Distributed Model Predictive Control for Multi-Vehicle Autonomous Driving

Cooperative vs. Non-cooperative Control

Rens Vermeer

Master of Science Thesis

Delft University of Technology

Delft Center for Systems and Control

Distributed Model Predictive Control for Multi-Vehicle Autonomous Driving

Cooperative vs. Non-cooperative Control

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

Rens Vermeer

August 6, 2020

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Abstract

In this thesis we consider the problem of controlling multiple autonomous vehicles in a highway scenario, via Model Predictive Control (MPC). By iteratively solving a motion planning Optimal Control Problem (OCP), MPC is perfectly suited for unknown dynamic environments, while optimally computing path and vehicle inputs. Moreover, MPC can ensure the satisfaction of collision avoidance constraints, a prerequisite for safe automated driving.

The collision avoidance constraints render the OCP non-convex. This thesis tackles this non-convexity by either designing nonlinear MPC controllers, or by convexifying these non-convex constraints.

Moreover, control of a large, networked system of automated vehicles is achieved by designing local, subsystem-based controllers. We analyse three different algorithms to distribute the plantwide OCP. All controllers are subjected to an objective analysis and compared to see which is the most efficient and most practical to implement. Centralized MPC is used as benchmark, since this gives the plantwide optimal solution. The first decomposed algorithm is decentralized MPC, where subsystems communicate a single time every MPC iteration and compute their new trajectory based on the previously communicated trajectory of neighboring subsystems. The second method is based on sub-optimal cooperative distributed MPC. Here, vehicles perform multiple sub-optimal iterations of a Gauss-Jacobi type distributed optimization. For the last method, based on a Generalized Potential Game, the vehicles sequentially solve and communicate the solution of their local OCP in order to find an ϵ-Nash Equilibrium. By relying on additional constraints or fixed ordering among vehicles, all three controllers are able to recursively feasible compute their own trajectory while avoiding other vehicles.

The distributed controllers are assessed in two different scenarios, using three different criteria, i.e., the overall effectiveness of the controller, the local effectiveness of the controller and the progress made by each vehicle in the simulation. The first criteria gives an indication of the level of cooperation among vehicles, the second shows the individual satisfaction of each vehicle with respect to its reference, and the last represents the overall progress each vehicle has made in the highway simulation.

Contents

List of Figures

Preface

Personally, in my background as test-engineer, performing test's and validating Advanced Driving Assistence Systems (ADAS), the subject of automated driving has always been a stimulating subject. It has been a wish to graduate my master's thesis in this field of research. I am very grateful to have Prof. Dr. ir. S. Grammatico as my assigned supervisor. After a short discussion, the idea of the thesis subject "Distributed Model Predictive Control for Autonomous Vehicles in a Highway Scenario" was born. I would also like to thank my daily supervisor M. Bianchi for his continuous support and guidance during the process.

The project is supervised by:
ir. M. Bianchi
prof. dr. ir. S. Grammatico

Delft, University of Technology — Rens Vermeer
August 6, 2020

Chapter 1

Introduction

Nowadays, smart mobility is a prerequisite for a sound economy and society. However, current mobility solutions are not sustainable due to accidents, pollution and congestion. Automated control of vehicles on highways and (sub)urban areas can significantly contribute to speed up the transition towards a sustainable mobility system, [15].

1-1 Automated Driving

As technology advances, human interaction with machines decreases. Processes were human interaction was necessary before, are becoming automated, and eventually autonomous. The first mobile robot was developed in 1966 [25]. The first autonomous cars have been around since the 1980's [48]. These required specially marked streets. Since then, automation of vehicles has progressed significantly. Instead of using specially marked streets, the focus has been on driving autonomous, in unknown, unmodified environments and in cooperation with other automated vehicles. In fact, autonomous driving is becoming a promising technology to enhance traffic efficiency and reduce the amount of resources wasted on accidents and congestions. It is predicted that, by 2030, more than 50% of vehicles on the road will be automated [38]. In the future most vehicles will be networked with other vehicles and possibly with some road infrastructural units. This represents a huge system, sharing information in real time both locally (short distance, e.g., for safety) as well as globally (with servers, e.g., for efficient navigation) [38].

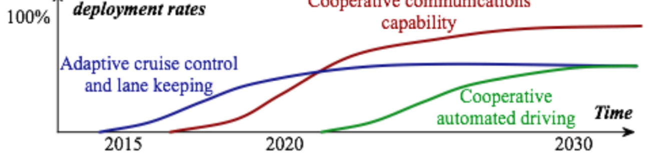


Figure 1-1: Vision of autonet2030 on automated driving, [38]

1-2 Research Objective

While automated driving can reduce traffic accidents and increase traffic efficiency, designing algorithms for safe and efficient automated driving systems is not a simple task.

Many different motion planning algorithms have been proposed over the years with different features, capabilities and mathematical frameworks, e.g., grid-based search [22], interval-based search [23], artificial potential fields [49] or sampling-based algorithms [28], just to name a few. In the case of automated driving vehicles, vehicles drive in unknown dynamic environments. Hence, the workspace is not known a-priori, this renders most of above mentioned algorithms infeasible. A motion planning algorithm based on MPC on the other hand, is perfectly suited for such an application. By iteratively solving a motion planning Optimal Control Problem (OCP), MPC is able to adapt to dynamic environments, while optimally (w.r.t to its objective function) compute a trajectory.

Although MPC is perfectly suited for such an algorithm, it still faces some issues that need to be tackled for safe autonomous driving, e.g., collision avoidance modelling. To ensure vehicles are able to avoid each other, constraints are imposed in the optimization. This is further explained in sections 3-2 and 3-3.

In this thesis, we consider the problem of controlling networks of automated vehicles, moving in a highway environment. Specifically, we focus on the path planning problem, where trajectories for all the vehicles have to be designed online to achieve efficient operation, while avoiding collisions between vehicles and infrastructure.

In fact, MPC is perfectly suited for this purpose, thanks to its ability to ensure optimality while guaranteeing constraint satisfaction. Nonetheless, some issues have to be tackled. The first problem we consider is the modelling of collision avoidance constraints. More precisely, either these non-convex constraints need to be convexified in order to be used in a standard MPC Quadratic Programming (QP) problem, or, a nonlinear MPC scheme has to be formulated to take the non-convex constraints into account.

Furthermore, solving the plantwide OCP in a centralized fashion is not organizationally practical, since online computation time rises with increasing number of subsystems, and all subsystems rely upon the same central agent, making it sensitive to hardware failure. Therefore, we consider the problem of distributing the problem among the different vehicles. To ensure feasibility of the distributed plantwide OCP, local subsystem will have to cooperate and communicate. The level of cooperation and the corresponding communication scheme depends on the type of distributed algorithm.

Therefore, our objective here is to compare and assess different distributed or decentralized MPC schemes. Our analysis focuses on three different types of criteria, i.e., the overall effectiveness of the controller, the local effectiveness of the controller and the progress made by each vehicle in the simulation.

1-3 Thesis Organization

The thesis is organized as follows. In chapter 2, some mathematical background is given regarding networked systems and game theory. Chapter 3 describes two methods to define collision avoidance constraints and introduces the different distributed control methods. Which are explained in detail in chapter 4. These controllers are assessed using the criteria explained in chapter 5. The corresponding results are shown and discussed in chapters 6 and 7. Conclusions on the research and future research topics are proposed in chapter 8.

1-4 Summary of Literature Review

At the start of this master thesis study, an extensive review is done in the literature study. As short summary in the different topics of the literature study is given here.

Collision Avoidance Modelling

Vehicle Collision Avoidance Modelling is an interesting subject which is widely studied. This is mainly due to the fact the non-convex collision avoidance constraints, need to be convexified in order to be used in linear or quadratic solvers, which in turn, are often preferred due to their widely studied stability and robustness properties. Among other things, the literature study recaps several methods to handle non-convex constraints, i.e.:

- Sequential Convex Programming (SCP), which approximates the non-convex optimization problem using a sequence of convex OCP's [6, 3]
- Semi-Definite Programming Relaxations (SDPR), which can be considered as an extension of linear programming where linear inequalities are replaced by matrix inequalities [7, 1, 36].
- Mixed Integer Programming (MIP), by relying on binary decision variables, MIP is able to convexify, non-convex constraints [16, 37, 39, 14, 1, 41]
- Non-Linear Programming (NLP), is able to directly implement non-linear and/or non-convex constraints

Distributed Model Predictive Control

Another topic discussed in the literature study is Distributed Model Predictive Control (D-MPC). Decomposing a large networked system can be done using several different techniques. In general, a division between three different types of Distributed MPC (D-MPC) can be made, namely:

- Synchronous D-MPC, where each local subsystem simultaneously solves their own local OCP once, before communicating it to their neighbors, [32, 13, 12, 27, 1, 52].
- Iterative D-MPC, which uses an iterative distributed optimization approach, where vehicles iterate during a single MPC cycle, until some convergence criteria is met, [43, 44, 9, 11, 53, 8, 45, 46, 30].
- Sequential D-MPC, here, local subsystems solve their local OCP sequentially. In this way, posterior vehicle uses communicated information of the anterior ones, [17, 16, 5]

The distributed controllers discussed in the literature review were not necessarily applied to a vehicle collision avoidance problem, but vital for the understanding of the topic of distributed MPC.

Game Theory

Another important aspect discussed in the literature study is game theory. First, some basic knowledge regarding game theory is recapped, before discussing three different methods to solve a Generalized Potential Game. All three solution methods are adopted from [17], and recapped here:

- An Open-Loop Gauss-Southwell Method, where only a selected player i computes a best response to the strategies adopted by the other players.
- An Open-Loop Gauss-Seidel Method, where all vehicles sequentially update their best response according to a predefined ordering.
- A Closed-Loop Gauss-Seidel Method, where all vehicles follow the same procedure as the open-loop Gauss-Seidel method, but only apply the first temporal step of the equilibrium solution and, successively, play again.

For all solution methods, recursive feasibility is proven in [17].

In this thesis, some methods based on the discussed control methods will be applied to a vehicle collision avoidance problem, together with other control methods not previously mentioned in the literature study.

Chapter 2

Mathematical Background

2-1 Networked Systems

Graph theory is used to model the interconnection between networked systems, e.g., multiple automated vehicles. Graphs represent how different systems are connected to each other. Such a connection is represented by a link (or edge), $e_{ij} = (i,j) \in \mathcal{E}$, that represents the existence of a connection between vertices $i, j \in \mathcal{V}$. Using a weight matrix $\mathcal{W}_{ij} \in \mathbb{R}_{+}^{n\times n}$, weights can be associated with links:

$$\text{weight of } \operatorname{link}(i,j) = \mathcal{W}_{ij} \begin{cases} > 0 & \text{if} \quad (i,j) \in \mathcal{E}, \\ = 0 & \text{if} \quad (i,j) \notin \mathcal{E}. \end{cases} \tag{2-1}$$

We formulate the definition of a graph, adopted from [21, 10]

Definition 2.1 Graphs
A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ is a triple that consists of a set of vertices (or nodes) $\mathcal{V} = \{1, \ldots, n\}$, a set of edges (or links) $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ and a weight matrix $\mathcal{W}$, according to (2-1).

A connection (or link) can have different meanings, depending on the application, e.g., the exchange of data or mass flow between subsystems. Two vertices $i, j \in \mathcal{V}$ are adjacent if there exists an edge $e_{ij} = (i,j) \in \mathcal{E}$. An adjacent vertex j for a vertex i is a neighbor of i. The set of all neighbors of i is denoted by $\mathcal{N}_i$, namely:

$$\mathcal{N}_i = \{j \in \mathcal{V} | j \neq i, (i,j) \in \mathcal{E}\}. \tag{2-2}$$

$\mathcal{G}$ is unweighted if $\mathcal{W}_{ij} \in \{0, 1\}$ for all i, j, i.e., if all existing links have weight 1. If so, $\mathcal{W}$ is called an adjacency matrix and $(\mathcal{V}, \mathcal{E}) \leftrightarrow \mathcal{W}$.

An illustration of a network of autonomous vehicles and their corresponding graph $\mathcal{G}$ and weight matrix $\mathcal{W}$ is shown below. In this case, if vehicles i, j are neighbors, e.g. $i, j \in \mathcal{N}_i$ an edge, $e_{ij} = 1 \in \mathcal{W}$, represents a communication link between vehicles i, j.

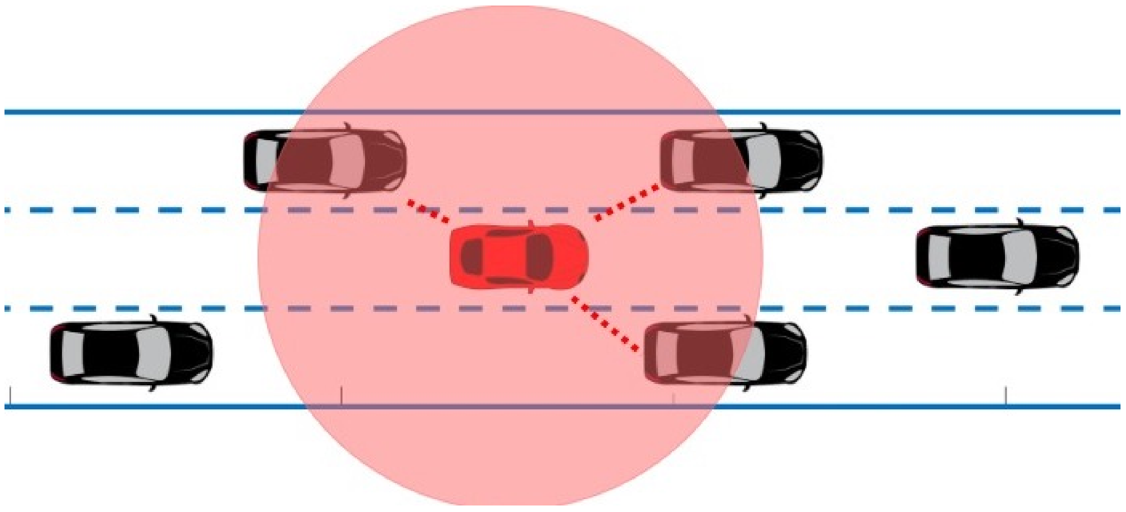


Figure 2-1: Example network of autonomous vehicles

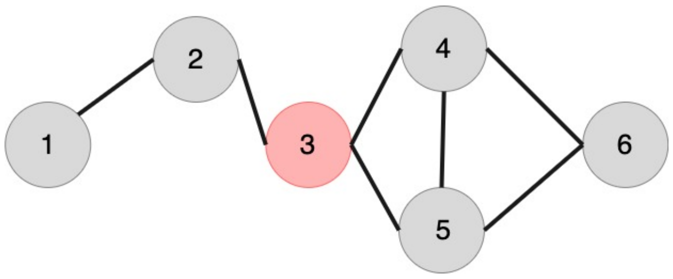


Figure 2-2: Undirected graph representing a network of vehicles

$$\mathcal{W} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0. \end{bmatrix}. \tag{2-3}$$

2-2 Model Predictive Control

Model Predictive Control is used to optimally control a process while ensuring the satisfaction of constraints. The basic idea is to use a dynamical model to predict the system behavior and optimize the predicted inputs over a finite-time prediction and control horizon, H_p and H_u respectively. Then, the first input is applied to the system and the process is repeated in a receding horizon implementation. Important benefits of MPC are the ability to handle constraints and adapt to future changes by iteratively solving the optimization problem. Figure 2-3 and 2-4 shows the concept of MPC for 2 iterations.

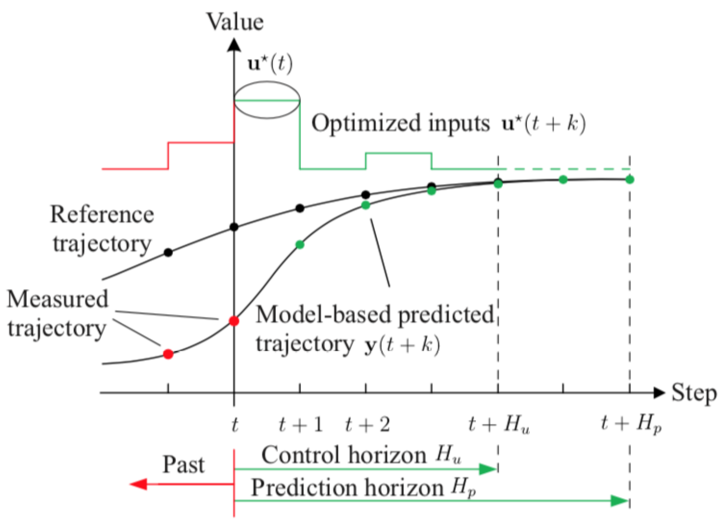


Figure 2-3: Concept of MPC, first iteration [1].

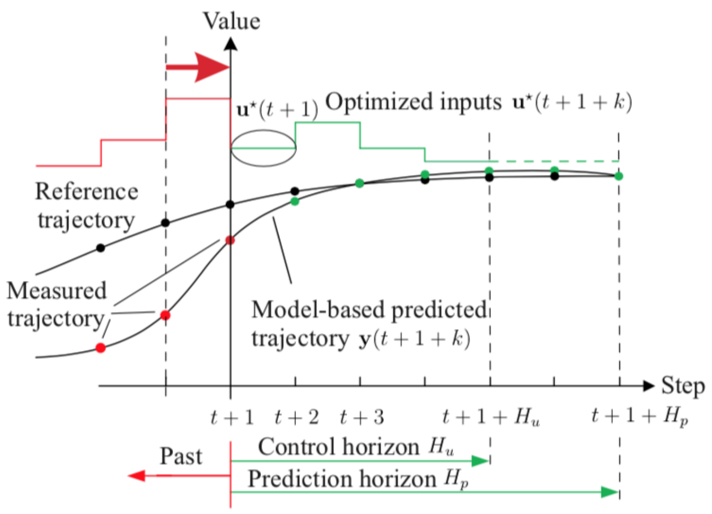


Figure 2-4: Concept of MPC, second iteration [1].

2-2-1 Dynamic Prediction Model

MPC utilizes a dynamical model to predict the future system behavior. A general discrete-time model can be expressed as:

$$\mathbf{x}(t+1) = f(\mathbf{x}(t), \mathbf{u}(t)), \tag{2-4a}$$

$$\mathbf{y}(t) = g(\mathbf{x}(t), \mathbf{u}(t)), \tag{2-4b}$$

where $\mathbf{x}(t) \in \mathbb{R}^n$ represents the states of the system and $\mathbf{u}(t) \in \mathbb{R}^m$ the control inputs. The output is denoted as $\mathbf{y}(t) \in \mathbb{R}^p$. The states evolution is described by the function $f(\mathbf{x}(t), \mathbf{u}(t))$, $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$, and the output function is $g : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^p$. Neglecting future disturbances or measurement noise, a (linearized) prediction model can be set up using the state evolution in (2-5), in order to reduce the computational burden. In this thesis, in the case of a convexified controller, the dynamical model is linearized around the initial conditions of every MPC iteration. Since the vehicles are moving, this linearization is done around a non-equilibrium. Hence, an affine term $\mathbf{E}$ is added to the state matrix $\mathbf{A}$ and input matrix $\mathbf{B}$, this is further explained in section 3-2-1. The future states can be predicted using the system model through iterative substitution.

$$\begin{aligned}
\mathbf{x}(t+1) =& \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{E} \\
\mathbf{x}(t+2) =& \mathbf{A}\mathbf{x}(t+1) + \mathbf{B}\mathbf{u}(t+1) + \mathbf{E} \\
=& \mathbf{A}^2\mathbf{x}(t) + \mathbf{A}\mathbf{B}\mathbf{u}(t) + \mathbf{A}\mathbf{E} + \mathbf{B}\mathbf{u}(t+1) + \mathbf{E} \\
& \vdots \\
\mathbf{x}(t+H_u) =& \mathbf{A}^{H_u}\mathbf{x}(t) + \mathbf{A}^{H_u-1}\mathbf{B}\mathbf{u}(t) + \cdots + \mathbf{B}\mathbf{u}(t+H_u-1) + \\
& \mathbf{A}^{H_u-1}\mathbf{E} + \cdots + \mathbf{E} \\
& \vdots \\
\mathbf{x}(t+H_p) =& \mathbf{A}^{H_p}\mathbf{x}(t) + \mathbf{A}^{H_p-1}\mathbf{B}\mathbf{u}(t) + \cdots + \mathbf{B}\mathbf{u}(t+H_p-1) + \\
& \mathbf{A}^{H_p-1}\mathbf{E} + \cdots + \mathbf{E}
\end{aligned} \tag{2-5}$$

The matrices $\mathbf{A}(t) \in \mathbb{R}^{n\times n}$, $\mathbf{B}(t) \in \mathbb{R}^{n\times m}$, $\mathbf{C}(t) \in \mathbb{R}^{p\times n}$ and $\mathbf{E}(t) \in \mathbb{R}^{n}$ represent the resulting system matrix, input matrix, output matrix and the affine matrix after linearization, respectively. The time argument of the matrices is omitted for simplicity of notation. Please note that a similar prediction model can be composed using a nonlinear model as well. These predicted states can then be used to setup the objective function.

