$$||(\bar{\mathbf{A}}^j(\bar{\mathbf{z}}^j(t-1))^\top)^\dagger||_{\mathbf{F}} \leq \sqrt{2\sum_{i=1}^{r_2} \sigma_i^2}. \tag{A.11}$$

By setting

$$c_i = \sqrt{2\sum_{i=1}^{r_1} \sigma_i^2}$$

and

$$c_j = \sqrt{2\sum_{i=1}^{r_2} \sigma_i^2}$$

and substituting in (A.8)*, we have*

$$\begin{aligned} &e_{predict} \\ &\leq c_i||\mathbf{b}^i(\mathbf{z}^i(t)) - \bar{\mathbf{b}}^i(\bar{\mathbf{z}}^i(t-1))||_2 \\ &+ c_j||\mathbf{b}^j(\mathbf{z}^j(t)) - \bar{\mathbf{b}}^j(\bar{\mathbf{z}}^j(t-1))||_2. \end{aligned} \tag{A.12}$$

This is the formulation for general polytopic shapes, for special case of rectangular shapes like vehicles according to (A.7)*,* $c_i = c_j = 1$*.*

Appendix B

Derivation of Dual Formulation from Primal

To derive the dual formulation(5.3) from primal (4.12), the Lagrangian is formed. The equivalent form of (4.12) is

$$\begin{aligned} &\min_{\mathbf{x},\mathbf{y},\mathbf{w}} \|\mathbf{w}\|_2 \\ &\text{s.t. } \mathbf{A}_1\mathbf{x} \preceq \mathbf{b}_1, \quad \mathbf{A}_2\mathbf{y} \preceq \mathbf{b}_2, \quad \mathbf{x}-\mathbf{y}=\mathbf{w} \end{aligned} \tag{B.1}$$

By forming the Lagrangian dual of (B.1) we have

$$\begin{aligned} g(\boldsymbol{\lambda}_{12},\boldsymbol{\lambda}_{21},\mathbf{s}) = \inf_{\mathbf{x},\mathbf{y},\mathbf{w}} \big(&\|\mathbf{w}\|_2 + \boldsymbol{\lambda}_{12}^\top(\mathbf{A}_1\mathbf{x}-\mathbf{b}_1)+ \\ &\boldsymbol{\lambda}_{21}^\top(\mathbf{A}_2\mathbf{y}-\mathbf{b}_2) + \mathbf{s}^\top(\mathbf{x}-\mathbf{y}-\mathbf{w})\big) \\ &\text{s.t. } \boldsymbol{\lambda}_{12},\boldsymbol{\lambda}_{21} \succeq 0. \end{aligned} \tag{B.2}$$

By rearranging the terms in (B.2), we have

$$\begin{aligned} g(\boldsymbol{\lambda}_{12},\boldsymbol{\lambda}_{21},\mathbf{s}) = &\inf_{\mathbf{w}}(\|\mathbf{w}\|_2 - \mathbf{s}^\top\mathbf{w}) \\ &+\inf_{\mathbf{x}}\big((\mathbf{A}_1^\top\boldsymbol{\lambda}_{12}+\mathbf{s})^\top\mathbf{x} - \boldsymbol{\lambda}_{12}^\top\mathbf{b}_1\big) \\ &+\inf_{\mathbf{y}}\big((\mathbf{A}_2^\top\boldsymbol{\lambda}_{21}-\mathbf{s})^\top\mathbf{y} - \boldsymbol{\lambda}_{21}^\top\mathbf{b}_2\big) \\ &\text{s.t. } \boldsymbol{\lambda}_{12},\boldsymbol{\lambda}_{21} \succeq 0. \end{aligned} \tag{B.3}$$

We can reformulate (B.3) using the definition of conjugate function. The conjugate function is defined as $f^*(\mathbf{s}) = \sup_{\mathbf{x}\in\mathrm{dom} f}(\mathbf{s}^\top\mathbf{x} - f(\mathbf{x}))$. Also we use the fact the $\inf f(\mathbf{x}) = -\sup(-f(\mathbf{x}))$. So we have

$$\inf_{\mathbf{x}}(f(\mathbf{x}) - \mathbf{s}^\top\mathbf{x}) = -\sup_{\mathbf{x}}(-f(\mathbf{x}) + \mathbf{s}^\top\mathbf{x}) \quad = -f^*(\mathbf{x}). \tag{B.4}$$

Therefore the first term of right-hand side of (B.3) can be written as

$$\inf_{\mathbf{w}}(\|\mathbf{w}\|_2 - \mathbf{s}^\top\mathbf{w}) = -f^*(\mathbf{s}) = -\|\mathbf{w}\|^*. \tag{B.5}$$

The conjugate of $\|\mathbf{w}\|$ is

$$f^*(\mathbf{s}) = \begin{cases} 0 & \|\mathbf{s}\|_* \leq 1 \\ \infty & \text{otherwise} \end{cases} \tag{B.6}$$