2-2-2 Objective Function

The objective function consist out of the cost to be minimized by the optimization and is divided into 2 parts, the running cost and terminal cost. Generally, the former penalizes the distance of the states and control input, from a reference along the prediction horizon $(t, t+1, \ldots, t+H_p-1)$, to allow for reference tracking. The latter penalizes the weighted distance of the states from the reference at the final prediction step H_p. The objective function reads as:

$$\mathbf{V}(\widetilde{\mathbf{x}}, \widetilde{\mathbf{u}}) = \underbrace{\sum_{k=1}^{H_p-1} \widetilde{\mathbf{x}}(k)^T\mathbf{Q}_i\widetilde{\mathbf{x}}(k) + \sum_{k=1}^{H_u-1} \widetilde{\mathbf{u}}(k)^T\mathbf{R}_i\widetilde{\mathbf{u}}(k)}_{\text{Running cost}} + \underbrace{\widetilde{\mathbf{x}}(H_p)^T\mathbf{P}\widetilde{\mathbf{x}}(H_p)}_{\text{Terminal cost}}, \tag{2-6}$$

where $\widetilde{\mathbf{x}}(k) = \mathbf{x}(k) - \mathbf{x}_{ref}(k)$ and $\widetilde{\mathbf{u}}(k) = \mathbf{u}(k) - \mathbf{u}_{ref}(k)$. Substituting (2-5) in (2-6), the objective function can be rewritten:

$$\mathbf{V}(\widetilde{\mathbf{x}}, \widetilde{\mathbf{u}}) = \widetilde{\mathbf{x}}^T\widetilde{\mathbf{Q}}\widetilde{\mathbf{x}} + \widetilde{\mathbf{u}}^T\mathbf{R}\widetilde{\mathbf{u}}. \tag{2-7}$$

Here, the terminal cost is augmented in the running cost using $\widetilde{\mathbf{Q}} = diag(\mathbf{Q}_1, \ldots.., \mathbf{Q}_{H_p-1}, \mathbf{P})$ and $\mathbf{Q}_i, \mathbf{P} \in \mathbb{R}^{n\times n}$ and $\mathbf{R} \in \mathbb{R}^{m}$. The state and input evolution over H_p and H_u is denoted by $\widetilde{\mathbf{x}}$ and $\widetilde{\mathbf{u}}$, respectively.

$$\widetilde{\mathbf{x}} = \begin{bmatrix} x(1) - x_{ref}(1) \\ x(2) - x_{ref}(2) \\ \vdots \\ x(H_p) - x_{ref}(H_p) \end{bmatrix}, \quad \widetilde{\mathbf{u}} = \begin{bmatrix} u(1) - u_{ref}(1) \\ u(2) - u_{ref}(2) \\ \vdots \\ u(H_u-1) - u_{ref}(H_u-1)] \end{bmatrix}. \tag{2-8}$$

In the case of linearized MPC, the number of optimization variables can be decreased by rewriting (2-7) as a function depending only on the control inputs. Here, a standard QP problem arises, where the optimization variable is the sequence of the predicted inputs. In view of (2-5), the overall optimization problem reads as:

$$\begin{aligned} &\min_{\mathbf{u}_{H_u}} \mathbf{J} && \text{(2-9a)} \\ &\text{s.t. } \mathbf{A}\mathbf{u} \leq \mathbf{b}, && \\ := \min_{\mathbf{u}_{H_u}} \quad &\frac{1}{2}\mathbf{u}_{H_u}^T \mathbf{P}\mathbf{u}_{H_u} + \frac{1}{2}\mathbf{q}\mathbf{u}_{H_u} + r_0 && \\ &\text{s.t. } \mathbf{A}\mathbf{u}_{H_u} \leq \mathbf{b}. && \text{(2-9b)} \end{aligned}$$

Each term in (2-9) is defined as (2-10).

$$\mathbf{u}_{H_u} = \begin{bmatrix} u(1) \\ u(2) \\ \vdots \\ u(H_u - 1) \end{bmatrix}, \qquad \mathbf{P} = \mathbf{S}^T\widetilde{\mathbf{Q}}\mathbf{S} + \mathbf{R}, \qquad \mathbf{q} = \begin{bmatrix} x(k)^T \tilde{\mathbf{x}}_{H_p}^T \tilde{\mathbf{u}}_{H_u} \end{bmatrix} \begin{bmatrix} \mathbf{S}^T\widetilde{\mathbf{Q}}\mathbf{T} \\ \widetilde{\mathbf{Q}}\mathbf{S} \\ \mathbf{R} \end{bmatrix},$$

T and **S** are denoted by:

$$\mathbf{T} = \begin{bmatrix} \mathbf{I} \\ \mathbf{A} \\ \mathbf{A}^2 \\ \vdots \\ \mathbf{A}^N \end{bmatrix}, \mathbf{S} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \cdots & \cdots & \cdots & \mathbf{0} \\ \mathbf{B} & \ddots & & & & \vdots \\ \mathbf{AB} & \ddots & \ddots & & & \vdots \\ \mathbf{A}^2\mathbf{B} & \ddots & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \mathbf{0} \\ \mathbf{A}^{N-1}\mathbf{B} & \cdots & \mathbf{A}^2\mathbf{B} & \mathbf{AB} & \mathbf{B} & \mathbf{0} \end{bmatrix}. \tag{2-10}$$

2-2-3 Constraints

Handling constraints directly in the optimization problem is one main advantage of MPC. This has the following benefits:

- Constraints can be implemented on the control input to ensure physical limits on actuators. If these are not respected, the controller saturates and can cause delays in the system or even destroy physical components.
- Constraints on states and output can ensure safe desired behavior of the system.
- Constraints can guarantee stability of the MPC controller.

How stability can be guaranteed for linear time-invariant MPC, with constraints is explained in Appendix A. In general, constraints result in constrained sets:

$$\begin{aligned} &\mathbf{x}(t+k) \in X \subseteq \mathbb{R}^n, k = 1, \ldots, H_p, \\ &\mathbf{y}(t+k) \in \mathcal{Y} \subseteq \mathbb{R}^p, k = 1, \ldots, H_p, \\ &\mathbf{u}(t+k) \in \mathcal{U} \subseteq \mathbb{R}^m, k = 0, \ldots, H_u - 1. \end{aligned} \tag{2-11}$$

2-3 Game Theory

Game theory is a mathematical method to analyze circumstances where the success of an individual is based upon the choices of the others. Game theory originates from the field of social sciences and economics, but has proven its potential in fields as engineering, computer science, philosophy and many other fields, [33], [46]. In this section we first give a simple example of a discrete game, before continuing with elements of game theory in the continuous space, that are of interest for automated driving applications. We adopt the theory from [29],[19],[18],[20],[30],[46].

An example of a Discrete Game in Normal-Form

The normal form is the most simple representation of strategic interactions in game theory, also known as the strategic or matrix form. A game written in normal form represents all discrete states of each agent and it's corresponding cost or utility. A well known example that can be found in literature is the prisoner's dilemma:

Example 2.1 Prisoner's Dilemma
Two members of a criminal gang are arrested and imprisoned. Each prisoner is in solitary confinement with no means of speaking to or exchanging messages with the other. The police admit they don't have enough evidence to convict the pair on the principal charge. They plan to sentence both to a year in prison on a lesser charge. Simultaneously, the police offer each prisoner a Faustian bargain. If he testifies against his partner, he will go free while the partner will get three years in prison on the main charge. But, there is a catch... If both prisoners testify against each other, both will be sentenced to two years in jail. In normal form, the prisoners dilemma game looks like:

	B refuses deal	*B turns state's evidence*
A refuses deal	*1 year, 1 year*	*3 years, 0 years*
A turns state's evidence	*0 years, 3 years*	*2 years, 2 years*

Or, in more compact and standard form:

$$\begin{array}{c|cc} & R & D \\ \hline R & 1,1 & 3,0 \\ \hline D & 0,3 & 2,2 \\ \hline \end{array} \tag{2-12}$$

Where R denotes a prisoner refusing the deal and D taking the deal, in other words turning state's evidence. Looking at Example 2.1, it is clear that the action of each prisoner would depend on the action of the other, if information could be exchanged. Game theory helps in formulating this in a mathematical manner. In a game, all agents act to self-interest. By this, it is meant that agents are looking for an action that reduces its own cost function, while still taking the actions of other agents into account. This definition of a discrete game can be generalized into a continuous action space.

2-4 Nash Equilibrium Problems

In general, the possible actions of a player might be infinite. For example, the action space can be represented by the whole space $\mathbb{R}^n$, or by a subset. A general game can represented by the interdependent optimization problems:

$$\forall i \in \mathcal{V} = \{1, \ldots, N\}, \quad \min_{x_i \in \mathcal{X}_i} V_i\left(x_i, \boldsymbol{x}_{-i}\right), \tag{2-13}$$

where $\mathcal{V}$ is the set of all players, x_i is the strategy of agent i, V_i is the cost function of agent i, that depends both on its own strategy and on the strategies of the other agents $\boldsymbol{x}_{-i} = (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$. The most studied notion of solution for the game (2-13) is the Nash Equilibrium (NE). To understand the concept of a NE, we will look at games from an individual agent's point of view. When an agents knows the strategy of other agents, his strategic problem becomes simple. Specifically, he would be left with the single-agent problem of choosing a cost function minimizing action. If the agents $-i$ were to commit to play $\boldsymbol{x}_{-i}$, agent i would face the problem of determining his best response.

Definition 2.2 Best Response
Player i's best response to the strategy profile $\boldsymbol{x}_{-i}$ is a mixed strategy $x_i^* \in \mathcal{X}_i$ such that $V_i\left(x_i^*, \boldsymbol{x}_{-i}\right) \leq V_i\left(x_i, \boldsymbol{x}_{-i}\right)$ for all strategies $x_i \in \mathcal{X}_i$.

Looking at Example 2.1, if prisoner B knows that prisoner A will refuse the deal, his best response is turning state's evidence. The best response is not necessarily unique and is not a solution concept. It does not identify an interesting set of outcomes in this general case. However, the notion of best response helps to define what is arguably the most central notion in game theory, the Nash Equilibrium.

Definition 2.3 Nash Equilibrium
A strategy profile $x_i \in \mathcal{X}$ is a NE if, for all agents i, x_i is a best response to $\boldsymbol{x}_{-i}$. Mathematically, this means:

a tuple $x^* \equiv (x_i^*)_{i=1}^N \in \widehat{\mathcal{X}} \triangleq \prod_{i=1}^N \mathcal{X}_i$ is a NE of the game $\mathcal{G}$ defined by the pair $(\widehat{\mathcal{X}}, \Theta)$, where Θ denotes the tuple $(V_i)_{i=1}^N$, if, for every $i = 1, \ldots, N$,

$$V_i\left(x_i^*, \boldsymbol{x}_{-i}^*\right) \leq V_i\left(x_i, \boldsymbol{x}_{-i}^*\right), \quad \forall x_i \in \mathcal{X}_i. \tag{2-14}$$

Example 2.2 Nash Equilibrium
Consider a game with 2 players each with one decision variable; that is, $n_1 = n_2 = 1$, and thus $n = 2$. For simplicity, we let $x \in \mathbb{R}$ and $y \in \mathbb{R}$ denote these 2 players' strategies, respectively. Let the players' problems be

$$\begin{array}{ll} \underset{x}{\text{minimize}} & (x-y)^2 \\ \textit{subject to} & 0 \leq x \leq 1 \end{array} \quad \textit{and} \quad \begin{array}{ll} \underset{y}{\text{minimize}} & xy + y^2 \\ \textit{subject to} & -1 \leq y \leq 1 \end{array}, \tag{2-15}$$

The optimal solutions are given by:

$$\mathcal{S}_1(y) = \begin{cases} 0 & \textit{if } y < 0 \\ y & \textit{if } 0 \leq y \leq 1 \\ 1 & \textit{if } y > 1 \end{cases} \quad \textit{and} \quad \mathcal{S}_2(x) = \begin{cases} 1 & \textit{if } x < -2 \\ -x/2 & \textit{if } -2 \leq x \leq 2 \\ -1 & \textit{if } x > 2 \end{cases}. \tag{2-16}$$

It is easy to check that the unique fixed point of the map: $\mathcal{S}_1 \times \mathcal{S}_2$, in other words, a pair (x, y) such that $x = \mathcal{S}_1(y)$ and $y = \mathcal{S}_2(x)$, is (0,0) which is the unique NE of this game.

In simple terms, a NE is a stable strategy profile, from which no agent has interest to unilaterally change its strategy.

In some applications, players might not care about changing their strategies to a best response when the decrease in cost is under a certain threshold. This leads to the idea of an ϵ-NE.

Definition 2.4 ϵ-Nash Equilibrium
Fix $\epsilon > 0$. A strategy profile $\boldsymbol{x} = (x_1, \ldots, x_n)$ is an ϵ-Nash equilibrium if, for all agents i and for all strategies $x_i' \neq x_i, V_i(x_i, \boldsymbol{x}_{-i}) \geq V_i(\boldsymbol{x}_{-i}', \boldsymbol{x}_{-i}) - \epsilon$.

A major benefit of ϵ-Nash Equilibria, is that they always exist for a certain $\epsilon > 0$. Further, algorithms that aim to identify ϵ-Nash Equilibria need to consider only a finite set of strategy profiles rather than the whole continuous space. However, "*there is no such thing as a free lunch*", ϵ-Nash Equilibria also have some drawbacks, e.g., Nash Equilibria are always surrounded by ϵ-Nash Equilibria, but the opposite is not true. A given ϵ-Nash Equilibrium is not necessarily close to a Nash Equilibrium. This is not further explained here, but the interested reader is referred to [29, Section 3.7].

2-4-1 Generalized Nash Equilibrium Problems

In this section we consider the concept of a Generalized Nash Equilibrium Problem (GNEP). The GNEP extends the classical Nash Equilibrium Problem by assuming that each player's feasible set can depend on the rival players' strategies.

Definition 2.5 Generalized Nash Equilibrium Problems
The feasible set of each player, dependent on other player's strategies, is defined as $\mathcal{X}_i(\boldsymbol{x}_{-i}) \subseteq \mathbb{R}^{n_i}$. Given the other player's strategies, the aim of player i is to choose a strategy x_i that solves the minimization problem

$$\min_{x_i \in \mathcal{X}_i(\bar{\boldsymbol{x}}_{-i})} V_i(x_i, \boldsymbol{x}_{-i}), \tag{2-17a}$$

$$s.t.\ x_i \in \mathcal{X}_i(\boldsymbol{x}_{-i}). \tag{2-17b}$$

A collective strategy $\bar{x}_i$ is called a GNE if, $\forall i$, x_i solves the above minimization problem, i.e.:

$$V_v(\bar{x}_i, \bar{\boldsymbol{x}}_{-i}) \leq V_v(y_i, \bar{\boldsymbol{x}}_{-i}), \quad \forall y_i \in \mathcal{X}_i(\bar{\boldsymbol{x}}_{-i}), \quad \forall i \in \mathcal{V}. \tag{2-18}$$

A Generalized Nash Equilibrium (GNE) is a point $\bar{\mathbf{x}}$ in which no player can decrease his objective function by changing unilaterally $\bar{x}_i$ to any other feasible point. An example of a GNEP is given below.

Example 2.3 GNEP
Consider a game with two players, i.e. $N = 2$, with $n_1 = 1$ and $n_2 = 1$, so that each player controls one variable.

Assume that the players' problems are:

$$\begin{array}{ll} \min_{x^1} \left(x^1 - 1\right)^2 & \min_{x^2} \left(x^2 - \frac{1}{2}\right)^2 \\ s.t. \quad x^1 + x^2 \leq 1, & s.t. \quad x^1 + x^2 \leq 1. \end{array} \tag{2-19}$$

The optimal solution sets are given by:

$$\mathcal{S}_1\left(x^2\right)=\begin{cases}1, & \textit{if } x^2\leq 0,\\ 1-x^2, & \textit{if } x^2\geq 0\end{cases} \quad \textit{and} \quad \mathcal{S}_2\left(x^1\right)=\begin{cases}\frac{1}{2}, & \textit{if } x^1\leq\frac{1}{2}\\ 1-x^1, & \textit{if } x^1\geq\frac{1}{2}.\end{cases} \tag{2-20}$$

Then it is easy to check that the GNE's of this problem are given by $(\alpha, 1-\alpha)$ *for every* $\alpha\in[1/2,1]$*. Note that the problem has infinitely many solutions.*

Lastly, the concept of a Generalized Potential Game (GPG) is introduced, which is an instance of the GNEP class. Roughly speaking, "a GPG is a GNEP where the players are (unknowingly) minimizing the same function and where the feasible set of each player is the section of a larger set in the product space $\mathbb{R}^n$", [40, 20].

Definition 2.6 Generalized Potential Game
A GNEP is a Generalized Potential Game if:

(a) *There exists a nonempty, closed set* $\mathcal{X}\subseteq\mathbb{R}^n$ *such that, for all* $i=1,\dots,N$,

$$\mathcal{X}_i\left(\mathbf{x}_{-i}\right)\equiv\left\{x_i\in\mathcal{X}_i:\left(x_i,\mathbf{x}_{-i}\right)\in\mathcal{X}\right\}, \tag{2-21}$$

where $\mathcal{X}_i\subseteq\mathbb{R}^{n_i}$ *are nonempty, closed sets such that* $\prod_{i=1}^N\mathcal{X}_i\cap\mathcal{X}\neq\emptyset$*, (i.e. the "feasible set" of the game is non empty).*

(b) *There exists a continuous function* $P(x):\mathbb{R}^n\to\mathbb{R}$ *such that for all* i*, for all* $\boldsymbol{x}_{-i}$ *(such that* $\mathcal{X}_i\left(\boldsymbol{x}_{-i}\right)$ *is not empty), and for all* $y_i,z_i\in\mathcal{X}_i\left(\boldsymbol{x}_{-i}\right)$,

$$V_i\left(y_i,\mathbf{x}_{-i}\right)-V_i\left(z_i,\mathbf{x}_{-i}\right)>0, \tag{2-22}$$

implies

$$P\left(y_i,\mathbf{x}_{-i}\right)-P\left(z_i,\mathbf{x}_{-i}\right)\geq\sigma\left(V_i\left(y_i,\mathbf{x}_{-i}\right)-V_i\left(z_i,\mathbf{x}_{-i}\right)\right). \tag{2-23}$$

where $\sigma:\mathbb{R}_+\to\mathbb{R}_+$ *is a forcing function:* $\lim_{k\to\infty}\sigma\left(t_k\right)=0\Rightarrow\lim_{k\to\infty}t_k=0$.

Chapter 3

Problem Statement

3-1 Automated Driving

In many automated driving applications, the goal is to achieve coordination of a large number of vehicles, for safe and efficient operation.

Controlling such a large network by a single processing unit not only requires high computational loads, that increases for an increasing number of subsystems, but it's also difficult to maintain, since all subsystems rely upon this central agent. In literature, this is overcome by distributing the plantwide OCP among all subsystems. In order to investigate the advantages and disadvantages of different distributed MPC algorithms, an objective analysis and comparison is performed in this thesis. More specifically, in this thesis we study:

- **Centralized MPC**, where all subsystems $i \in \mathcal{V}$ are controlled by a central agent. This is prone to give the plantwide Pareto optimal solution and will be used as benchmark.
- **Decentralized MPC**, where each local subsystem synchronously computes a new trajectory $\mathbf{x}_i$, based on the previously computed (and communicated) trajectories of its neighbors $j \in \mathcal{N}_i \subseteq \mathcal{V}$,
- **Cooperative Distributed MPC**, where each local subsystem i performs multiple sub-optimal iterations p of a Gauss-Jacobi type distributed optimization, by communicating its trajectory $\mathbf{x}_i$ to its neighbors $j \in \mathcal{N}_i$ every algorithmic iteration, before applying a new input $\mathbf{u}_i$.
- **Non-cooperative Distributed MPC**, where each local subsystem i sequentially solves and communicates their trajectory $\mathbf{x}_i$ to its neighbors $j \in \mathcal{N}_i \cap \mathcal{O} \subset \mathcal{V}$, in order to find an ϵ-Nash Equilibrium. Here, $\mathcal{O}$ represents the set of higher order vehicles.

All four controllers are explained in-depth in chapter 4. How each vehicle is modelled and how the collision avoidance constraints are composed is explained in the following sections.

3-2 Vehicle Modelling

3-2-1 Vehicle Kinematics

We use a bicycle model to represent the kinematics of each vehicle. An advantage of using the bicycle model instead of using a more complex dynamical model is the reduction in computational time of every MPC iteration. Although the kinematic model is relatively simple, it is accurate enough to represent the trajectory of a vehicle and can be used to address collision avoidance.

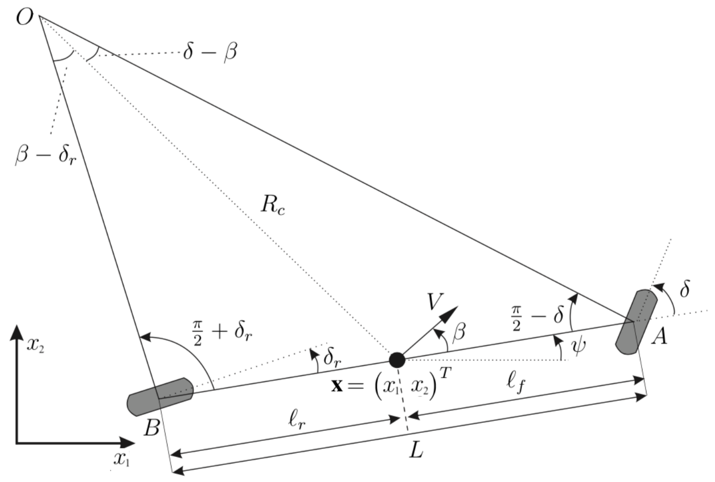


Figure 3-1: Kinematic bicycle model of the vehicle,[2, eq. (1)].

Using a kinematic bicycle model implies the following assumptions:

- The front and rear axle are each represented by a single wheel, centered between the actual tyres position of each axle.
- The vehicle movement is assumed to be planar, i.e. pitch and roll dynamics of the vehicle are neglected
- Resistant forces are neglected, i.e. aerodynamic drag and rolling resistance are zero.
- No slip occurs at the tyres, hence the velocity vector of each tyre is parallel to each tyre, i.e. the driving direction is parallel to the direction of wheel travel.
- No external forces are applied to the front and rear tyres.
- The side-slip angle $\beta(t)$ (fig. 3-1), i.e., the angular difference between the velocity vector V and the longitudinal direction of the vehicle, is neglected.

3-2-2 Vehicle Model

Nonlinear Vehicle Model

The nonlinear kinematic bicycle model is:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}\left(\boldsymbol{x}(t), \boldsymbol{u}(t)\right) = \begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{\vartheta}(t) \\ \dot{\psi}(t) \\ \dot{v}(t) \end{bmatrix} = \begin{bmatrix} v(t)\cos(\vartheta(t)) \\ v(t)\sin(\vartheta(t)) \\ \frac{v(t)}{L}\tan(\psi(t)) \\ \omega(t) \\ a(t) \end{bmatrix}, \tag{3-1}$$

where $\boldsymbol{f} : \mathbb{R}^5 \times \mathbb{R}^2 \rightarrow \mathbb{R}^5$ is a continuous nonlinear smooth function, $\dot{x}_1(t)$ and $\dot{x}_2(t)$ represent the vehicle's longitudinal and lateral velocity with respect to the global coordinate axis, $\vartheta(t)$ represents the vehicle's yaw angle relative to the positive x-axis, $v(t)$ the vehicle's velocity with respect to its body frame. The inputs are steering rate $\omega_i(t)$, and the acceleration $a_i(t)$. Note that all positions (and velocities) are measured from the vehicle's Center of Gravity (CoG).

Linear Vehicle Model

For controllers based on game theory or some based on distributed optimization, convexity is a prerequisite for convergence. As a result, the vehicle model and constraints need to be linearized and convexified. Hence, the non-linear vehicle model is linearized around the initial conditions of every MPC iteration: $\boldsymbol{x}_0, \boldsymbol{u}_0$. Optionally, to increase accuracy of the linearized model, it is possible to linearize around a shifted optimal trajectory from the previous MPC step. The resulting (continuous-time) state-space model is of the form:

$$\begin{aligned} \dot{x}(t) &= \dot{x}_0(t) + \dot{\Delta x}(t), \\ \dot{x}(t) &= \dot{x}_0(t) + \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}_0} \Delta x(t) + \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right|_{\boldsymbol{u}_0} \Delta u(t), \\ \dot{x}(t) &= E + A\Delta x(t) + B\Delta u(t), \end{aligned} \tag{3-2}$$

where,

$$E = \begin{bmatrix} v_0 sin(\theta_0)\theta_0 \\ -v_0 cos(\theta_0)\theta_0 \\ \frac{-v_0}{Lcos^2(\psi_0)}\psi_0 \\ 0 \\ 0 \end{bmatrix}, \quad A\Delta x(t) = \begin{bmatrix} -v_0 sin(\theta_0)\theta_\Delta + v_\Delta cos(\theta_0) \\ v_0 cos(\theta_0)\theta_\Delta + v_\Delta sin(\theta_0) \\ \frac{-v_0}{Lcos^2(\psi_0)}\psi_\Delta + \frac{v_\Delta tan(\psi_0)}{L} \\ 0 \\ 0 \end{bmatrix}, B\Delta u(t) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \omega \\ a \end{bmatrix}. \tag{3-3}$$

After successful linearization, all matrices are discretized using a zero-order hold approximation.

3-3 Collision Avoidance Modelling

To avoid collision between vehicles, constraints are imposed so that vehicles are not allowed to drive to close to each other. The feasible set of the optimization consist of the complement of multiple forbidden convex sets, see figure 3-2.

The green area is the feasible set of motion of vehicle v_1. The red area is an infeasible set of motion of vehicle v_1. Hence, the domain of vehicle v_1's motion is non-convex and therefore the motion planning OCP is non-convex. Different solution methods are used to handle the non-convex OCP, depending on the type of controller needed.

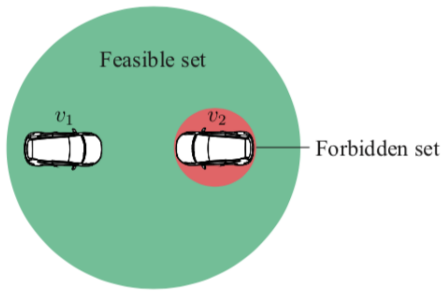


Figure 3-2: Non-convexity of the collision avoidance constraints, [1].

Nonlinear Programming Constraints

A major benefit of Non-Linear Programming (NLP) is the ability of handling non-convex constraints and nonlinear equations of motion. This gives a more accurate model compared to linearized models. These benefits come at the cost of higher computational effort.

NLP Hard constraints For general vehicle collision avoidance between multiple moving vehicles or obstacles, the following constraint is used. Here $\mathbf{x}_i(t+k)$ denotes the prediction of the position of the i-th vehicle at the k-th sample, H_p is the prediction horizon, $d_{\text{safe}}^{(i,j)}$ is a predefined safety distance between vehicles i and j.

$$\forall k \in 1, \ldots, Hp, \forall i, j \in \mathcal{V}, i \neq j, \quad \|\mathbf{x}_i(t+k) - \mathbf{x}_j(t+k)\| \geq d_{\text{safe}}^{(i,j)}, \tag{3-4}$$

Note that this constraint can also be used to avoid collision with static obstacles by simply changing the time varying position vector $\mathbf{x}_j(t+k)$ with that of a static obstacle, e.g., $\mathbf{v}_p$. For nonlinear centralized control, the above constraint can be used to avoid collision between vehicles or obstacles. Unfortunately, for distributed control, implementing (3-4) is not possible due to the non-convexity. Convexity is a prerequisite for decomposing the problem formulation into smaller subproblems used for distributed optimization. In literature, this problem is generally overcome by convexifying the constraints for collision avoidance instead of hard constraints

NLP Soft constraints By imposing cost in the objective function, to ensure that vehicles will not come close to each other, collisions can also be avoided. These collision avoidance potential function are referred to as soft constraints.

We consider collision cost that is inversely proportional to the distance between vehicles. Resulting in the potential function:

$$\sum_{j \in \mathcal{N}, j \neq i} \sum_{k=1}^{H_p - 1} \gamma_3 \left\| L_{col}\left(\mu_{ij}(t+k)\right)\right\|^2, \tag{3-5a}$$

$$L_{col}(t) = \frac{1}{\|\mathbf{x}_i(t+k) - \mathbf{x}_j(t+k)\| - d_{safe}^{(i,j)}} \tag{3-5b}$$

$$D_i = \sqrt{(a/2)^2 + (L/2+b)^2}. \tag{3-5c}$$

Here, $\mathbf{x}_i$ represents the states of subsystem i and γ_i represents a positive weighting factor corresponding to the collision avoidance cost. It can be seen clearly from (3-5b) that vehicles closer to each other correspond to higher cost values. When properly tuned, this cost ensures that vehicles will avoid each other, since the vehicles are inclined to lower cost area's in the road space-time due to the repulsive force of the potential functions.

Mixed Integer Programming

A method commonly used in literature to convexify non-convex constraints is Mixed Integer Programming (MIP). MIP is a powerful tool for planning and control problems because of its modelling capabilities and the availability of good solvers.

Linear constraints are used to convert a non-convex problem into a linear/quadratic convex programming problem. Each vehicle is mathematically represented by a polyhedron, e.g., a combination of linear constraints.

MIP Vehicle Collision Avoidance Constraints For general vehicle collision avoidance between multiple moving vehicles, (3-4) is used as starting point. The ∞-norm is used to convert this quadratic function to a linear approximation. This results in the following:

$$\begin{aligned} |x_i(t+k) - x_j(t+k)| &\geq d_{\text{safe}}^{(i,j)}, k = 1, \ldots, H_p, \\ \text{OR} \quad |y_i(t+k) - y_j(t+k)| &\geq d_{\text{safe}}^{(i,j)}, k = 1, \ldots, H_p. \end{aligned} \tag{3-6}$$

This can be expressed as "greater than" OR logic statements. These OR statements can be expressed as AND statements using binary variables, this method is referred to as the Big-M method. To reduce the total number of binary variables, a method first introduced in [37] is used to reduce the total number of binary variables from 4 to 2 for each vehicle or obstacle. Note that the variables x_i and y_i change over time, but their argument is left out for simplicity of notation.

$$\begin{aligned} x_j - x_i &\geq d_{\text{safe}}^{(i,j)}, \quad v_i \text{ is to the left of } v_j, \\ \text{OR} \quad x_i - x_j &\geq d_{\text{safe}}^{(i,j)}, \quad v_i \text{ is to the right of } v_j, \\ \text{OR} \quad y_j - y_i &\geq d_{\text{safe}}^{(i,j)}, \quad v_i \text{ is below } v_j, \\ \text{OR} \quad y_i - y_j &\geq d_{\text{safe}}^{(i,j)}, \quad v_i \text{ is above } v_j. \end{aligned} \tag{3-7}$$

$$\begin{aligned} x_j - x_i &\geq d_{\text{safe}}^{(i,j)} - \mathrm{M}\left(\mathrm{c}_1^{(i,j)}(t+k) + \mathrm{c}_2^{(i,j)}(t+k)\right), \\ \text{AND} \quad x_i - x_j &\geq d_{\text{safe}}^{(i,j)} - \mathrm{M}\left(1 - \mathrm{c}_1^{(i,j)}(t+k) + \mathrm{c}_2^{(i,j)}(t+k)\right), \\ \text{AND} \quad y_j - y_i &\geq d_{\text{safe}}^{(i,j)} - \mathrm{M}\left(1 + \mathrm{c}_1^{(i,j)}(t+k) - \mathrm{c}_2^{(i,j)}(t+k)\right), \\ \text{AND} \quad y_i - y_j &\geq d_{\text{safe}}^{(i,j)} - \mathrm{M}\left(2 - \mathrm{c}_1^{(i,j)}(t+k) - \mathrm{c}_2^{(i,j)}(t+k)\right). \end{aligned} \tag{3-8}$$

Here, $\mathrm{c}_{1,2}(i,j)(t+k) \in \{0,1\}$ is a set of binary variables and M is a positive number that is much larger than any position or velocity to be encountered in the problem. Note that also the binary variables $c_1^{(i,j)}, c_2^{(i,j)} \in \{0,1\}$ change over time. When using the Big-M method, only 2 binary variables are used between 2 vehicles at every step in the horizon, e.g., for any combination of c_1 and c_2, exactly one constraint is active as shown in below truth-table 3-1.

Truth Table			
c_1	c_2	Formula	Active Constraint
0	0	$c_1 + c_2$	first
1	0	1-$c_1 + c_2$	second
0	1	1+$c_1 - c_2$	thirth
1	1	2-$c_1 - c_2$	fourth

Table 3-1: Truth table showing the active constraints using the Big-M method

MIP Obstacle Collision Avoidance Constraints Obstacle avoidance using MIP can be done similar as vehicle to vehicle collision avoidance. For simplicity, obstacles are represented as static 4-sided polyhedrons (N = 4) in the 2D-space, but this method can be easily extended to higher dimension polyhedrons with $N > 4$ andmoving in certain directions.

Figure 3-3 shows the representation of an obstacle modeled as a 4-sided polyhedron. The obstacle is constructed by defining 4 outward normal vectors $\mathbf{q}_i - \mathbf{y}^{(p)}$, where $\mathbf{q}_i \in \mathcal{R}^2, i = 1, \dots, 4$ represents the middle point of each of the four sides.

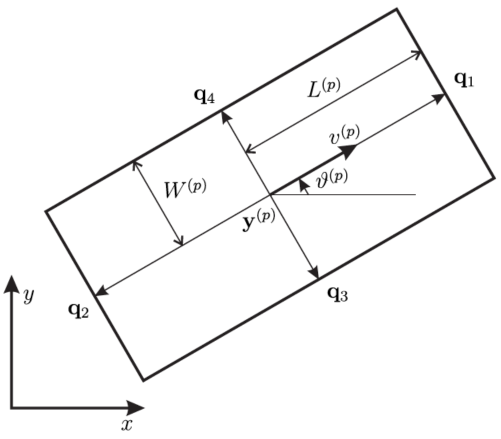


Figure 3-3: Representation of a 4-sided polyhedron obstacle,[1].

Using the definition of a halfspace, the $i-th$ halfspace used to form the side of an obstacle is defined as $\left(\mathbf{q}_i - \mathbf{y}^{(p)}\right)(\mathrm{x} - \mathbf{q}_i) \le 0$. The area of obstacle v_p at point $(x \quad y)^T$ is defined using these halfspaces to represent a polyhedron, [1, section 4.4.2]:

$$v_p = \left\{(x \quad y)^T \in \mathbb{R}^2 |\ \mathbf{A}(x \quad y)^T \le \mathbf{b}\right\}, \tag{3-9}$$

where:

$$\mathbf{A} = \begin{pmatrix} \left(\mathbf{q}_1 - \mathbf{y}^{(p)}\right)^T \\ \left(\mathbf{q}_2 - \mathbf{y}^{(p)}\right)^T \\ \left(\mathbf{q}_3 - \mathbf{y}^{(p)}\right)^T \\ \left(\mathbf{q}_4 - \mathbf{y}^{(p)}\right)^T \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} \left(\mathbf{q}_1 - \mathbf{y}^{(p)}\right)^T \mathbf{q}_1 \\ \left(\mathbf{q}_2 - \mathbf{y}^{(p)}\right)^T \mathbf{q}_2 \\ \left(\mathbf{q}_3 - \mathbf{y}^{(p)}\right)^T \mathbf{q}_3 \\ \left(\mathbf{q}_4 - \mathbf{y}^{(p)}\right)^T \mathbf{q}_4 \end{pmatrix}. \tag{3-10}$$

Expressing these equations in terms of position $\mathbf{y}^{(p)}$, orientation $\vartheta^{(p)}$ and obstacle dimensions, width $W^{(p)}$ and length $L^{(p)}$, the area of an obstacle can be described by the following inequality:

$$\begin{pmatrix} \cos\left(\vartheta^{(p)}\right) & \sin\left(\vartheta^{(p)}\right) \\ -\cos\left(\vartheta^{(p)}\right) & -\sin\left(\vartheta^{(p)}\right) \\ \sin\left(\vartheta^{(p)}\right) & -\cos\left(\vartheta^{(p)}\right) \\ -\sin\left(\vartheta^{(p)}\right) & \cos\left(\vartheta^{(p)}\right) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \leq \begin{pmatrix} L^{(p)} + x^{(p)}\cos\left(\vartheta^{(p)}\right) + y^{(p)}\sin\left(\vartheta^{(p)}\right) \\ L^{(p)} - x^{(p)}\cos\left(\vartheta^{(p)}\right) - y^{(p)}\sin\left(\vartheta^{(p)}\right) \\ W^{(p)} + x^{(p)}\sin\left(\vartheta^{(p)}\right) - y^{(p)}\cos\left(\vartheta^{(p)}\right) \\ W^{(p)} - x^{(p)}\sin\left({}^{(p)}\right) + y^{(p)}\cos\left(\vartheta^{(p)}\right) \end{pmatrix}. \tag{3-11}$$

In order for vehicles to avoid obstacles, the area of a vehicle i should be outside the area of an obstacle v_p. Similar to vehicles collision avoidance constraints, this can be defined in ”greater than” OR logic statements:

$$\begin{aligned} & \cos\left(\vartheta^{(p)}\right)x^{(i)} + \sin\left(\vartheta^{(p)}\right)y^{(i)} \geq \\ & L^{(p)} + x^{(p)}\cos\left(\vartheta^{(p)}\right) + y^{(p)}\sin\left(\vartheta^{(p)}\right) \\ \text{OR} \quad & -\cos\left(\vartheta^{(p)}\right)x^{(i)} - \sin\left(\vartheta^{(p)}\right)y^{(i)} \geq \\ & L^{(p)} - x^{(p)}\cos\left(\vartheta^{(p)}\right) - y^{(p)}\sin\left(\vartheta^{(p)}\right) \\ \text{OR} \quad & \sin\left(\vartheta^{(p)}\right)x^{(i)} - \cos\left(\vartheta^{(p)}\right)y^{(i)} \geq \\ & W^{(p)} + x^{(p)}\sin\left(\vartheta^{(p)}\right) - y^{(p)}\cos\left(\vartheta^{(p)}\right) \\ \text{OR} \quad & -\sin\left(\vartheta^{(p)}\right)x^{(i)} + \cos\left(\vartheta^{(p)}\right)y^{(i)} \geq \\ & W^{(p)} - x^{(p)}\sin\left(\vartheta^{(p)}\right) + y^{(p)}\cos\left(\vartheta^{(p)}\right). \end{aligned} \tag{3-12}$$

Again, using binary variables and the Big-M method, these ”greater than” OR logic statements can be converted to "smaller than" AND logic statements. The MIP obstacle collision avoidance constraints become:

$$\begin{aligned} & \cos\left(\vartheta^{(p)}\right)x^{(i)} + \sin\left(\vartheta^{(p)}\right)y^{(i)} \geq L^{(p)} + x^{(p)}\cos\left(\vartheta^{(p)}\right) + \\ & \qquad y^{(p)}\sin\left(\vartheta^{(p)}\right) - \mathrm{M}\left(\mathrm{c}_1^{(i,p)} + \mathrm{c}_2^{(i,p)}\right) \\ \text{AND} \quad & -\cos\left(\vartheta^{(p)}\right)x^{(i)} - \sin\left(\vartheta^{(p)}\right)y^{(i)} \geq L^{(p)} - x^{(p)}\cos\left(\vartheta^{(p)}\right) - \\ & \qquad y^{(p)}\sin\left(\vartheta^{(p)}\right) - \mathrm{M}\left(1 + \mathrm{c}_1^{(i,p)} - \mathrm{c}_2^{(i,p)}\right) \\ \text{AND} \quad & \sin\left(\vartheta^{(p)}\right)x^{(i)} - \cos\left(\vartheta^{(p)}\right)y^{(i)} \geq W^{(p)} + x^{(p)}\sin\left(\vartheta^{(p)}\right) - \\ & \qquad y^{(p)}\cos\left(\vartheta^{(p)}\right) - \mathrm{M}\left(2 - \mathrm{c}_1^{(i,p)} - \mathrm{c}_2^{(i,p)}\right) \\ \text{AND} \quad & -\sin\left(\vartheta^{(p)}\right)x^{(i)} + \cos\left(\vartheta^{(p)}\right)y^{(i)} \geq W^{(p)} - x^{(p)}\sin\left(\vartheta^{(p)}\right) + \\ & \qquad y^{(p)}\cos\left(\vartheta^{(p)}\right) - \mathrm{M}\left(1 - \mathrm{c}_1^{(i,p)} + \mathrm{c}_2^{(i,p)}\right), \end{aligned} \tag{3-13}$$

where $c_1^{(i,p)}, c_2^{(i,p)} \in \{0,1\}$.

Miscellaneous Collision Avoidance Constraints

Some additional Quadratically Constraint Quadratic Programming (QCQP) methods to convexify collision avoidance constraints are recapped here. These methods are explained in detail in the literature study.

- Sequential Convex Programming (SCP), uses a sequence of affine approximations of the non-convex constraints, that form a conservative approximation of the actual obstacles or vehicle, which is improved every SCP iteration, see figure 3-4.

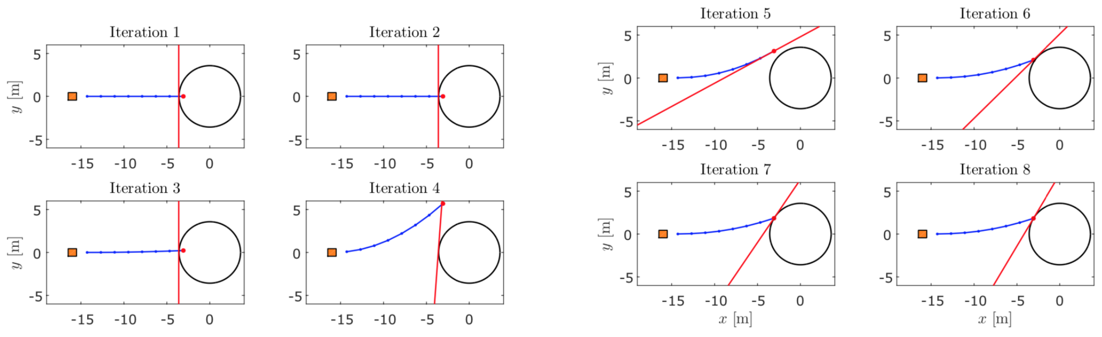


Figure 3-4: Sequential Convex Programming Iterations, [3].

- Semi-Definite Programming Relaxations (SDPR), uses convex positive semi-definite inequality constraints and the Schur-decomposition to relax the original non-convex constraints to a QCQP problem.

3-3-1 Constraint Comparison

All the different types of collision avoidance constraints described earlier, have their own characteristics. To investigate the effect of these different constraints, we do a small comparison between the methods that are applied in this thesis, i.e., MIP and NLP.

A vehicle is approaching an obstacle of 3.25 m (single lane width is 3.5 m) and will have to avoid it to continue driving. The contours of the obstacle and vehicle representation depends on the type of constraint used. The actual vehicles and obstacle are presented as the colored shapes and the outlined shapes are the prediction of the vehicle states in the prediction horizon of the MPC.

The mixed integer constraints (3-13) used in this thesis, form a four-sided polyhedron in $\mathbb{R}^2$. This results in a larger forbidden area as apposed to the NLP quadratic circle constraint (3-4), in $\mathbb{R}^2$. Therefore, using the MIP constraints will require more evasive steering action, as can be seen in figure 3-5.

The quadratic potential function (3-5), also forms a circle in $\mathbb{R}^2$. The cost of this potential function approaches infinity when approaching the border of the obstacle. In this case, the potential function acts as a non-decreasing monotonic cost function. Hence, there is also cost associated with a vehicle close to the obstacle. As a result, a vehicle does not only avoid the border of the vehicle, it also avoids the neighboring free space up to a certain distance, depending on the weights given to the potential function.

Clearly, if the area of a vehicle can be represented by a circle in $\mathbb{R}^2$, the quadratic circle constraints are the most beneficial since these require the least evasive action.

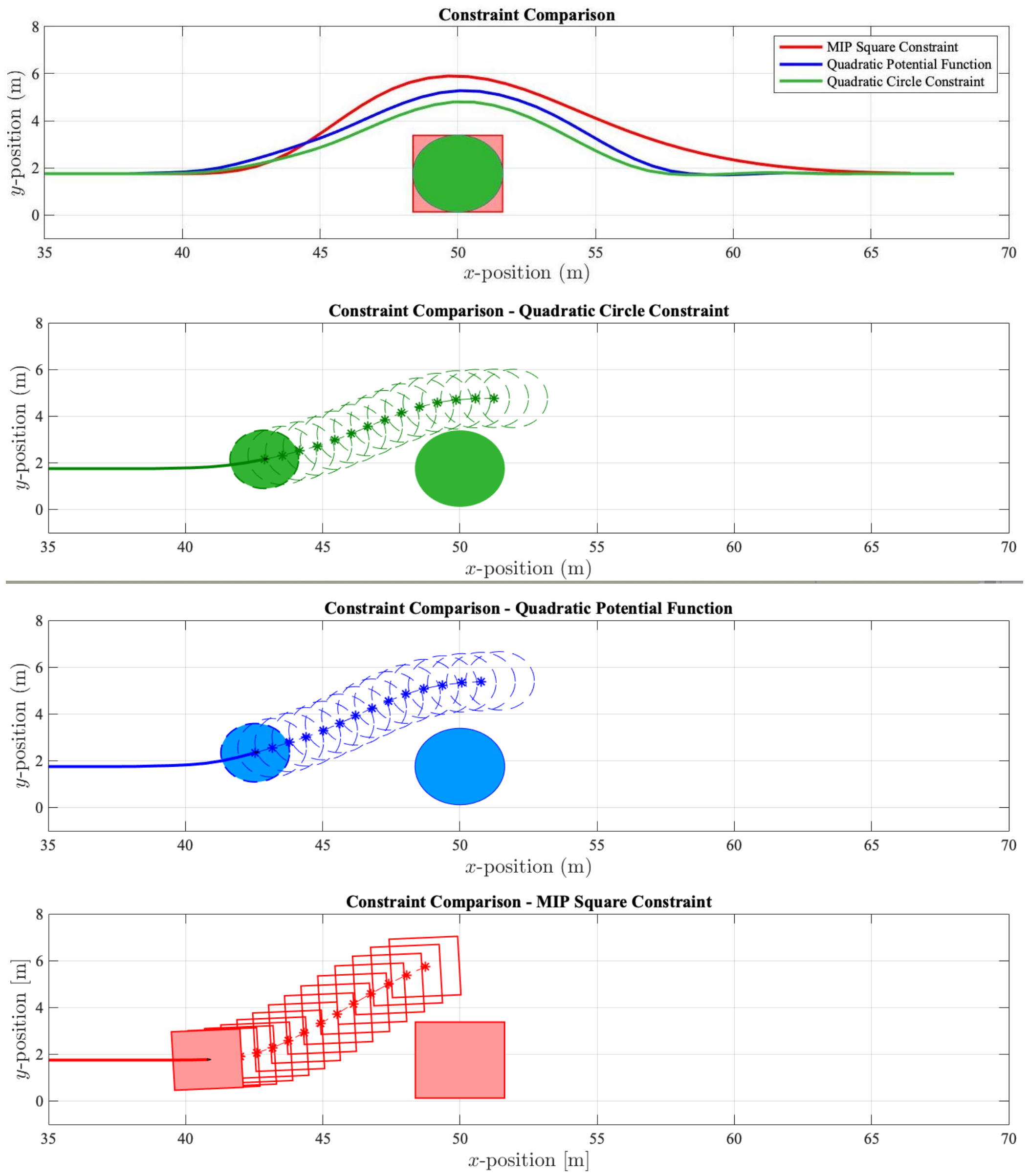


Figure 3-5: Collision Avoidance Constraint Comparison

Chapter 4

Controller Architecture

4-1 High-and Mid-level Control

In this work, we consider a hierarchical control architecture. First, each vehicle is embedded with a High Level Controller (HLC), that works as a reference generator. Here, an ideal trajectory is generated independently by every vehicle, according to simple logical rules. Then, using the reference generated by the HLC, a Mid Level Controller (MLC), computes a collision free path that is compatible with the vehicle dynamics. Specifically, different MPC controllers are tested as MLC, where the simplified bicycle dynamics (3-1), are used to model each vehicle. Finally, a Low Level Controller (LLC) are implemented to generate the actual inputs of the vehicles' actuators. Figure 4-1 shows the complete controller architecture. In this work, we only focus on the path planning algorithms show inside the dashed square.