The derivation of (B.6) (derivation for conjugate of $\|\cdot\|$) is proven at [8]. The second term in the right-hand side of (B.3) is

$$\begin{aligned} &\inf_{\mathbf{x}} \left((\mathbf{A}_1^\top \boldsymbol{\lambda}_{12} + \mathbf{s})^\top \mathbf{x} - \boldsymbol{\lambda}_{12}^\top \mathbf{b}_1 \right) \\ &= \begin{cases} -\mathbf{b}_1^\top \boldsymbol{\lambda}_{12} & \mathbf{A}_1^\top \boldsymbol{\lambda}_{12} + \mathbf{s} = 0 \\ -\infty & \text{otherwise} \end{cases} \end{aligned} \tag{B.7}$$

and similarly the third term is

$$\begin{aligned} &\inf_{\mathbf{y}} \left((\mathbf{A}_2^\top \boldsymbol{\lambda}_{21} - \mathbf{s})^\top \mathbf{y} - \boldsymbol{\lambda}_{21}^\top \mathbf{b}_2 \right) \\ &= \begin{cases} -\mathbf{b}_2^\top \boldsymbol{\lambda}_{21} & \mathbf{A}_2^\top \boldsymbol{\lambda}_{21} - \mathbf{s} = 0 \\ -\infty & \text{otherwise} \end{cases} \end{aligned} \tag{B.8}$$

By substituting (B.5), (B.7) and (B.8), in (B.3), the formulation (5.3) is obtained.

Appendix C

KKT Conditions for Geometric Interpretations

The dual variables have interesting geometric interpretation. All these geometric meanings are obtained from the Karush-Kuhn-Tucker (KKT) conditions for problem (5.2). The dual problem expressed in (5.3)

$$\mathbf{A}_1 = \begin{bmatrix} —\mathbf{a_1}_1— \\ —\mathbf{a_1}_2— \\ \vdots \\ —\mathbf{a_1}_{m_1}— \end{bmatrix}, \quad \mathbf{A}_2 = \begin{bmatrix} —\mathbf{a_2}_1— \\ —\mathbf{a_2}_2— \\ \vdots \\ —\mathbf{a_2}_{m_2}— \end{bmatrix},$$
$$\mathbf{b}_1^\top = \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1m_1} \end{bmatrix},$$
$$\mathbf{b}_2^\top = \begin{bmatrix} b_{21} & b_{22} & \dots & b_{2m_2} \end{bmatrix},$$
$$\boldsymbol{\lambda}_{12}^\top = \begin{bmatrix} \lambda_{121} & \lambda_{122} & \dots & \lambda_{12m_1} \end{bmatrix},$$
$$\boldsymbol{\lambda}_{21}^\top = \begin{bmatrix} \lambda_{211} & \lambda_{212} & \dots & \lambda_{21m_2} \end{bmatrix}.$$

Since the primal primal problem is a convex function, the KKT condition is necessary and sufficient for the points to be primal and dual optimal. Other than feasibility of primal and dual problem, there is complementary slackness

$$\lambda_{12i}(\mathbf{a_1}_i\mathbf{x} - b_{1i}) = 0, \quad i \in \{1, 2, \dots, m_1\}, \tag{C.1}$$
$$\lambda_{21i}(\mathbf{a_2}_i\mathbf{y} - b_{2i}) = 0, \quad i \in \{1, 2, \dots, m_2\}, \tag{C.2}$$

By expanding (C.1) and (C.2) we will have

$$\mathbf{x}^\top \mathbf{a_1}_i^\top \lambda_{12i} = b_{1i}\lambda_{12i}, \ i \in \{1, 2, \dots, m_1\}, \tag{C.3}$$
$$\mathbf{y}^\top \mathbf{a_2}_i^\top \lambda_{21i} = b_{2i}\lambda_{21i}, \ i \in \{1, 2, \dots, m_2\} \tag{C.4}$$

where $\mathbf{a_1}_i$ and $\mathbf{a_2}_i$ are row vectors of $\mathbf{A}_1$ and $\mathbf{A}_2$, respectively. The other KKT condition is stationarity condition which is base on the primal problem in (B.1)

$$\mathbf{A}_1^\top \boldsymbol{\lambda}_{12} + \mathbf{s} = 0, \tag{C.5}$$

$$\mathbf{A}_2^\top \boldsymbol{\lambda}_{21} - \mathbf{s} = 0, \tag{C.6}$$

$$\frac{\mathbf{w}}{\|\mathbf{w}\|} - \mathbf{s} = 0. \tag{C.7}$$

As seen, (C.5) and (C.6) reflect the equality constrains of the dual problem (5.3). Combining all together, at optimum we have

$$\mathbf{s}^\top \mathbf{x} = -\mathbf{b}_1^\top \boldsymbol{\lambda}_{12} > 0$$
$$\mathbf{s}^\top \mathbf{y} = +\mathbf{b}_2^\top \boldsymbol{\lambda}_{21} < 0$$