4-1-1 High-Level-Controller: Reference Generator

To ensure all vehicles follow basic traffic rules (e.g., no right side overtaking maneuvers, no unnecessary driving in left lane), every Mid-level MPC controller is augmented with a HLC. This HLC receives perception information from sensors, e.g. available space in the road space-time. In the case of decentralized or distributed controllers, the HLC also receives communicated/estimated trajectories of other subsystems. Using this information, the lateral reference trajectory and the velocity reference can be altered (if needed) according to its surrounding. In this work, a three-lane highway is used.

Using the information received from perception sensors, e.g., LIght Detection And Ranging (LIDAR), RAdio Detection And Ranging (RADAR) and/or camera systems, or information received via a communication network, the HLC is able to change the lateral and velocity reference. Here, the lateral reference corresponds to one of the 3 road lanes and the velocity reference defines the set speed of a vehicle. Possible situations where the HLC changes the originally specified reference are:

- A vehicle is approaching a slower vehicle in the same lane and the higher adjacent lane is free $\longrightarrow$ shift lane up.

- A vehicle is driving in the middle or left lane and a lower adjacent lane is free $\longrightarrow$ shift lane down.

- A vehicle is approaching a slower vehicle in a higher lane $\longrightarrow$ lower velocity to match upper lane vehicle's velocity.

- A vehicle is approaching a slower vehicle and higher lanes are occupied $\longrightarrow$ lower velocity to match velocity of vehicle in front

The HLC uses only logic statements to determine if a reference should be altered. Algorithms (1) and (2) give an overview how the HLC is composed for a decentralized (ór distributed) controller for subsystem i. Note that the HLC is the same for the centralized as for the decentralized controllers, the only difference being that the decentralized controllers only takes its neighbors $\mathcal{N}_i$ into account.

The HLC is divided into 3 sections, i.e., Part I,II,III. The first part is responsible for shifting the lateral reference a lane higher when a vehicle is approaching a slower vehicle or when a vehicle is performing an overtaking maneuver and its initial lateral reference is actually 1 or 2 lanes lower. Note that when a lane changing maneuver is set into action, this shift in reference is hold for at least αH_p, where $\alpha \in [0, \infty)$. This is done to avoid that vehicles are continuously changing lane, portraying unrealistic behaviour. While checking availability of the adjacent lanes, the logic statements only take vehicles in the same lane, $\Delta L(x_i, x_j) = 0$, or vehicles in the higher or lower adjacent lane into account, $\Delta L(x_i, x_j) = 1 \wedge \ (x_i < x_j \vee x_i > x_j)$. Longitudinally, a safety distance $d_{covered}$ is used. This distance represent the distance covered over a prediction horizon. Part II not only shifts the lateral reference 1 or 2 lanes lower, if free, it also prevents the vehicles of returning back to their higher initial reference lane if the current lower lane is still free. The last part restricts vehicles of passing other vehicles from their right side, respectively.

4-1-2 Mid-Level-Controller: MPC

Using the reference trajectories received from the HLC, the Mid-Level-Controller (MLC) computes a collision free path, using the estimated/ communicated trajectories of other subsystems. As MLC, we chose MPC, to easily take into account the vehicles' dynamics and collision avoidance constraints. We test different MPC frameworks which are discussed in sections 4-2,4-3,4-4 and 4-5.

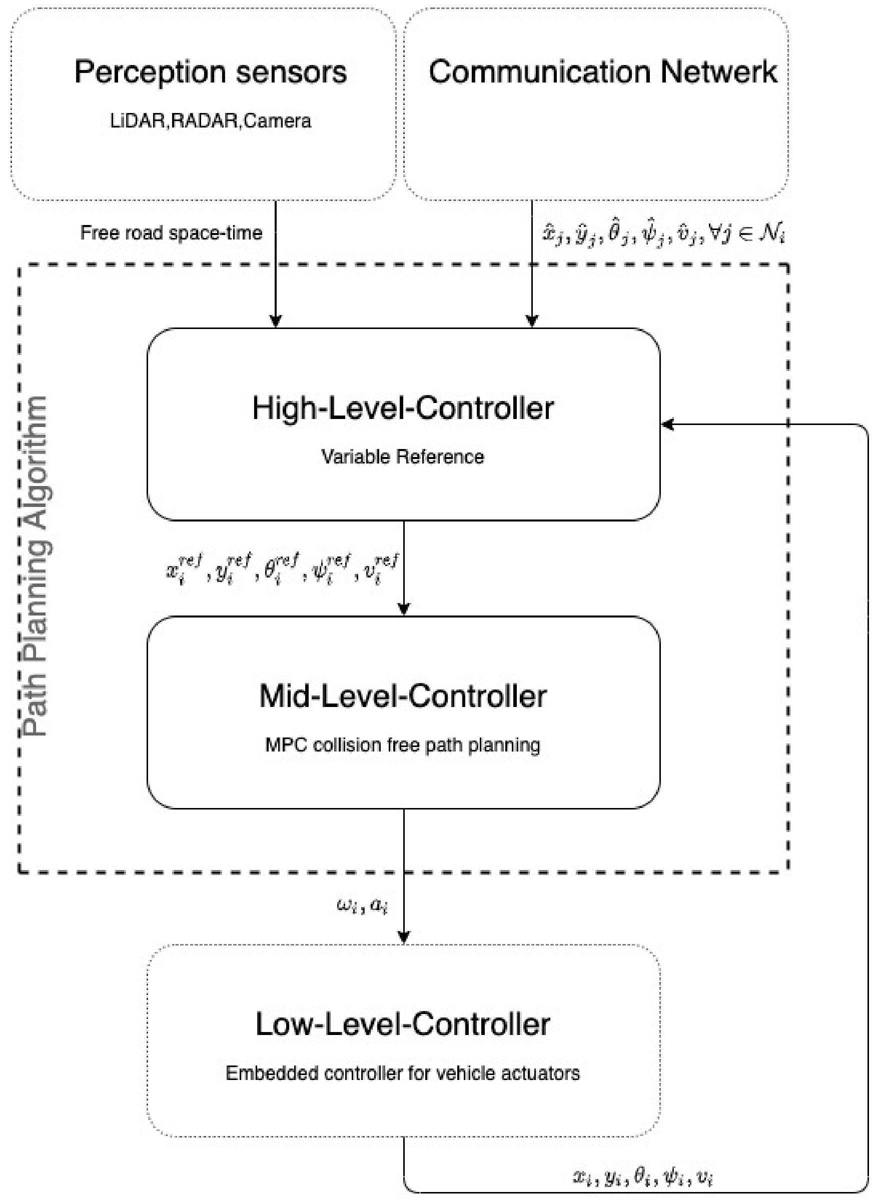

Perception sensors
LiDAR,RADAR,Camera
Communication Netwerk
Free road space-time
$\hat{x}_j, \hat{y}_j, \hat{\theta}_j, \hat{\psi}_j, \hat{v}_j, \forall j \in \mathcal{N}_i$
Path Planning Algorithm
High-Level-Controller
Variable Reference
$x_i^{ref}, y_i^{ref}, \theta_i^{ref}, \psi_i^{ref}, v_i^{ref}$
Mid-Level-Controller
MPC collision free path planning
ω_i, a_i
Low-Level-Controller
Embedded controller for vehicle actuators
$x_i, y_i, \theta_i, \psi_i, v_i$

Figure 4-1: Controller architecture

Algorithm 1: High Level Controller, variable reference

```
$d_{covered} = v_i h H_p$;
PART I, shift lane up
for $k = 1 : H_p$ do
    for $j \in \mathcal{N}_i$ do
        Hold lane shift for $\alpha H_p$:
        if $hold\_y(i) == 1$ then
            $y_{ref}(i,k) = hold\_yref(i)$;
            if $count(i) \leq \alpha H_p$ then
                $hold\_y(i) = 1$;
                $count(i) = count(i) + 1$;
            else
                $hold\_y(i) = 0$;
                $count(i) = 0$;
            end
          Shift lane up if approaching slower vehicle:
        else if $x_i < x_j$ & $v_i > v_j$ & $\Delta L(y_i, y_j) = 0$ & $|x_i - x_j| < \frac{d_{covered}}{2}$ & $lane\_occupied(above) == 0$ then
            $y_{ref}(i,k) = y_{set}(i) + lane\_width$;
            $hold\_y(i) = 1$;
            $hold\_yref(i) = y_{ref}(i,k)$;
          Shift lane up if lane lower is not yet free while overtaking:
        else if $x_i < x_j$ & $v_i > v_j$ & $\Delta L(y_i, y_j) = 1$ & $y_i > y_j$ & $|x_i - x_j| < d_{covered}$ & $lane\_occupied(below) == 1$ then
            if $lane\_occupied(current) == 1$ then
                $y_{ref}(i,k) = y(i,1) + lane\_width$;
                $hold\_y(i) = 1$;
                $hold\_yref(i) = y_{ref}(i,k)$;
            else
                $y_{ref}(i,k) = y_{set}(i) + lane\_width$;
                $hold\_y(i) = 1$;
                $hold\_yref(i) = y_{ref}(i,k)$;
            end
        else
            $y_{ref}(i,k) = y_{set}(i)$;
        end
    end
end
PART II, shift lane down if lower lane is free
for $k = 1 : H_p$ do
    for $j \in \mathcal{N}_i$ do
        if $lane\_occupied(current) == 0$ then
            $y_{ref}(i,k) = y(i,1)$;
        else if $lane\_occupied(below) == 0$ & $lane\_occupied(current) == 0$ then
            $y_{ref}(i,k) = y_{set}(i) - lane\_width$;
        else
            $y_{ref}(i,k) = y_{set}(i)$;
        end
    end
end
```

Algorithm 2: High Level Controller, variable reference

```

PART III, No overtaking on the right of other vehicles
for $k = 1 : H_p$ do
    for $j \in \mathcal{N}_i$ do
        if $hold\_v(i) == 1$ then
            $v_{ref}(i,k) = hold\_vref(i)$;
            if $count2(i) \leq \alpha H_p$ then
                $hold\_v(i) = 1$;
                $count2(i) = count2(i) + 1$;
            else
                $hold\_v(i) = 0$;
                $count2(i) = 0$;
            end
        else if $x_i < x_j$ & $y_i \leq y_j$ & $v_i \geq v_j$ then
            $v_{ref}(i,k) = min(v_j, v_{set}(i))$;
            $hold\_v(i) = 1$;
            $hold\_vref(i) = v_{ref}(i,k)$;
        else
            $v_{ref}(i,k) = v_{set}(i)$;
        end
    end
end

FUNCTION [output]=lane_occupied(case)
if $case == below$ then
    for $j \in \mathcal{N}_i$ do
        if $y_i < y_j$ & $\Delta L(y_i, y_j) \leq 2$ & $|x_i - x_j| < d_{covered}$ & $x_i \leq x_j$ then
            $output = 1$
            BREAK
        end
    end
else if $case == current$ then
    for $j \in \mathcal{N}_i$ do
        if $\Delta L(y_i, y_j) = 1$ & $|x_i - x_j| < \frac{d_{covered}}{2}$ & $x_i < x_j$ & $v_i > v_j$ then
            $output = 1$
            BREAK
        end
    end
else if $case == above$ then
    for $j \in \mathcal{N}_i$ do
        if $y_i < y_j$ & $\Delta L(y_i, y_j) = 1$ & $|x_i - x_j| < \frac{d_{covered}}{2}$ & $x_i \leq x_j$ then
            $output = 1$
            BREAK
        end
    end
else
    $output = 0$
end
```

4-2 Centralized Model Predictive Control

Traditionally, control of large, networked systems is achieved by designing local, subsystem-based controllers, which ignore the interaction between subsystems. This decentralized control often leads to poor performance when the interaction between subsystems is significant. Centralized control overcomes this limitation. In this chapter we introduce Centralized Model Predictive Control (C-MPC), that leads to a solution which is globally optimal for the overall system. Although centralized architectures are often not practical, because of computational burden or communication limits, C-MPC can serve as a benchmark for comparing and assessing different Distributed Model Predictive Control (D-MPC) approaches.

In C-MPC, one controller is responsible for controlling all the subsystems. Each vehicles' dynamics are represent by the bicycle model presented in (3-1). The dynamics of all subsystems are combined which results in the plantwide model, e.g. (4-1). Here, for simplicity, the overall system is represented as a discrete, nonlinear model of the form:

$$\mathbf{x}(t+1) = \begin{bmatrix} f_1\left(\mathbf{x}_1(t), \mathbf{u}_1(t)\right) \\ \vdots \\ f_N\left(\mathbf{x}_N(t), \mathbf{u}_N(t)\right) \end{bmatrix} \tag{4-1}$$

We define a the standard quadratic MPC cost function. The objective is to track a reference $\mathbf{r}_i(t+k)$, while avoiding collisions between vehicles. The resulting MPC problem is:

$$V^\star(\mathbf{x}(\cdot), \mathbf{u}(\cdot)) = \min_{\mathbf{u}(\cdot),\mathbf{x}(\cdot)} \sum_{i=1}^{N} \left(\sum_{k=1}^{H_p-1} \ell_{x_i}\left(\mathbf{x}_i(t+k), \mathbf{r}_i(t+k)\right) + V_{if}\left(x_i(H_p)\right) \right.$$
$$\left. \sum_{k=0}^{H_u-1} \ell_{u_i}\left(\mathbf{u}_i(t+k)\right) \right), \tag{4-2a}$$

subject to $(\forall i, j \in \mathcal{V})$:

$$\mathrm{x}_i(t+1+k) = f_i\left(\mathrm{x}_i(t+k), \mathbf{u}_i(t+k)\right), \quad k = 0, \dots, H_p, \tag{4-2b}$$
$$\mathrm{x}_i(t+k) \in \mathcal{X}_i, \quad k = 1, \dots, H_p, \tag{4-2c}$$
$$\mathrm{u}_i\left(t+H_p\right) \in \mathcal{U}_i, \quad k = 0, \dots, H_u - 1, \tag{4-2d}$$
$$c_{c.a.}^{(i,j)}\left(\mathrm{x}_i(t+k), \mathrm{x}_j(t+k)\right) \leq 0, \quad k = 1, \dots, H_p. \tag{4-2e}$$

Here the meaning of the constraints is as follows:

- $f_i : \mathbb{R}^{n_i} \times \mathbb{R}^{m_i} \to \mathbb{R}^{n_i}$ describes the dynamic prediction model of vehicle i, $i = 1, \dots, N$.
- $\mathbf{x}_i(t+k) \in \mathcal{X}_i \subseteq \mathbb{R}^{n_i}$ denotes the states of vehicle i, for $k = 1, \dots, H_p$. $\mathcal{X}_i$ is the constraint set of the states of vehicle i for
- $\mathbf{u}_i(t+k) \in \mathcal{U}_i \subseteq \mathbb{R}^{m_i}$ denotes the control inputs to vehicle i, where $m_i \in \mathbb{N}$. $\mathcal{U}_i$ is the constraint set of the inputs to vehicle i.

- $c_{c.a.}^{(i,j)} : \mathbb{R}^{n_i} \times \mathbb{R}^{n_j} \to \mathbb{R}$ represents a coupling constraint between vehicles i and j,

$$c_{c.a.}^{(i,j)}\left(\mathbf{x}_i(t+k), \mathbf{x}_j(t+k)\right) = d_{\text{safe}}^{(i,j)} - \left\|\mathbf{x}_i(t+k) - \mathbf{x}_j(t+k)\right\|. \tag{4-3}$$

  Note that the handling of (4-3) varies depending on the collision avoidance method.

The objective function is defined as:

- $\ell_{x_i} : \mathbb{R}^{n_i} \times \mathbb{R}^{n_i} \to \mathbb{R}$ is a function for the evaluation of the tracking error of vehicle i between its predicted trajectory $\mathrm{x}_i(t+k)$ and reference trajectory $\mathbf{r}_i(t+k)$ for $k = 1, \ldots, H_p - 1$. We will usually consider the following quadratic form for ℓ_{x_i}, weighted by the matrix $\mathbf{Q}_i(k)$:

$$\ell_{x_i}\left(\mathbf{x}_i(t+k), \mathbf{r}_i(t+k)\right) = \left\|\mathbf{x}_i(t+k) - \mathbf{r}_i(t+k)\right\|_{\mathbf{Q}_i}^2. \tag{4-4}$$

- $\ell_{u_i} : \mathbb{R}^{m_i} \to \mathbb{R}$ defines the input cost at each step in the prediction of vehicle i. Using $\mathbf{R}_i(k)$ as weighting matrix, ℓ_{u_i} is defined as:

$$\ell_{u_i}\left(\mathbf{u}_i(t+k)\right) = \left\|\mathbf{u}_i(t+k)\right\|_{\mathbf{R}_i}^2. \tag{4-5}$$

In our implementation we solved the OCP in (4-2) using nonlinear optimization framework CasADi [4], which implements an interior-point algorithm. Convergence of the solver is speed up by solving the OCP in a direct multiple shooting fashion and by initializing each MPC iteration, exploiting the information of the previous iteration.

4-3 Decentralized Model Predictive Control

In centralized control, all subsystems rely upon the central agent, making plantwide control difficult to coordinate and maintain. These obstacles discourage implementation of centralized control for large-scale plants. By decomposing the central problem into N subproblems and relying on communication between neighboring subsystems, each subsystem is able to compute its own control input based on the information of its neighbors. This method is adopted from [13],[32].

Definition 4.1 Neighbors $\mathcal{N}_i$
A pair of vehicles $i, j \in \mathcal{V}$ are considered neighbors if:

$$\|\mathbf{p}_i(t) - \mathbf{p}_j(t)\| \leq \alpha d_{covered}, \tag{4-6}$$

where $d_{covered} = v_i h H_p$, $\mathbf{p}_i(t) = \sqrt{\mathbf{x}_i^2(t) + \mathbf{y}_i^2(t)}$ and $\alpha \in (1,2)$. A pair of vehicles $i, j \in \mathcal{V}^N$ are considered "close" neighbors if:

$$|\mathbf{x}_i(t+k) - \mathbf{x}_j(t+k)| \leq d_{covered} \;\&\; |\mathbf{y}_i(t+k) - \mathbf{y}_j(t+k)| < \mu. \tag{4-7}$$

where μ denotes a single road-width.

In literature, this method is also referred to as (Cooperative) Distributed (Nonlinear) MPC ([32],[13],[12],[51]), but since this method does not rely on distributed optimization, and to avoid confusion with other methods, here it is named Decentralized MPC (Dec. MPC).

All subsystems solve their local subproblem synchronously. Hence, at every time instance

t the estimated trajectories of all neighbors $j \in \mathcal{N}_i, \forall i \in \mathcal{V}$ should be known. These are assumed to be the optimal trajectories obtained from the previous MPC iteration. Since prior to $t = 0$ no OCP has been solved, these have to be initialized. This starts by initializing the estimated control $\hat{u}_i$.

Definition 4.2 Initialization of estimated control
At every time instance τ in interval $[t, t + H_p]$, the estimated control for each vehicle is defined as follows:

$$\begin{cases} \hat{u}_i\left(\tau, t\right) = u_i^*\left(\tau; (t-1)\right) \;\text{ if } \tau \in [t, (t-1) + H_p) \\ \hat{u}_i\left(\tau, t\right) = 0, \;\text{ if } \mathbf{x}_i\left(t\right) = \mathbf{r}_i(Hp) \;\; \textbf{or} \text{ if } \tau \in [(t-1) + H_p, t + H_p] \end{cases} \tag{4-8}$$

$u_i^*\left(\tau; (t-1)\right)$ denotes the optimal solution solved at the previous MPC iteration with initial state $\mathbf{x}_i\left((t-1)\right)$. Since prior to t_0 no OCP has been solved, an initialization method is defined for the time interval $[(t_0 - 1), (t_0 - 1) + H_p]$. Algorithm 3 is introduced:

Algorithm 3: OCP Initialization

```
1 if $\tau \in (t_0 - 1)$ then
2     solve (4-9) with initial state $\mathbf{x}_i\left(t_0 - 1)\right)$, $\mathbf{x}_j\left(t_0 - 1\right)$ and $\hat{u}_j\left(\tau; x_j(t_0 - 1)\right) = 0$
3     for all $\tau \in [(t_0 - 1), (t_0 - 1) + H_p]$ and $\mathcal{K} = +\infty$.
4
```

The optimal control obtained by Algorithm 3, with the mentioned conditions, is the estimated control for the time interval $[t_0, t_0 + H_p]$.

Please note, $\mathcal{K} = +\infty$, implies that the compatibility constraint is not important prior to t_0. The state and control trajectories obtained at $(t_0 - 1)$ over interval $\tau \in [(t_0 - 1),$ $(t_0 - 1) + H_p]$ are denoted by $\mathbf{x}_i^*\left(\tau; \mathbf{x}_i\left((t_0 - 1)\right)\right)$ and $u_i^*\left(\tau; \mathbf{x}_i\left((t_0 - 1)\right)\right)$. This optimal control is applied to the i^{th} vehicle over $[(t_0 - 1), t_0]$.

Again, using nonlinear bicycle model (3-1) to represent the vehicles' dynamics, the Decentralized Nonlinear MPC for vehicle collision avoidance problem for subsystem $i \in \mathcal{V}$ at time t is formulated as:

$$V_i^\star(\mathbf{x}_i(0), \hat{\mathbf{x}}_j(\cdot), \mathbf{u}_i(\cdot)) = \min_{\mathbf{x}(\cdot), \mathbf{u}(\cdot)} \left(\sum_{k=1}^{H_p - 1} \ell_{x_i}\left(\mathbf{x}_i(t+k), \mathbf{r}_i(t+k)\right) + V_{if}\left(x_i(H_p)\right), \right.$$

$$\left. + \sum_{k=0}^{H_u - 1} \ell_{u_i}\left(\mathbf{u}_i(t+k)\right) + \sum_{j \in \mathcal{N}_i, j \neq i} \sum_{k=1}^{H_p - 1} \gamma \left\| L_{col}\left(\mu_{ij}\left(\mathbf{x}_i, \hat{\mathbf{x}}_j\right)\right) \right\|^2 \right) \tag{4-9a}$$

subject to $(\forall i, j \in \mathcal{N}_i)$:

$$\mathbf{x}_i(t+1+k) = f_i\left(\hat{\mathbf{x}}_i(t+k), \mathbf{u}_i(t+k)\right), \quad k = 0, \ldots, H_p, \tag{4-9b}$$

$$\hat{\mathbf{x}}_j(t+1+k) = f_j\left(\mathbf{x}_j(t+k), \hat{\mathbf{u}}_j(t+k)\right), \quad k = 0, \ldots, H_p, \tag{4-9c}$$

$$\mathbf{x}_i(t+k) \in \mathcal{X}_i, \quad k = 1, \ldots, H_p, \tag{4-9d}$$

$$\mathbf{u}_i\left(t + H_p\right) \in \mathcal{U}_i, \quad k = 0, \ldots, H_u - 1, \tag{4-9e}$$

$$\left\| \mathbf{x}_i\left(t+k\right) - \hat{\mathbf{x}}_i\left(t+k\right) \right\| \leq h^2 \mathcal{K} \tag{4-9f}$$

The first functions and variables in (4-9) are defined equivalently to (4-2a). The last term in (4-15a) denotes the collision avoidance soft constraint. Below all remaining functions and constraints are explained:

- $L_{col}\left(\mu_{ij}\left(x_i, \hat{x}_j\right)\right) : \mathbb{R}^{n_i} \times \mathbb{R}^{n_j \to \mathbb{R}}$ describes the cost associated with the collision avoidance of vehicle i with vehicle j $\in \mathcal{N}_i$:

$$\sum_{j \in \mathcal{N}, j \neq i} \sum_{k=1}^{H_p-1} \gamma \left\| L_{col}\left(\mu_{ij}(t+k)\right)\right\|^2 \text{ where, } L_{col}(t) = \begin{cases} \frac{1}{\|x_i(t)-x_j(t)\|-2D_i} \text{ if } j \in \mathcal{N}_i, \\ \\ 0 \text{ if } j \notin \mathcal{N}_i. \end{cases} \tag{4-10}$$

  here, $2D_i$ is defined similar to (3-5), γ denotes a weighting term.

- $f_j : \mathbb{R}^{n_j} \times \mathbb{R}^{m_j} \to \mathbb{R}^{n_j}$ describes the dynamic prediction model of neighboring vehicle $j \in \mathcal{N}_i$.

- $\hat{\mathrm{x}}_j(t+k) : \mathbb{R}^{n_j}$ denotes the estimated/communicated states of vehicle j from the previous MPC iteration, for $k = 1, \ldots, H_p$.

- $\|x_i\left(t+k\right) - \hat{x}_i(t+k)\| \leq h^2\mathcal{K} : \mathbb{R}^{n_i} \times \mathbb{R}^{n_j} \to \mathbb{R}$ represents the compatibility constraint that ensures feasibility of each computed control input. Here, $h^2\mathcal{K}$ denotes the multiplication of the squared sample time with a constant $\mathcal{K} \in (0, \infty)$ and $h^2\mathcal{K} \approx 0$.

Similar to C-MPC, the OCP in (4-9) is solved using the nonlinear optimization framework CasADi.

4-4 Cooperative Distributed Model Predictive Control

In Dec. MPC, each subsystem communicates ones every MPC iteration to its neighbors, (2-2). The newly computed trajectories are constrained to not deviate too much from the previously computed trajectory by a compatibility constraint. This ensures feasibility, but comes at the cost of optimality. Cooperative Distributed MPC (Coop. D-MPC) overcomes this limitation, set by the compatibility constraint and simple communication scheme, by communicating several times every MPC iteration and generally using distributed optimization to solve the OCP.

In [43, Section 3-5] an example of linear time-invariant Coop. D-MPC is given for networked agents where exponential closed-loop stability is guaranteed. This method is also used in [34], [42] and [26]. When ran for sufficient suboptimal iterations, i.e., $p \longrightarrow \infty$, the algorithm converges to the plantwide Pareto-optimal solution is achieved by explicitly modelling the effects of inputs of subsystem j on the states of subsystem $i, \forall i, j \in \mathcal{V}$. [44] gives a similar approach for nonlinear plants, where an extra distributed gradient algorithm is used to ensure a decrease in the non-convex objective function. Unfortunately, both distributed algorithms only take simple input bounding constraints into account.
In [30] a Coop. D-MPC framework is shown based on game theory. Here, all agents solve their individual cost function and cooperate to find a global cost decreasing solution. Feasible Cooperation MPC is introduced in [45], this method is also based on game theory and has a local cost function that is replaced by a cost function that measures the system-wide impact of the local control inputs. This is done with the purpose of avoiding competition and to increase the cooperation among subsystems in a D-MPC scheme.

In our case, the collision avoidance constraints make the OCP non-convex and time-varying. As a result, above mentioned algorithms can not be directly implemented. Other distributed algorithms such as, primal and dual decomposition and Alternating Direction

Method of Multipliers (ADMM) have as significant drawback that the constraints are only satisfied asymptotically. In [53] and [47], ADMM is applied to a vehicle collision avoidance problem. In the first, the algorithm is required to run "tens of iterations" before convergence, making it unsuitable for fast driving vehicles which need quick decision making. In the latter, the ADMM algorithm is only iterated a single time. By dividing the free road space-time between holonomic vehicles, collision avoidance is ensured at the cost of having a conservative feasible space for each vehicle. In this thesis, an adaptation to [43] is used. Sub-optimal cooperative distributed MPC is implemented. Here, a coordinator is not necessary and suboptimal input trajectories can be injected into the subsystems, allowing the plant to stop the iterative process before reaching the stopping criteria. This property increases the flexibility of distributed MPC, and the plantwide control strategy can be treated as suboptimal MPC.

Sub-Optimal Cooperative Control for Vehicle Collision Avoidance

For any initial state $x(0)$, the suboptimal MPC is initialized with a feasible input trajectory $\tilde{\mathbf{u}}(0)$. For subsequent sample times, $\tilde{\mathbf{u}}(t+1)$ is denoted as the *warm start*, i.e., a feasible input sequence for $x(t+1)$ used to initialize the suboptimal MPC algorithm. Each sample time, $\tilde{\mathbf{u}}(t+1)$ is obtained by shifting the previous warm start one step forward. The first (applied) control input is omitted and at the end the sequence is augmented with a zero:

$$\tilde{\mathbf{u}}_i(t+1) = \{u_i(1), u_i(2), \dots, u_i\left(H_p - 1\right), 0\}. \tag{4-11}$$

Each local subsystem performs p iterations of a feasible path algorithm and computes $\mathbf{u}$ such that the solution of the OCP is improved w.r.t their own local objective function. At each sample time, the first input in the (suboptimal) trajectory is applied, $u = u(0)$. The state is updated by the state evolution equation $x(t+1) = f(x(t), u(t))$.

When applying the algorithm proposed in [43, Section 3-5] to a vehicle collision avoidance problem, some adaptations inherent to the problem set up occur. First, it is noted that the vehicle states are uncoupled. This can be explained by considering a possibly non-minimal centralized model between subsystem $i, j \in \mathcal{V}$.

$$x(t+1) = Ax(t) + \sum_{j \in \mathcal{V}} B_j u_j, \quad y_i(t) = C_i x(t) \tag{4-12}$$

Then, for each input/output pair (u_j, y_i) the triple (A, B_j, C_i) is transformed into its Kalman canonical form, [35]:

$$\begin{aligned} \begin{bmatrix} z_{ij}^{oc} \\ z_{ij}^{oc} \\ z_{ij}^{o} \\ z_{ij}^{oc} \end{bmatrix} &= \begin{bmatrix} A_{ij}^{oc} & 0 & A_{ij}^{oc\bar{c}} & 0 \\ A_{ij}^{\bar{o}oc} & A_{ij}^{\bar{o}c} & A_{ij}^{\bar{o}co\bar{c}} & A_{ij}^{\bar{o}c\bar{c}} \\ 0 & 0 & A_{ij}^{o\bar{c}} & 0 \\ 0 & 0 & A_{ij}^{\bar{c}c} & A_{ij}^{\bar{o}\bar{c}} \end{bmatrix} \begin{bmatrix} z_{ij}^{oc} \\ z_{ij}^{oc} \\ z_{ij}^{o} \\ z_{ij}^{oc} \end{bmatrix} + \begin{bmatrix} B_{ij}^{oc} \\ B_{ij}^{oc} \\ 0 \\ 0 \end{bmatrix} u_j, \\ y_{ij} &= \begin{bmatrix} C_{ij}^{oc} & 0 & C_{ij}^{o\bar{c}} & 0 \end{bmatrix} \begin{bmatrix} z_{ij}^{oc} \\ z_{ij}^{oc} \\ z_{ij}^{oc} \\ z_{ij}^{oc} \end{bmatrix}, \quad y_i = \textstyle\sum_{j \in \mathcal{V}} y_{ij}, \end{aligned} \tag{4-13}$$

here, z_{ij}^{oc} denotes the modes of A that are both observable by y_i and controllable by u_j. The distributed state space model that denotes the effects of subsystems $j \in \mathcal{V}$ to subsystem i

becomes:

$$A_{ij} \leftarrow A_{ij}^{oc}, \quad B_{ij} \leftarrow B_{ij}^{oc}, \quad C_{ij} \leftarrow C_{ij}^{oc}, \quad x_{ij} \leftarrow z_{ij}^{oc}. \tag{4-14}$$

By definition, no input of one vehicle can control the states of another vehicle, hence z_{ij}^{oc} is empty and the vehicle states are fully decoupled. The resulting vehicle models only consist of local states and inputs, equivalent to (3-2).

The second adaption is the feasibility of the computed suboptimal trajectories. Due to the collision avoidance constraints, the algorithm either needs to run for a fixed number of iterations p, or a compatibility constraint, similar to (4-9f), should be applied to ensure feasibility of the suboptimal trajectories when the algorithm is stopped prior to the stopping criteria. This latter option is applied to preserve the benefits of suboptimal MPC.

Taking these adaptations into account, the Coop. Distr. MPC algorithm for vehicle collision avoidance for subsystem $i \in \mathcal{V}$ at time t is formulated as:

$$V_i^\star(\mathbf{x}_i(0), \mathbf{u}_i(\cdot)) = \min_{\mathbf{x}(\cdot),\mathbf{u}(\cdot)} \left(\sum_{k=1}^{H_p-1} \ell_{x_i}\left(\mathbf{x}_i(t+k), \mathbf{r}_i(t+k)\right) + V_{if}\left(x_i(H_p)\right) \right.$$
$$\sum_{k=0}^{H_u-1} \ell_{u_i}\left(\mathbf{u}_i(t+k)\right) \tag{4-15a}$$

subject to ($\forall i, j \in \mathcal{N}_i$):

$$\mathbf{x}_i(t+1+k) = f\left(\mathbf{x}_i(t+k), \mathbf{u}_i(t+k)\right), \quad k = 0, \ldots, H_p, \tag{4-15b}$$
$$\hat{\mathbf{x}}_j(t+1+k) = f_j\left(\hat{\mathbf{x}}_j(t+k), \hat{\mathbf{u}}_j(t+k)\right), \quad k = 0, \ldots, H_p, \tag{4-15c}$$
$$\mathbf{x}_i(t+k) \in \mathcal{X}_i, \quad k = 1, \ldots, H_p, \tag{4-15d}$$
$$\mathbf{u}_i\left(t+k\right) \in \mathcal{U}_i, \quad k = 0, \ldots, H_u, \tag{4-15e}$$
$$c_c^{(i,j)}\left(\mathrm{x}_i(t+k), \mathrm{x}_j(t+k)\right) \leq 0, \quad k = 1, \ldots, H_p, \quad j > i, \tag{4-15f}$$
$$\boldsymbol{v}_i = \boldsymbol{v}_i^p, \quad \forall i \in \mathcal{V}, \tag{4-15g}$$
$$\left\|\mathbf{x}_i\left(t+k\right) - \hat{\mathbf{x}}_i\left(t+k\right)\right\| \leq h^2 \mathcal{K}. \tag{4-15h}$$

Agent i is trying to follow its reference trajectory $\mathbf{r}_i$ with a minimum change in its inputs u_i. The functions and variables in constraint (4-15b) - (4-15f) are defined using the linear vehicle model and collision avoidance constraints in section 3-2-2 and 3-3. Constraint (4-15h) is defined equivalent to (4-9f). The plantwide cooperative control iteration, performed in parallel by every subsystem, is given by:

$$\mathbf{u}_i^{p+1} = \sum_{j \in \mathcal{V}^i} w_j \sigma\left(\boldsymbol{v}_i^\star, \boldsymbol{v}_j^p\right), \quad \sum_{j \in \mathcal{V}^i} w_j = 1, \quad \forall w_j > 0, \tag{4-16}$$

where $\sigma\left(u_i^\star, u_j^p\right)$ denotes the convex combination step for subsystem i. This distributed optimization is of the Gauss–Jacobi type. Note that for simplicity the dependence of $u_i^\star$ on u_j^p is omitted. At the last iterate $\bar{p}$, we set $\mathbf{u} \leftarrow (\mathbf{v}_1^p, \mathbf{v}_2^p)$ and apply $u(0)$ to the plant.

The following properties follow.

Lemma 4.1 Feasibility
Given a feasible initial guess $\tilde{\mathbf{u}}_i(0)$*, the iterates satisfy:*

$$\left(\boldsymbol{v}_i^p, \boldsymbol{v}_j^p\right) \in \mathcal{U}_i^N \times \mathcal{U}_j^N, \quad \boldsymbol{x}_i^p \in \mathcal{X}_i \cap \mathcal{C} \quad \forall i \in \mathcal{V}, \tag{4-17}$$

where $\mathcal{C}$ *denotes the constrained set from* (4-15f)*. If ran for a sufficiently large, fixed number* p*, or when using compatibility constraint* (4-15h) *and the iterative algorithm is stopped prematurely at* $p \geq 1$*, all (sub-optimal) trajectories are feasible.*

Lemma 4.2 Convergence
The plantwide cost,

$$V(\mathbf{x}(0), \boldsymbol{v}^p) = \sum_{\forall i \in \mathcal{V}} V_i(\mathbf{x}_i(0), \boldsymbol{v}_i^p) \tag{4-18}$$

is non-increasing and converges as $p \longrightarrow \infty$*.*

Observation 4.1 Optimality
As $p \longrightarrow \infty$*, the plantwide cost* $V(\mathbf{x}(0), \boldsymbol{v}^p)$ *converges to a constant value, but convergence to the Pareto optimal solution can not be guaranteed, due to the non-convexity of* (4-15)*.*

Proof of Lemma 4.1, 4.2 and Observation 4.1 is given in Appendix B.

4-5 Non-cooperative Distributed Model Predictive Control

In a road traffic environment, traffic participants might be selfish decision makers. Typically, each driver behaves according to its own individual interests, while sharing the road space-time with the other drivers. This makes game theory an interesting framework to model and cope with non-cooperative behaviors in automated driving. We formulate a Non-cooperative Distributed MPC (Noncoop. D-MPC) framework based on a Generalized Potential Game. To find a solution to the game we adopt a method from [17, Section VI].

A Generalized Potential Game for Vehicle Collision Avoidance

We will consider the mixed integer MPC framework where each agent i solves its own OCP

$$V_i^\star(\mathbf{x}_i(0), \mathbf{u}_i(\cdot)) = \min_{\mathbf{x}(\cdot), \mathbf{u}(\cdot)} \left(\sum_{k=1}^{H_p - 1} \ell_{x_i}\left(\mathbf{x}_i(t+k), \mathbf{r}_i(t+k)\right) + V_{if}\left(x_i(H_p)\right) \right.$$
$$\sum_{k=0}^{H_u - 1} \ell_{u_i}\left(\mathbf{u}_i(t+k)\right) \tag{4-19a}$$

subject to ($\forall i, j \in \mathcal{N}_i$):

$$\mathbf{x}_i(t+1+k) = f\left(\mathbf{x}_i(t+k), \mathbf{u}_i(t+k)\right), \quad k = 0, \ldots, H_p, \tag{4-19b}$$

$$\hat{\mathbf{x}}_j(t+1+k) = f_j\left(\hat{\mathbf{x}}_j(t+k), \hat{\mathbf{u}}_j(t+k)\right), \quad k = 0, \ldots, H_p, \tag{4-19c}$$
$$\mathbf{x}_i(t+k) \in \mathcal{X}_i, \quad k = 1, \ldots, H_p, \tag{4-19d}$$
$$\mathbf{u}_i\left(t+k\right) \in \mathcal{U}_i, \quad k = 0, \ldots, H_u, \tag{4-19e}$$
$$c_c^{(i,j)}\left(\mathrm{x}_i(t+k), \mathrm{x}_j(t+k)\right) \leq 0, \quad k = 1, \ldots, H_p, j > i. \tag{4-19f}$$

By computing a solution to (4-19), each agent $i \in \mathcal{V}$ can be driven towards its set point in the road space-time over the horizon H_p. However, each control strategy is computed by assuming the strategies of the neighbors are given. If even a single of these strategies change, the computed strategies may not be optimal anymore or even unsafe. To achieve this, (4-19) is formulated as a Generalized Potential Game (Definition 2.6).

Similar to [17], the feasible set of each player is defined as, $\mathcal{X}_i\left(\boldsymbol{x}_{-i}\right) := \{\boldsymbol{x}_i \in \mathbb{R}^{n_i} | A\left(x_i, x_{-i}\right) \leq b\}$ and $\mathcal{X} := \{x \in \mathbb{R}^n | Ax \leq b\}$. By noticing that each $V_i\left(x_i\right)$ depends only on the local variable x_i, we introduce the function $P : \mathbb{R}^n \to \mathbb{R}$, defined as $P(x) := \sum_{i\in\mathcal{V}} V_i\left(x_i\right)$, that satisfies for all $i \in \mathcal{V}$, for all x_{-i}, and for all $x_i, y_i \in \mathcal{X}_i\left(x_{-i}\right)$,

$$P\left(\mathbf{x}_i, \mathbf{x}_{-i}\right) - P\left(\mathbf{y}_i, \mathbf{x}_{-i}\right) = V_i\left(\mathbf{x}_i\right) - V_i\left(\mathbf{y}_i\right). \tag{4-20}$$

According to Definition 2.6, P is an exact potential function for the proposed automated driving coordination game. The best response mapping from player i, given the strategies of its neighbours $\mathbf{x}_{-i}$:

$$x_i^\star\left(x_{-i}\right) \in \left\{ \begin{array}{l} \operatorname{argmin}_{x_i} V_i\left(x_i\right) \\ \text{ s.t. } \left(x_i, x_{-i}\right) \in \mathcal{X}. \end{array} \right. \tag{4-21}$$

Using Definition 2.4, with $\epsilon > 0$, $x^\star \in \mathcal{X}$ is an ϵ-Nash Equilibrium (ϵ-NE) of (4-21) if, for all $i \in \mathcal{V}$,

$$V_i\left(x_i^*\right) \leq \inf_{x_i \in \mathcal{X}_i\left(x_{-i}^*\right)} V_i\left(x_i\right) + \varepsilon. \tag{4-22}$$

An ε-NE is a set of strategies that, up to an accuracy ϵ, are individually optimal, given the constraints.

We adopt a best-response based algorithm for computing an ε-NE via an iterative procedure given in [17, Section VI]. We refer to a *temporal* step as a sample step of the decision variables over H_p and to *algorithmic* step as an iterative step to compute a solution of the game. The vector of decision variables at the k-th algorithmic step of player i and of its neighbors, respectively, are denoted as $\boldsymbol{x}_i(k)$ and $\boldsymbol{x}_{-i}(k), k \in H_p$. The i-th vector of decision variables at time t, computed at the algorithmic step k, is denoted as $\boldsymbol{x}_i(t|k)$. Lastly, a cost variation is introduced $\Delta V_i(k) := V_i\left(x_i(k)\right) - V_i\left(x_i^*(k)\right)$, with $x_i^*(k) \in \boldsymbol{x}_i^\star\left(\boldsymbol{x}_{-i}(k)\right)$.

An inter-vehicle ordering relation at time t, i.e., $\succ_t$, is defined to formulate the sequence in which each OCP is solved. Given any pair of vehicles $(i,j) \in \mathcal{V}^2$, we say that j has lower order than i at time t, namely $j \prec_t i$ when:

$$\left\|\mathbf{p}_j\right\|_2 > \left\|\mathbf{p}_i\right\|_2, \tag{4-23}$$

where $\|\cdot\|_2$ denotes the Cartesian distance with respect to the origin of the global world frame, e.g., the vehicle that has progressed the most in the road space-time. The subscripts will refer to vehicles which follow the ordering in $\mathcal{O}_t$. For any vehicle $i, x_{-i}(k)$ is obtained by stacking $x_j(k+1)$ for all $j \prec_t i$ and $x_j(k)$ for $j \succ_t$ i.

We investigate a closed-loop implementation of a Gauss-Seidel type algorithm, equivalent to [17, Section VI]. The vehicles implement only the first temporal step of the equilibrium solution, before a new one is calculated. This limits the amount of communication to $\boldsymbol{x}_i(t+1|k), k \in \mathbb{N}$, since the players are only interested in the "next" action of its neighbours. In Algorithm 4 each player is assumed to have estimates for the remaining parts of the strategies, i.e., $\hat{\boldsymbol{x}}(h|k),\ \forall h \in \mathcal{H}_t := \{t+2, \dots, t+T\},\ k \in \mathbb{N}$. Note that this requires additional linear constraints and restricts the feasible set to $\hat{\mathcal{X}}_t := \{x \in \mathbb{R}^n | A_t x \leq b_t\} \subseteq \mathcal{X}$.

Algorithm 4: Gauss-Seidel Method (Closed Loop)

```
1 Choose a feasible point x(0) $\in \hat{\mathcal{X}}_t$, set $k := 0$
2 while x(k) is not an $\varepsilon -$ NE do
3     Broadcast $x_i(t+1|k)$ to $\mathcal{N}_i, \forall i \in \mathcal{O}_t$
4     for all $i \in \mathcal{O}_t$ do
5         $\boldsymbol{x}_i(k+1) := \begin{cases} \boldsymbol{x}_i^*(k) & \text{if } \Delta V_i(k) \geq \varepsilon \\ \boldsymbol{x}_i(k) & \text{otherwise} \end{cases}$
          Broadcast $\boldsymbol{x}_i(t+1|k+1)$ to all $j \succ_t i$
6     end
7     Set k:=k+1
8 end
```

Since each best-response mapping in (4-21) is based on estimated strategies of their neighbours, feasibility and existence of an ε-NE over the full horizon is not self-evident. According to [17, Lemma 2] and [17, Proposition 4], when assuming a non-empty set $\hat{\mathcal{X}}_t$ for all t, and let $\mathcal{O}_t \subseteq \mathcal{I}$ be an ordered set of vehicles and $\epsilon > 0$, Algorithm 4 provides a feasible collective strategy $(x_i(t+1|k), x_{-i}(t+1|k))$ and computes an $\varepsilon -$ NE, $\overline{\boldsymbol{x}} \in \hat{\mathcal{X}}_t$ of the GPG.

Chapter 5

Controller Assessment

In this chapter we present the testing scenarios and the assessment criteria for our controllers. The two main evaluation criteria are the plantwide feasibility of the controller and its effectiveness.

5-1 Testing Scenario

5-1-1 Scenario I: Unrestricted Highway

As first scenario we consider a standard highway scenario where N vehicles are driving in parallel, in front or behind each other. All vehicles are trying to track their own velocity reference.

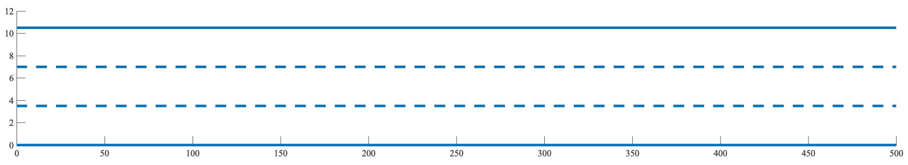


Figure 5-1: Unrestricted Highway testing scenario

5-1-2 Scenario II: Roadworks Obstructed Highway

In the unrestricted scenario, no obstructions are present. To increase heterogeneity, we assume that the 3-laned highway is restricted to only a single lane due to road works. All vehicles are warned for these roadworks by road signs. These signs tell the vehicles to slow down to 70 km/h and eventually merge into a single lane. This results in a more challenging scenario, making the differences in the controllers possibly more noticeable.

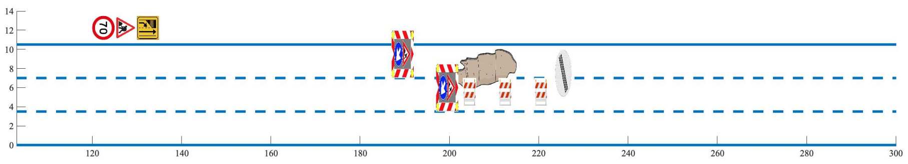


Figure 5-2: Obstructed Highway testing scenario

5-2 Evaluation Criteria

The evaluation of the MPC strategies are done according to two criteria: feasibility of the control strategy and effectiveness of the controller. The former focuses on the computational time and the robustness of a controller, the latter focuses on the trajectory quality.

5-2-1 Feasibility

The most important aspect in control for autonomous vehicles is the controller's feasibility. If the controller is not able to find the newly computed control input within the fixed sampling time interval h, safety can be compromised. Also robustness to hardware failure is an important aspect which needs to be taken into account.

Computation Time

An important aspect to take into account in the MPC implementation, is the sampling time h. A prerequisite is that a feasible control input is found within a single MPC iteration, i.e., within the sampling time. Obviously, the scale of the OCP plays an important role here, computation time scales exponentially with an increasing numbers of subsystems. For embedded control, the following inequality should hold: $T_{sol} \leq h$. Here, T_{sol} is the time it takes to compute a single prediction and control horizon and h the sampling time.

The computation time is evaluated by computing the mean and maximum values over all time steps of a simulation.

Robustness to Hardware Failure

Possibly the second most important aspect in embedded control for autonomous vehicles is robustness. This can be achieved by building-in safety systems, but it can also be achieved in the design of the control framework. Centralized control is able to account for plantwide interactions but, organizationally, all subsystems rely upon the central agent, making plantwide control not robust to hardware failure. If the central controller fails, or communication with the central controller fails, all vehicles are essentially out of control. This deters implementation of plantwide control. Decentralized control overcomes this limitation by decomposing the plantwide problem into subproblems and allowing each vehicle to compute its own control input, depending on the communicated/estimated states of the other vehicles. In this case, if communication fails, only a single vehicle will be out of control and safety could be assured by an emergency controller. Note that the design of emergency controllers is out of the scope of this research.

Robustness to hardware failure is not quantitatively evaluated, it is simply taken into account in the assessment of each controller.

5-2-2 Trajectory Quality

The quality of the vehicle trajectories are evaluated by computing different objectives. The first criteria is the overall plantwide effectiveness, i.e., the summed cost of all local

subsystems with respect to their reference. It is expected that, the more vehicles cooperate with each other, the lower the overall cost is.
The second criteria is the individual satisfaction of each subsystem. Plantwide effectiveness (and control) can come at the cost of some subsystems having to sacrifice their own objective to increase plantwide efficiency. It is expected that the non-cooperative distributed controller, based on game theory, has a higher individual satisfaction since this controller finds a(n) (epsilon)-Nash Equilibrium. Here, all subsystems agree upon their actions based on the actions of the others.
The final criteria represents the progress each vehicle has made. Different then above mentioned criteria, this is not with respect to some reference, but purely the cost associated with the progress made in the simulation, i.e., the summed progress of all paths.

For all criteria, the following holds. The standard weighting matrices Q_i and R_i in the OCP may vary between controllers, e.g., because of the presence of potential functions in the objective function, the state weights possibly need to be re-tuned accordingly. Because of this, the computed objective values between controllers can not be compared. Therefore, the applied trajectories are substituted into different assessment objective functions, (5-1). Here, the weighting matrices Q_i^{result} and R_i^{result} are equivalently defined for all controllers and all vehicles. Below, the different objective functions are listed.

- **Overall effectiveness:**

$$V(\mathbf{x})_{\sum_{\forall i \in \mathcal{V}}} = \frac{\sum_{i=1}^{N} \left(\sum_{k=1}^{H_p} \|\mathbf{x}_i(t+k) - \mathbf{r}_i(t+k)\|^2_{\mathbf{Q}^{result}} \right)}{\bar{V}_{C-MPC}} \tag{5-1a}$$

  where,

$$Q_i^{result} = blkdiag(zeros(3), \psi_{weight}, \frac{v_{weight}}{v_i^{set}}). \tag{5-1b}$$

- **Individual satisfaction:**

$$V(\mathbf{x})_i = \frac{\sum_{k=1}^{H_p} \|\mathbf{x}_i(t+k) - \mathbf{r}_i(t+k)\|^2_{\mathbf{Q}^{result}}}{\bar{V}_{C-MPC}} \tag{5-2}$$

- **Progress of all paths:**

$$V(\mathbf{x})_{path} = \frac{\sum_{i=1}^{N} \left(\sum_{k=1}^{H_p} (\mathbf{x}_i(t+k) - \mathbf{x}_{max}) + \Delta \mathbf{y}_i(t+k) \right)}{\bar{V}_{C-MPC}} \tag{5-3}$$

The first bullet point represents the overall effectiveness, which is measured by evaluating the rate at which vehicles meet their velocity reference and the possible steering angles needed to overtake vehicles. This summed cost is normalized with the mean summed cost of C-MPC, i.e., $\bar{V}_{C-MPC}$.

The individual satisfaction is measured by evaluating the variance of the cost of each subsystem with respect to their reference, again normalized with the mean summed cost of C-MPC. Since all values are normalized with the mean summed cost of C-MPC, the combination of an overall summed cost equivalent to C-MPC, and a smaller variance in this cost, would correspond to a more individually optimal solution. The opposite is true for larger variances.

The cost of the progress made in the path is measured by adding the difference in progress in longitudinal direction with respect to the maximum longitudinal distance, x_{max}, to the sum of the difference in lateral direction between every MPC iteration. Again, normalized by the mean summed cost of C-MPC.

Chapter 6

Simulation Results Unrestricted Highway Scenario

This chapter presents and discusses the simulation results of the MPC path planning algorithms according to the criteria discussed in chapter 5, in the Unrestricted Highway scenario.

For simplicity, all vehicles are assumed to be identical in size. The figures show the resulting trajectories at the following time steps:

- The initial time step to show the starting positions of the vehicles.
- Time steps in which it can be seen that the collisions are avoided.
- Time steps where it can be seen that the reference changes.

The predicted trajectories of all vehicles are shown using circles, the actual vehicle state is shown with a colored circle in the same corresponding color. The steering angle and velocity (and corresponding reference) are pictured below the vehicles trajectories. A final plot is shown which depicts the completed trajectories of each vehicle.

The discussed testing scenario are shown for $N = 12$ subsystems in the Unrestricted Highway scenario and the Roadworks Obstructed Highway. First, the globally optimal centralized solutions are shown for different time steps in the simulation, followed by a final overview of the traveled path. For decentralized/distributed controllers, the traveled paths are often really similar. Hence, to save space only the final overview of the traveled path is shown. After each decentralized/distributed controller, the mean objective values normalized w.r.t. the centralized controller are shown, together with the computational times and communication burden of those controllers.

All simulations are performed in MATLAB [31]. The optimization problems are solved using different optimization software libraries. C-MPC and Dec. MPC are defined as nonlinear MPC controllers, which are solved using CasADi, [4], which is a state of the art software framework for nonlinear optimization. Coop. D-MPC and Non-coop. D-MPC use mixed integer variables to convexify the constraints, making the problems a Mixed Integer Quadratic Programming (MIQP) problem. These are solved using the Gurobi MIQP solver, [24]. All simulations are conducted on a computer equipped with an Intel Core i7, 2,7 GHz Quad-Core processor and 16 GB RAM memory.

6-1 Centralized MPC

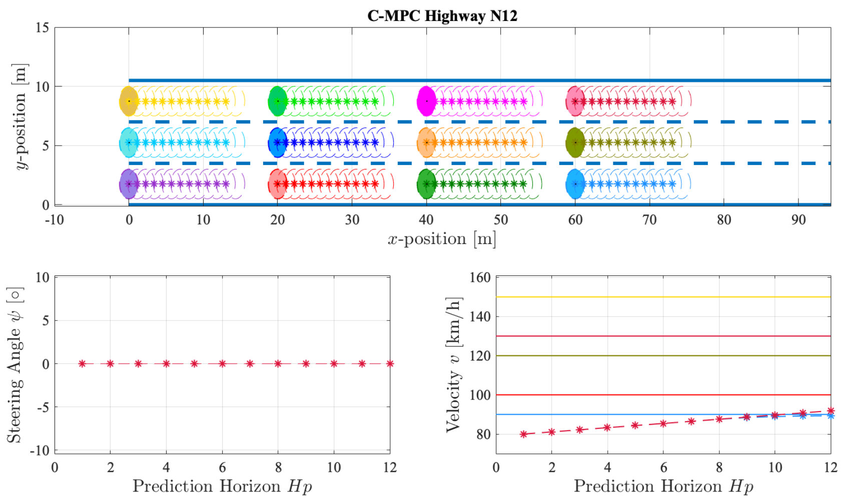


Figure 6-1: MPC iteration = 1, Initial conditions

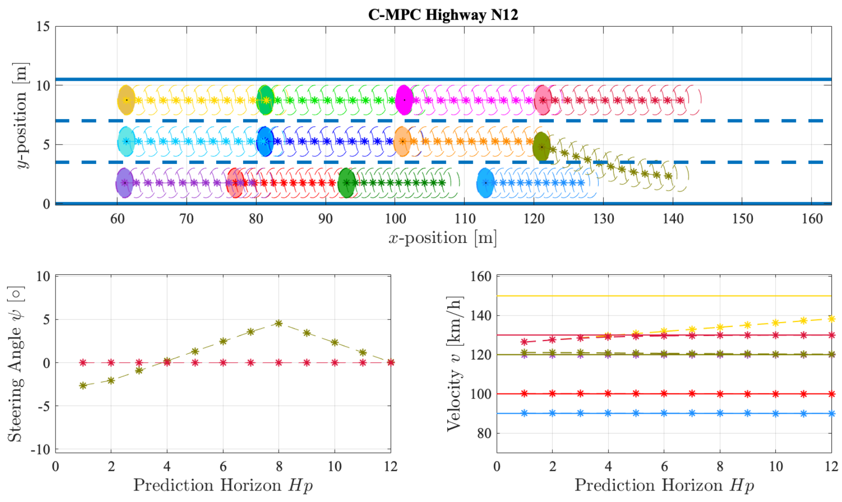


Figure 6-2: MPC iteration = 44, Lateral reference of the olive green vehicle is shifted to lower lane after overtaking.

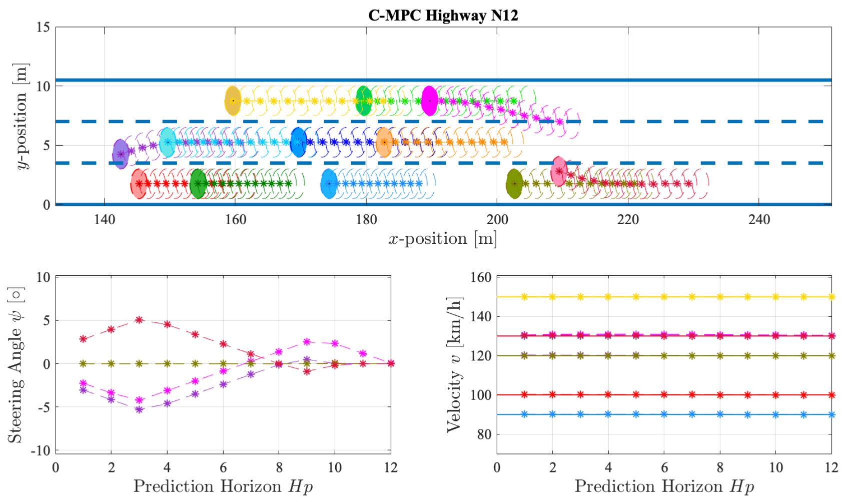


Figure 6-3: MPC iteration = 93, several more lane changes are performed to overtake or merge to a lower lane.

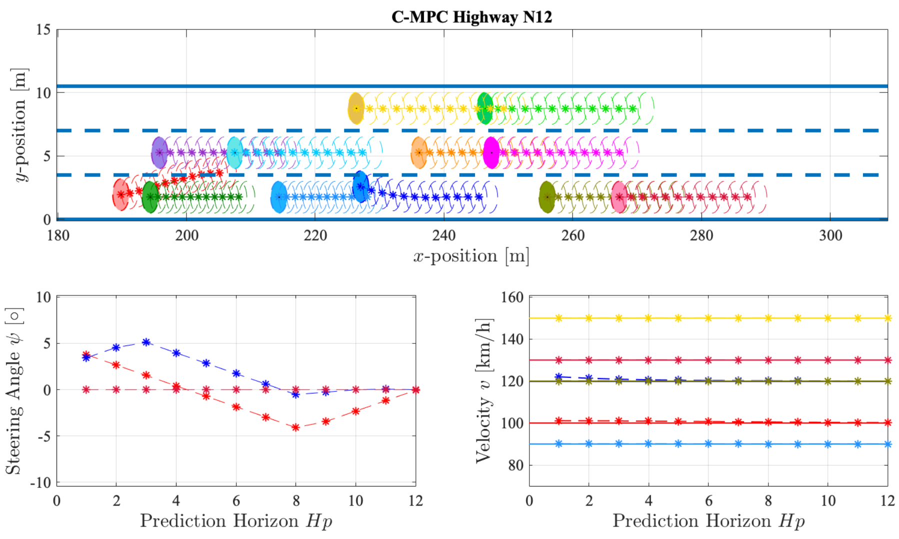


Figure 6-4: MPC iteration = 125, all vehicles are approaching there velocity reference.

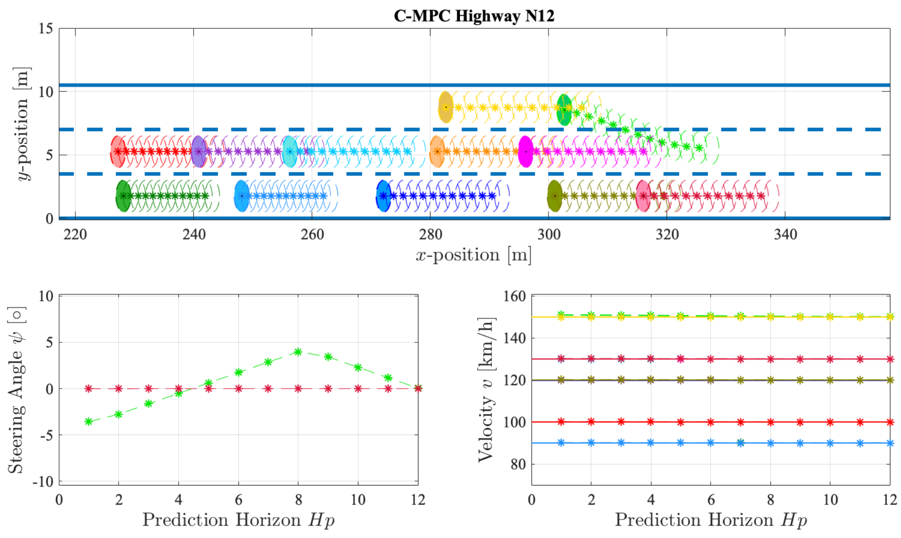


Figure 6-5: MPC iteration = 152, all vehicles have reached there velocity reference, when lower adjacent lanes are free, a lane change is performed.

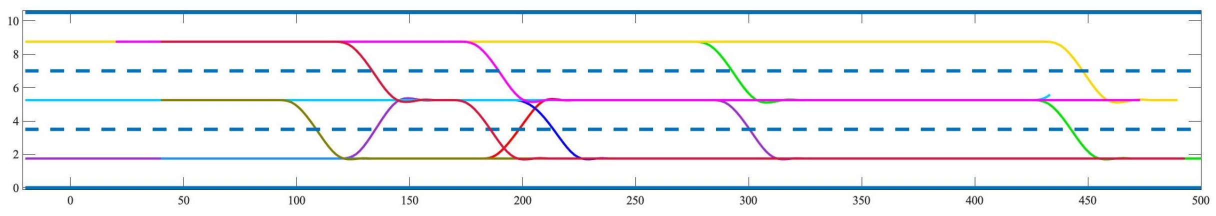


Figure 6-6: Testing scenario Highway, C-MPC for N=12 vehicles, full trajectories.

Trajectory Quality C-MPC

Below a bar graph is shown representing the individual cost values of C-MPC for each subsystem (for $N = 12$) using (5-2). This cost and according variance are later on used as benchmark to compare the individual satisfaction of each subsystem with respect to other distributed controllers. Note that the normalized objective values (5-1) and normalized path cost (5-3) are not shown since these would all equal one.

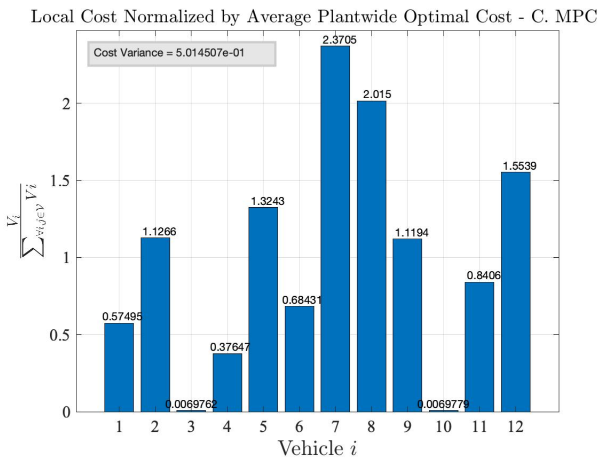


Figure 6-7: Local cost values C-MPC, scenario I

sub

Computation Time C-MPC

The figure below shows the minimum, maximum and mean computational times of C-MPC for increasing number of subsystems. The mean computation time represents the average time it takes the centralized controller to compute and predict the states of all vehicles for a single MPC iteration. As explained before, for (embedded) control it is important to finish computation before a predefined sample time. Hence, the minimum and maximum computation times are shown as well. Clearly, for increasing number of subsystems the maximum computation time increases as well. The overall mean computation time of C-MPC lies within 44 milliseconds for $N = 2$ and 360 milliseconds for $N = 12$.

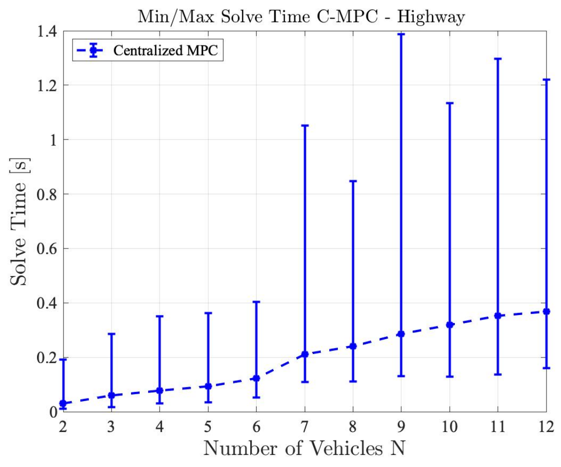


Figure 6-8: min, max and mean solve time C-MPC, scenario I

6-2 Decentralized MPC

Figure 6-9 only shows the final applied trajectories are shown, for N = 12 vehicles.

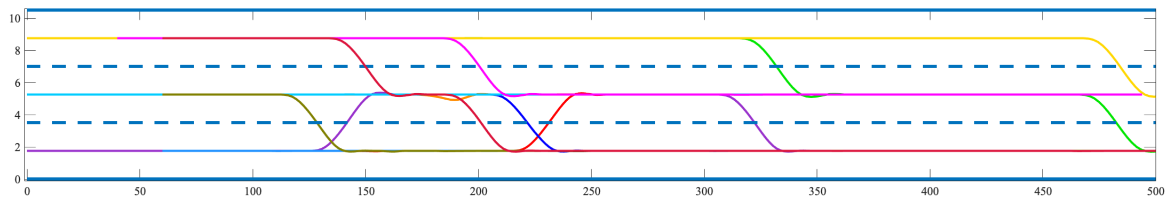


Figure 6-9: Testing scenario Highway, Dec. MPC for N=12 vehicles, full trajectories.

Trajectory Quality Dec. MPC

Figure 6-10 depicts the evaluation function values of Dec. MPC according to (5-1), (5-2) and (5-3), normalized with respect to those of C-MPC. The mean objective values represent the costs associated with the weighted steering angle ψ and weighted velocity error v_e. When comparing the trajectories between both controllers, the objective values are close to each other as well, e.g., the mean normalized objective values of Dec. MPC are close the one (the mean normalized objective values of C-MPC). Only for rising number of N, the differences become larger. This is due to the rise in complexity of the scenario and the lack of cooperation compared to the centralized controller.

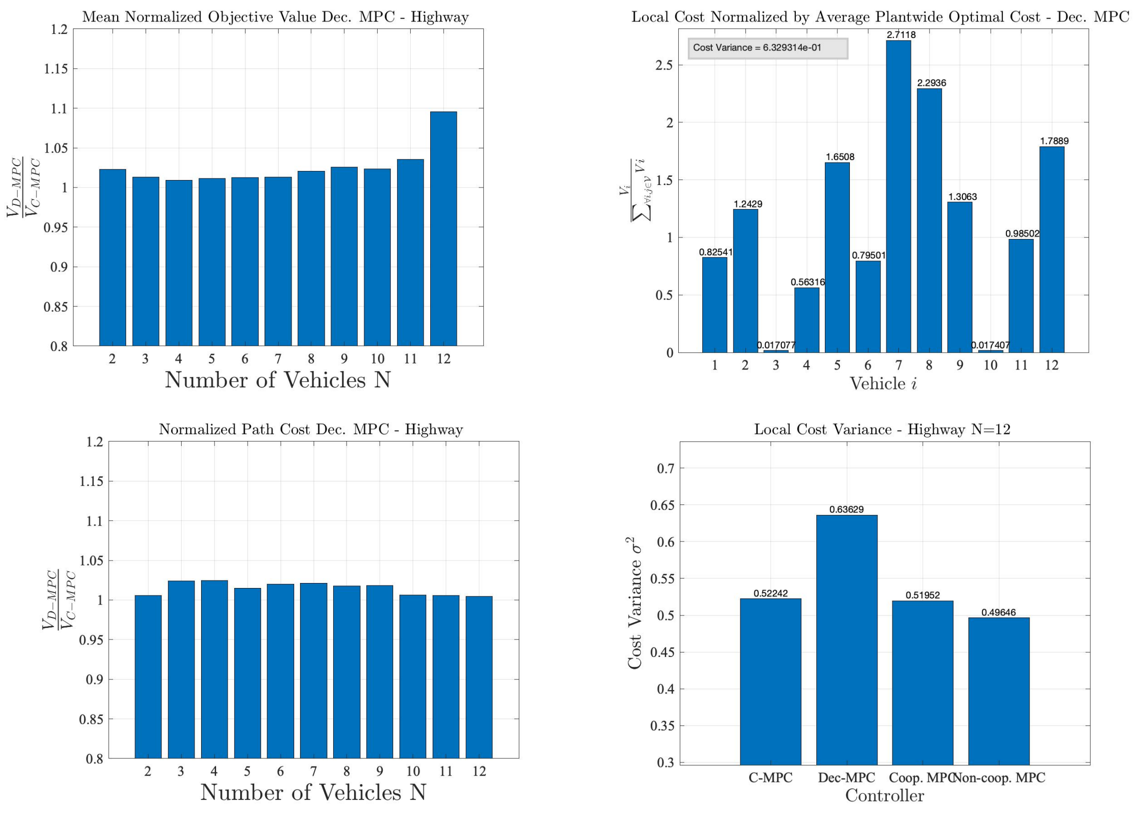


Figure 6-10: Mean Objective, Individual Satisfaction and Path Cost Dec. MPC, normalized with respect to average of C-MPC, Scenario I

One cause of the difference in cost, in the case of $N = 12$ vehicles is caused by a timing difference in Dec. MPC, that cases one vehicle to deviate from his lateral reference, as shown in the orange trajectory ($i = 5$) in figure 6-9 and local cost plot for $i = 5$ (compared to C-MPC $i = 5$). As expected, the variance of the cost between local subsystems is the highest for Dec. MPC since this controller has the lowest cooperation among subsystems.

Although the mean objective values of Dec. MPC are higher for any number of subsystems, the cost related to the progress made is similar to C-MPC in the case of $N = 12$ subsystems for Dec. MPC.

Computation Time Dec. MPC

Figure 6-11 depicts the average time required to solve a single Dec. MPC iteration for a single vehicle with $N = \{1, 2, \ldots, 12\}$, compared to the average time it takes the C-MPC controller.

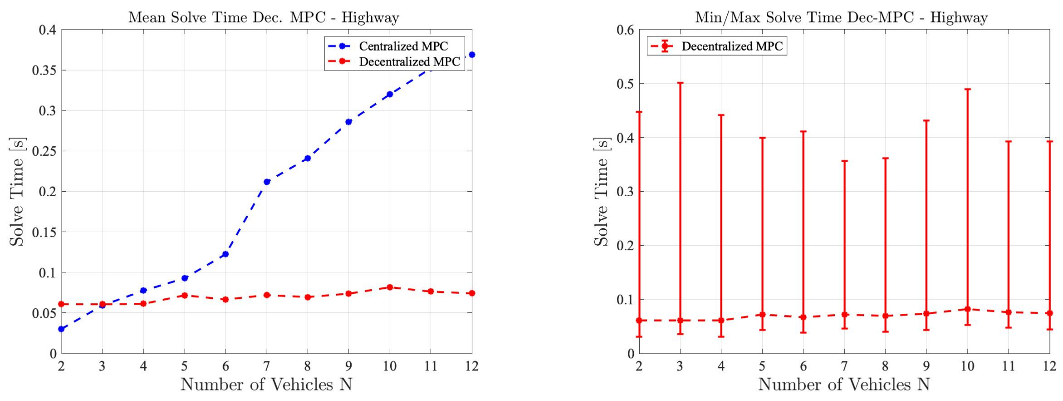


Figure 6-11: Mean, Min., and Max. computation time Dec. MPC, Scenario I

In the decentralized case, the mean computation time is computed by logging the time it takes for a single vehicle to solve its own local OCP and then taking the mean of these values. As expected this value remains equivalent for a growing number of subsystems. For $N = 2$ vehicles, the computation time for Dec. MPC is higher compared to the time it takes for C-MPC to solve the OCP. This can be explained by the addition of the compatibility constraint (4-9f) in Dec. MPC, which is not present in C-MPC. The smaller the value for $h^2\kappa$, the smaller the feasible space, the longer it takes for the non-linear OCP to be solved. The overall mean computation time of Dec. MPC lies within 60 and 82 milliseconds for any configuration of N.

Communication Dec. MPC

Having less cooperation among subsystems has a benefit in the sense that it requires less communication. In Dec. MPC, each subsystem communicates its own computed trajectory to neighboring subsystems before computing a new trajectory. For simplicity it is assumed that relative distance between neighboring subsystems does not affect the communication time.

The actual time it takes for vehicles to communicate depends on the communication protocol. Different Vehicle-to-Vehicle (V2V) communication protocols exist using different types of technology. A possible communication protocol is introduced in [50] using Dedicated Short Range Communication (DSRC). Here, automated vehicles communicate with each other at a rate of 10 message/second. Hence for simplicity, in this thesis we can assume that a single communication cycle takes $\pm 0.1s$.. Note that this is not yet taken into account in the computation time and will increase the overall time it takes for a vehicle to complete a single MPC iteration.

6-3 Cooperative Distributed MPC

The overall applied trajectories for $N = 12$ subsystems is shown in figure 6-12.

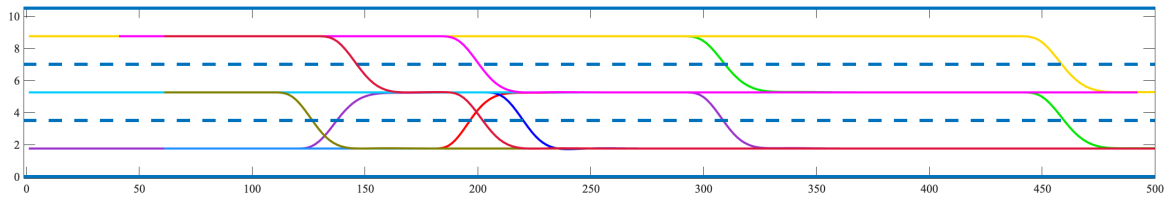


Figure 6-12: Testing scenario Highway, Coop. D-MPC for N=12 vehicles, full trajectories.

Trajectory Quality Coop. D-MPC

As expected, the objective values of (Sub-optimal) Cooperative Distributed MPC are close to the centralized version but do not converge to the centralized solution, see Observation 4.1. Below results are achieved using a maximum number of cooperative iterations of $p = 3$. Even lower objective values can be achieved by increasing p, but this decrease in objective value does not weigh up to the increased effort in communication.

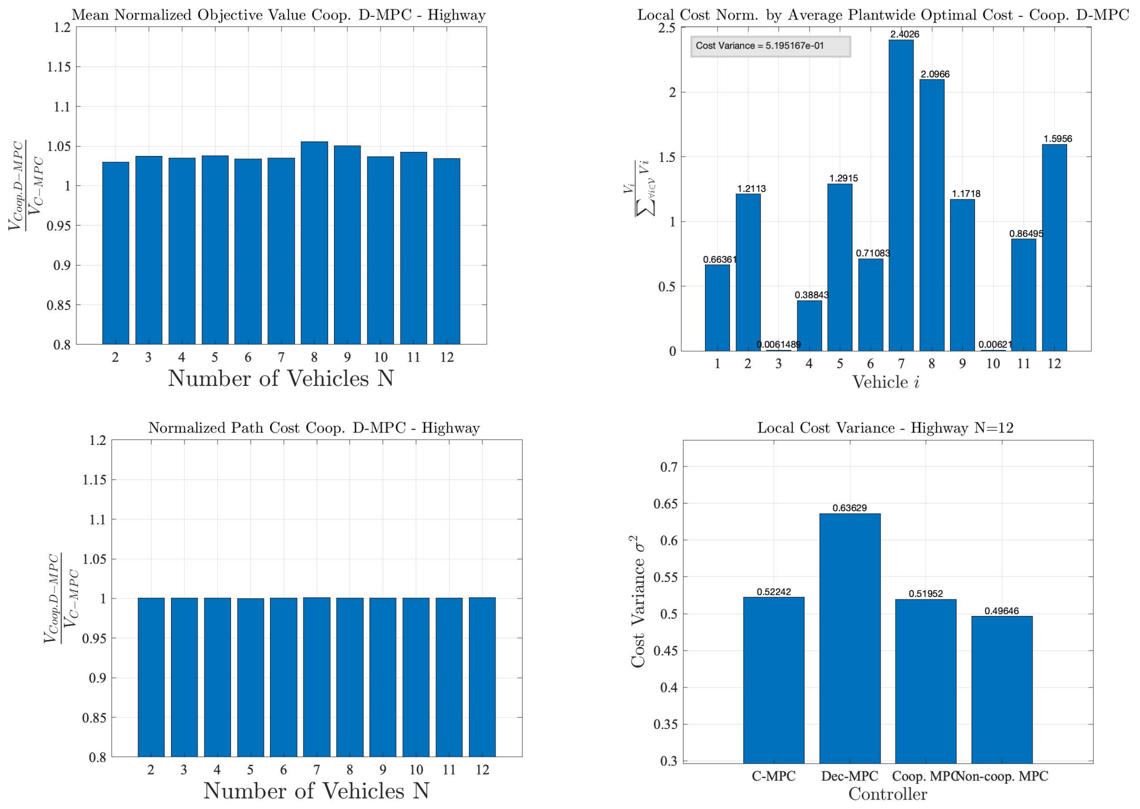


Figure 6-13: Mean Objective, Individual Satisfaction and Path Cost Dec. MPC, normalized with respect to average of C-MPC, Scenario I

Not only do the mean objective values closely represent the values of the centralized controller up to a difference of max 7%, the cost related to the progress made in the applied path equals that of the centralized version up to an accuracy of 0.1%.

The cost variance for $N = 12$ is similar to the centralized version with a variance of 0.5224 and 0.5195, for C-MPC and Coop. D-MPC, respectively.

Computation Time Coop. D-MPC

The computation time of Coop. D-MPC is computed as the average of the sum of computation times of a single MPC iteration for a single vehicle. For $p = 3$, each vehicle (re-)computes his trajectory 3 times. Hence, the sum of computation times is the time it took to (re-)compute its trajectory 3 times.

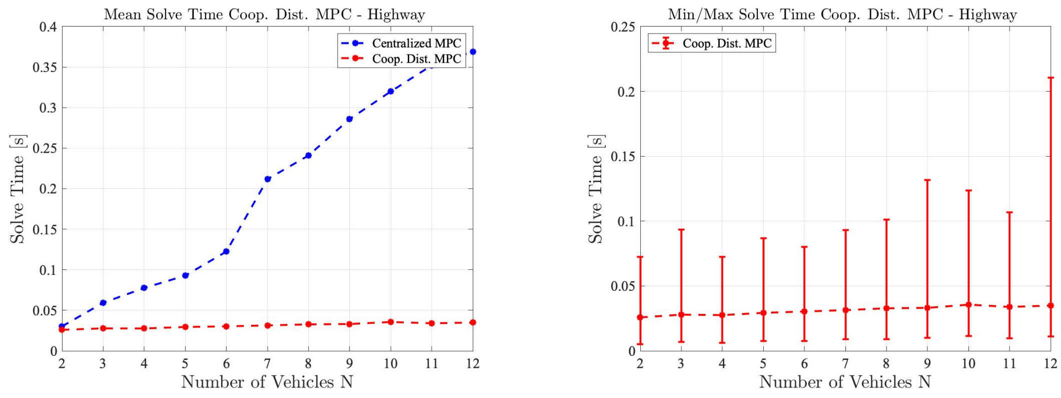


Figure 6-14: Mean computation time Coop. D-MPC, Scenario I

Clearly, the mean, min. and max. computation times of Coop. D-MPC are significantly lower than that of C-MPC. This not only is an effect of distributing the OCP over N subsystems, but also due to the fact Coop. D-MPC runs a linearized OCP in stead of a nonlinear one. As explained in chapter 3-2, the original nonlinear OCP is convexified using mixed integer variables. An inherent downside of mixed integer variables is the fact that computational time increases exponentially with increasing number of mixed integer variables. By cleverly making use of the Big-M method, this total number of binary variables can be reduced and hence there effect on the computational time is reduced too. The mean computation time of Coop. D-MPC lies within 25 to 35 milliseconds for any configuration of N.

Communication Coop. D-MPC

Similar to the computational time, the amount of communication depends on the number of cooperative iterations p. All results are achieved using $p = 3$, hence all neighboring subsystems communicate 3 times every MPC iteration. Note that in the case of Coop. D-MPC, the controller is able to stop prematurely. In this case, the number of communications is fixed to the actual applied number of cooperative iterations.

6-4 Non-Cooperative Distributed MPC

The last controller to be assessed in the first scenario is the Non-Coop. D-MPC based on a Generalized Potential Game. In figure 6-15, the final trajectories are displayed, for N = 12 vehicles.

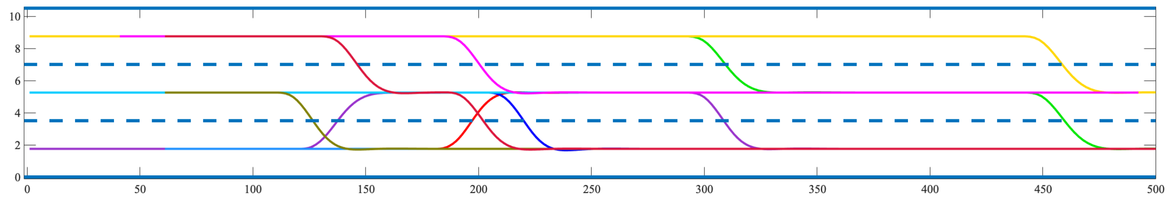


Figure 6-15: Testing scenario Highway, Non-Coop. D-MPC for N=12 vehicles, full trajectories.

Trajectory Quality Non-Coop. D-MPC

The objective values of Non-Coop. D-MPC are the closest to the values of C-MPC, with a maximum increase of 3.2% for $N = 5$ subsystems. Looking at the overall cost and path cost shown in figure 6-16, it can be stated the trajectories are close to 100% equivalent to those of the C-MPC controller.

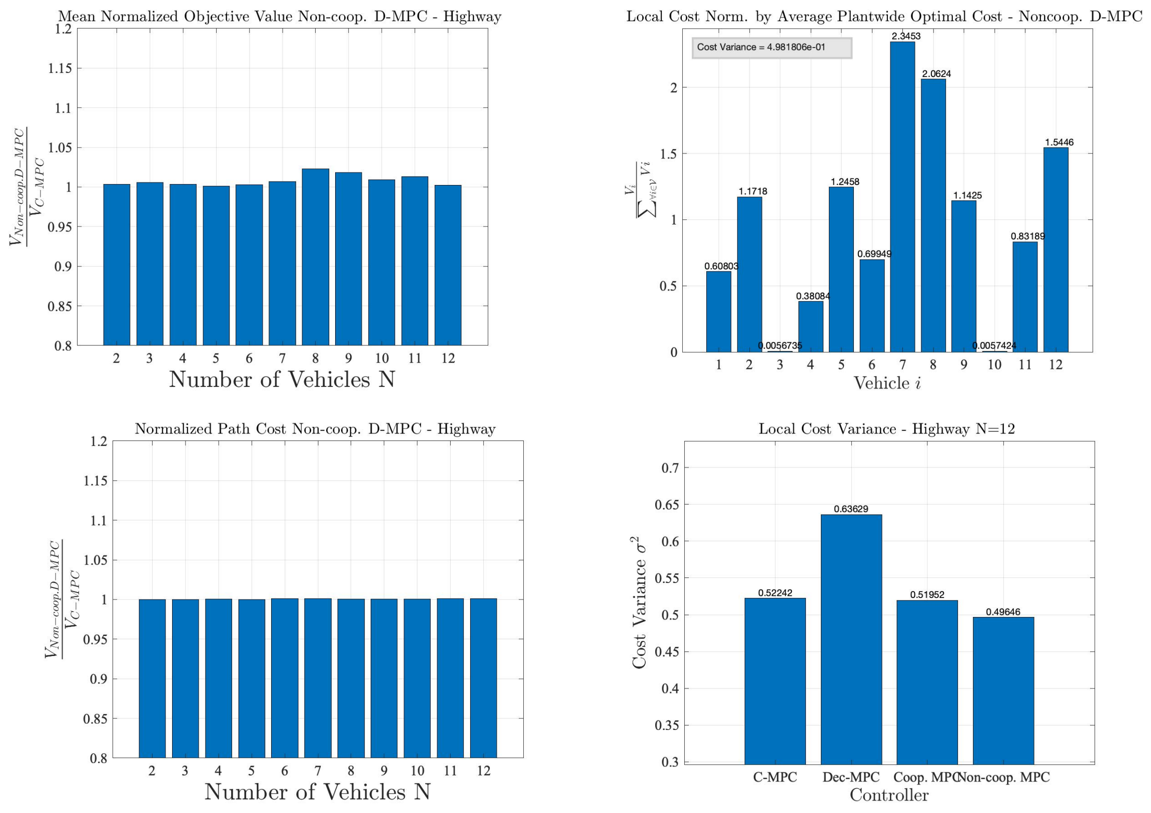


Figure 6-16: Mean Objective, Individual Satisfaction and Path Cost Dec. MPC, normalized with respect to average of C-MPC, Scenario I

As shown in section 4-5, in the Non-Coop. D-MPC controller, all subsystems communicate to try to find an ϵ-Nash Equilibrium. We would expect to find this notion of equilibrium back in the variance of the local cost, e.g., we would expect to have less variance since all subsystems (or rather players) agree upon the actions of the others and find there optimal solution accordingly. A minor decrease in variance is shown in figure 6-16 but this is almost negligible.

Since the cost for the progress in the path is equal and all trajectories are essentially equivalent to those of C-MPC, it is also expected that the individual cost of each subsystem(/player) is equivalent.

Computation Time Non-Coop. D-MPC

In contrast to the Coop. D-MPC controller, the number of algorithmic iterations is not fixed in the case of Non-Coop. D-MPC. Here, the players iterate until an ϵ-Nash Equilibrium is found, for any arbitrarily small ϵ. As a result, the computational time may vary too.

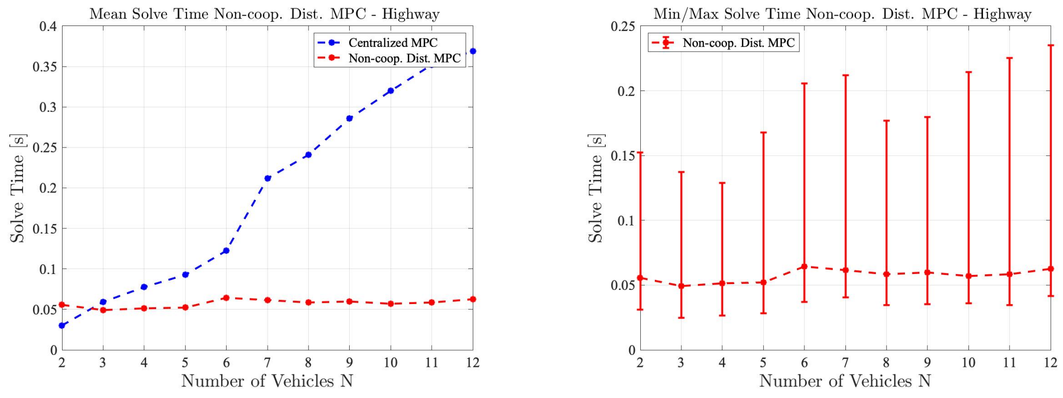


Figure 6-17: Mean computation time Non-Coop. D-MPC, Scenario I

Logging the number of game iterations showed that in general, in most MPC iterations, the players of the game found an ϵ-Nash-Equilibrium after 3 game iterations, occasionally needing 4 game iterations. As a result, the mean computation time of a single MPC iteration lies within 50 to 65 milliseconds for any configuration of N. Similar to Coop. D-MPC, Non-Coop. D-MPC benefits from faster computational times due to linearizing and convexifying the OCP.

Communication Non-Coop. D-MPC

Having a solution close to the Pareto plantwide optimal solution, even with increased consensus among subsystems/players, comes at the cost of requiring a significant amount of communication between subsystems. In stead of locally computing a solution in parallel of each other, the Non-Coop. D-MPC controllers solve there own local OCP sequentially and communicate this solution to players with lower ordering. As a result, the amount of communication of Non-Coop. D-MPC not only depends on the amount of game iterations, it also depends on the total number of subsystems/players, i.e.:

$$n_{comm}(N, n_{game}) = N * n_{game}(N), \tag{6-1}$$

where n_{comm} represents the amount of communications and n_{game} the median of the amount of game iterations. Figure 7-18 shows the average amount of communication depending on the total number of players N times the logged number of game iterations for each configuration of N.

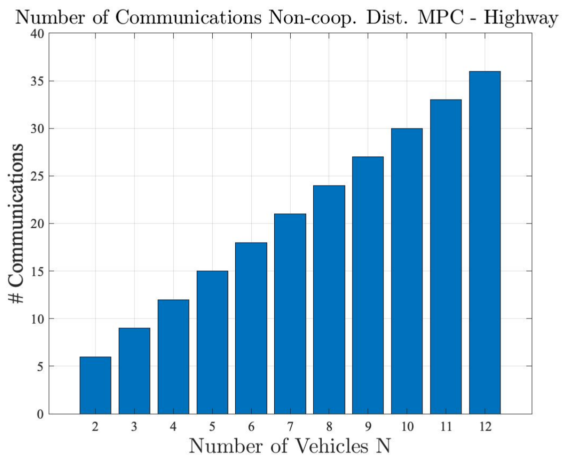


Figure 6-18: Non-Coop. D-MPC communications

6-5 Controller Comparison - Unrestricted Highway Scenario

In this section we focus on the results of our controllers for $N = 12$ vehicles and perform a comparison between them. Note that although all MPC controllers are different, trajectories might be similar due to the fact the ideal reference generated by the HLC is the same for all controllers.

Although all the MPC controllers are different, the ideal reference generated by the HLC is the same. Because of this, all trajectories are similar, but still, some numerical differences arise. Figure 6-19 shows the numerical values corresponding to trajectory quality criteria (5-1), (5-2) and (5-3) for $N = 12$ subsystems.

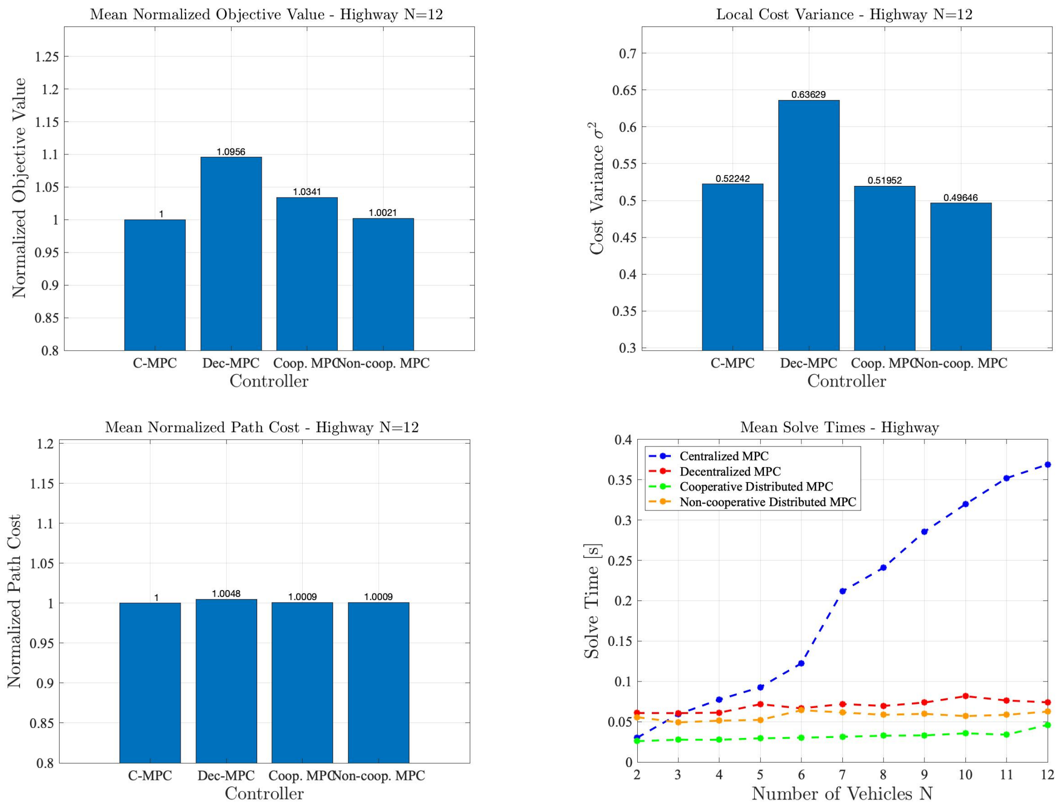


Figure 6-19: Trajectory Quality: Objectives, Costs, Cost Variance and Solve Time, Scenario I

As benchmark we use C-MPC, since this gives the plantwide Pareto optimal solution. Purely looking at cost and objective values, Non-Coop. D-MPC approaches this solution the most, having only an increase of 0.2% on the mean normalized objective value and an even smaller increase of $< 0.01\%$ on the path cost. This shows the controller has made virtually equivalent progress in the simulation and shows similar reference tracking behaviour as C-MPC.

Second comes Coop. D-MPC with an increase in objective of 3.4% and a similar increase in path cost of only $< 0.01\%$. Here, the sub-optimality of Coop. D-MPC plays a roll in the increase in mean normalized objective value. As shown in Appendix B, applying more cooperative iterations decreases the summed objective value close to the plantwide Pareto optimal solution. Lastly, the objective value and path cost of Dec. MPC show the highest increase of objective value but similar progress cost, with an increase of 9.6% and

0.48%, respectively.

Looking at the computation times in figure 6-19, C-MPC requires the longest time to solve the plantwide OCP, as expected. Also the maximum occurred computation time is significantly bigger for C-MPC, shown in figure 6-20. Second comes Dec. MPC, the main reason for this is that Dec. MPC is the only decentralized/distributed controller based on a non-linear optimization scheme, which in general, requires longer computation than convex solvers. The fastest to solve the plantwide OCP for a single MPC iteration, is Coop. D-MPC. By having a fixed[1] number of iterations and relying on a MIQP solver. Close after Coop. D-MPC comes Non-coop. D-MPC with an average computation time slightly higher than that of Coop. D-MPC. This is due to the non-fixed convergence criteria of the game theoretic controller.

Lastly, we assess the communicational load of our controllers, shown in figure 6-20. Clearly, the communicational load of Non-Coop. D-MPC is the highest. By having a sequential solving scheme, the communicational load increases with increasing number of subsystems. Second comes Coop. D-MPC, with a fixed number of iterations, $p = 3$ (assuming the cooperative iterations are not stopped prematurely). C-MPC and Dec. MPC both only require a single communication every MPC iteration.

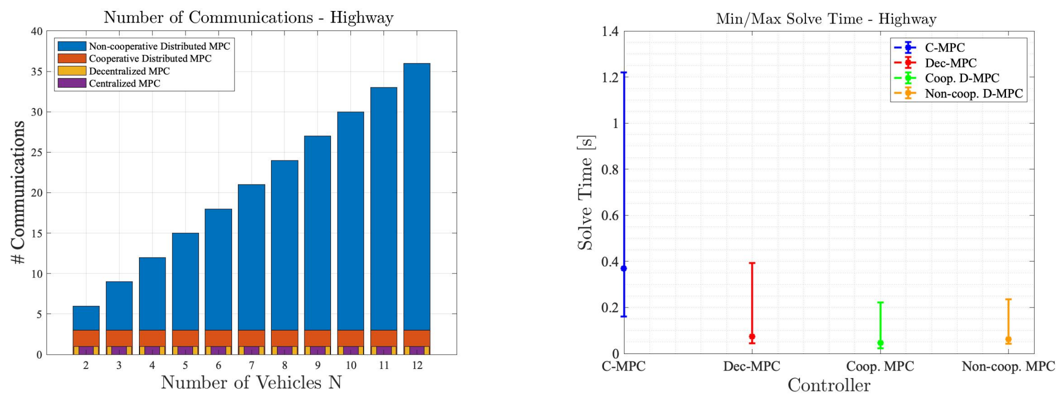


Figure 6-20: Communicational load - Unrestricted Highway Scenario

6-6 Conclusion Unrestricted Highway Scenario

In the first scenario, where vehicles simply accelerate to their reference velocity while having to avoid each other, we have seen that different methods of distributing or decentralizing the plantwide OCP gave different numerical results, even though all controllers are able to meet their reference, avoid collisions and even have similar trajectories.

Non-coop. D-MPC based on a Generalized Potential Game gives the closest result to the centralized solution, The controller makes virtually equivalent progress in the simulation and shows similar reference tracking behaviour as C-MPC. This is due to the fact each local subsystem sequentially solves their own local OCP. Hence, every subsystem computes their new trajectory based on the most recent information of others. This reduces possible estimation errors and increases efficiency of the overall control method.

[1]Number of cooperative iterations is fixed, although the controller is able to stop the iterative process before convergence

Coop. D-MPC and Dec. MPC follow, both with slightly higher objective values than C-MPC, but similar path costs. Showing that both controllers are slightly less efficient in reference tracking and cooperation, but overall progress made in the simulation is similar to that of C-MPC. In the case of Coop. D-MPC, efficiency can also be improved by increasing the number of cooperative iterations.

It can be concluded that in a simple scenario, e.g., no obstacles or obstructions, all controllers give similar results as C-MPC. This is due to the fact that, although all controllers are inherently different, the ideal reference generated by the HLC is the same for all controllers. Depending on the choice of controller, efficiency can be improved at the cost of having more communication and higher computational load. Additionally, since all controllers suffer from an occasional increase in computation time, for embedded control it is recommended that all controllers will be equipped with an extra emergency controller. Since it is highly likely that at some point in the experiment the controller will not be able to finish computation within the predefined sample time. Although, of course this is highly dependent on the processing power in the embedded controller. The design of an emergency controller is out of the scope of this thesis.

Chapter 7

Simulation Results Obstructed Highway Scenario

This chapter presents and discusses the simulation results for our second scenario: Roadworks Obstructed Highway. First the results are shown, and later on discussed. Similar to the first scenario, only the full trajectories of C-MPC are shown. For completeness we add the trajectories of the other controllers in Appendix C.

7-1 Centralized MPC

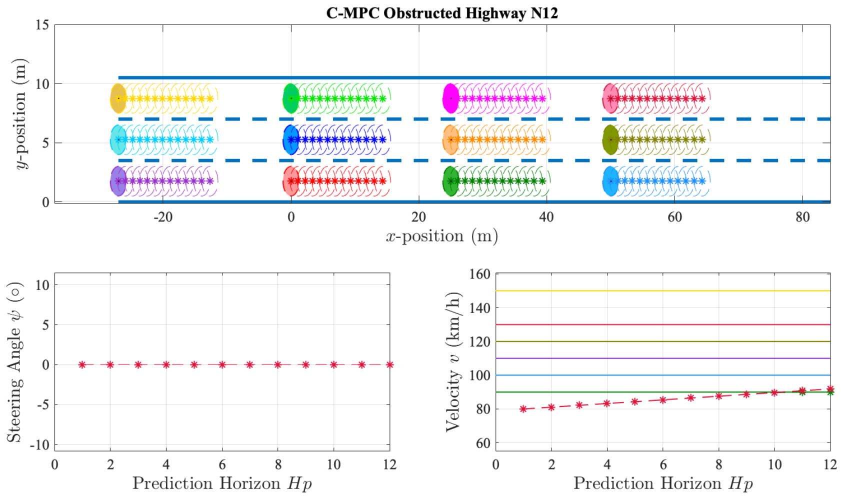


Figure 7-1: MPC iteration = 1, initial conditions.

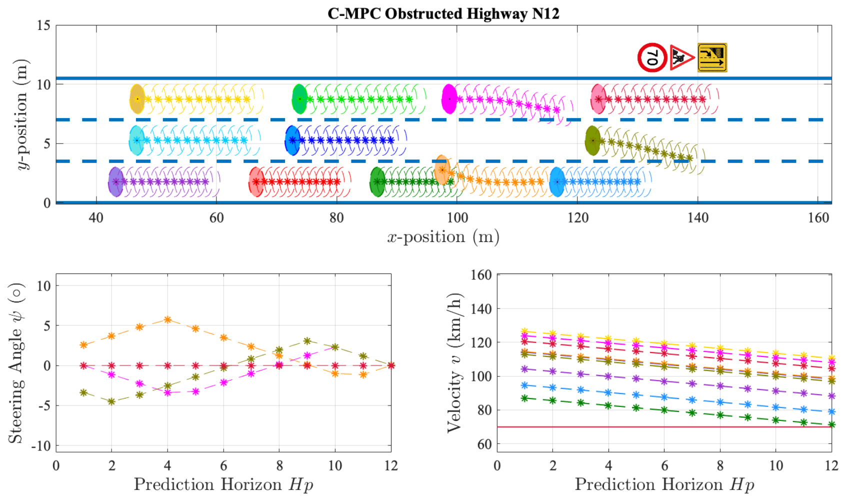


Figure 7-2: MPC iteration = 51, velocity reference is lowered to 70 kp/h, due to roadworks.

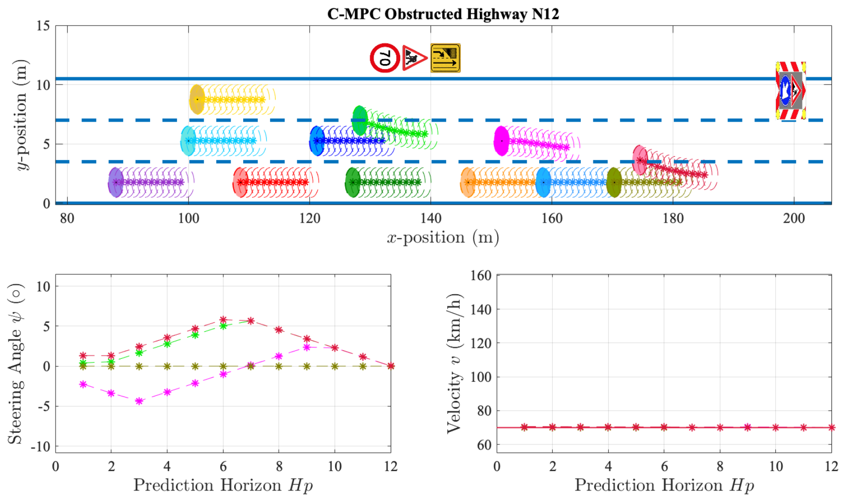


Figure 7-3: MPC iteration = 91, Approaching the roadworks, all vehicles will have to merge to the lowest lane

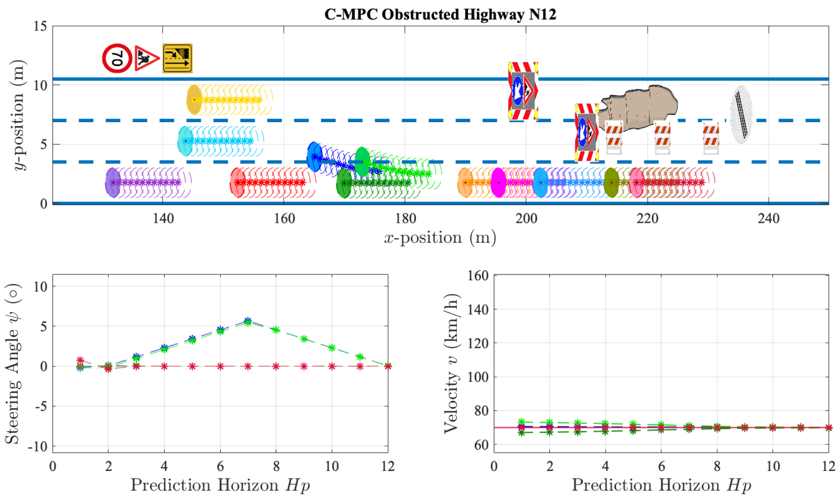


Figure 7-4: MPC iteration = 136, cooperation among vehicles is required to merge to the lowest lane.

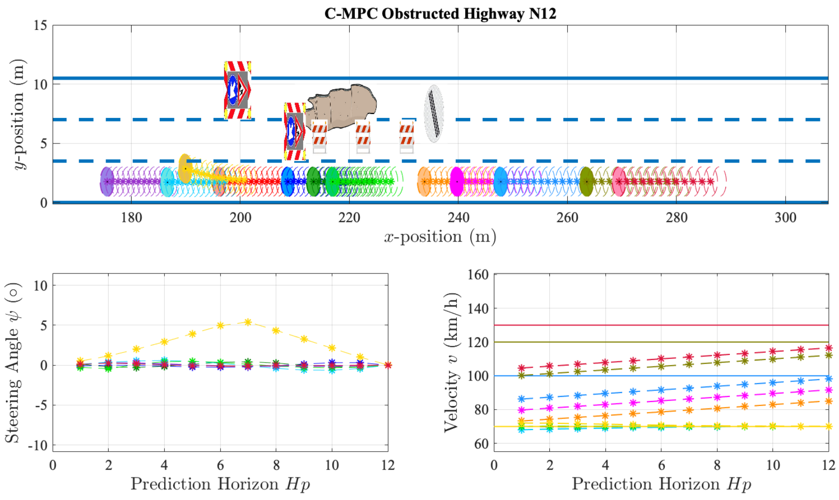


Figure 7-5: MPC iteration = 181, if roadworks are avoided, velocity reference is increased to initial velocity reference.

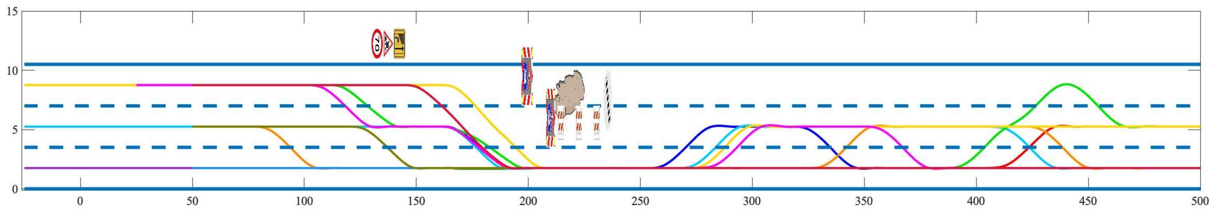


Figure 7-6: Testing Scenario Obstructed Highway, C-MPC for N=12 vehicles, full trajectories.

Trajectory Quality C-MPC

Figure 7-7 shows the individual cost values of C-MPC of each subsystem (for $N = 12$) in the second scenario. this cost is a result of (5-2). This cost and according variance are later on used as benchmark to compare the individual satisfaction of each subsystem with respect to other distributed controllers. Note that the normalized objective values (5-1) and normalized path cost (5-3) are not shown since these would all equal one.

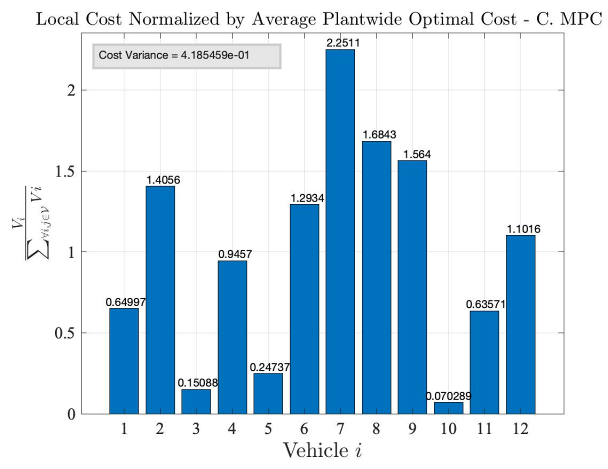


Figure 7-7: Local cost values C-MPC, scenario II

Computation Time C-MPC

Figure 7-8 below shows the minimum, maximum and mean computational times of C-MPC for $N = 12$ subsystems. The mean computation time represents the average time it takes the centralized controller to compute and predict the states of all vehicles for a single MPC iteration. Clearly, for increasing number of subsystems the maximum computation time increases as well. The overall mean computation time of C-MPC lies within 37 milliseconds for $N = 2$ and 710 milliseconds for $N = 12$. Not only does the mean computation time increase significantly compared to the first scenario, the maximum computation time that has occurred during the simulation has increased to unacceptable numbers, up to $36s$ for a single MPC iteration.

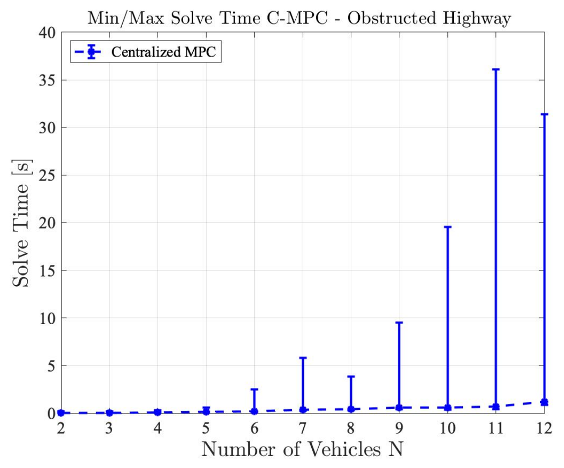


Figure 7-8: min, max and mean solve time C-MPC, scenario II

7-2 Decentralized MPC

The applied trajectories of Dec. MPC for N=12, in the second scenario are shown in figure 7-9. Intermediate trajectories are shown in Appendix C. All vehicles are able to avoid collision while tracking their reference.

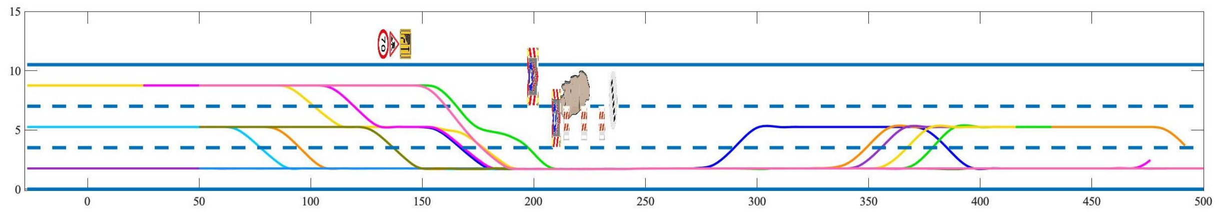


Figure 7-9: Testing scenario Obstructed Highway, Dec. MPC for N=12 vehicles, full trajectories.

Trajectory Quality Dec. MPC

Figure 7-10 depicts the mean objective values of Dec. MPC, normalized with respect to those of C-MPC. The objective and cost values represent the costs associated with evaluation criteria (5-1), (5-2) and (5-3).

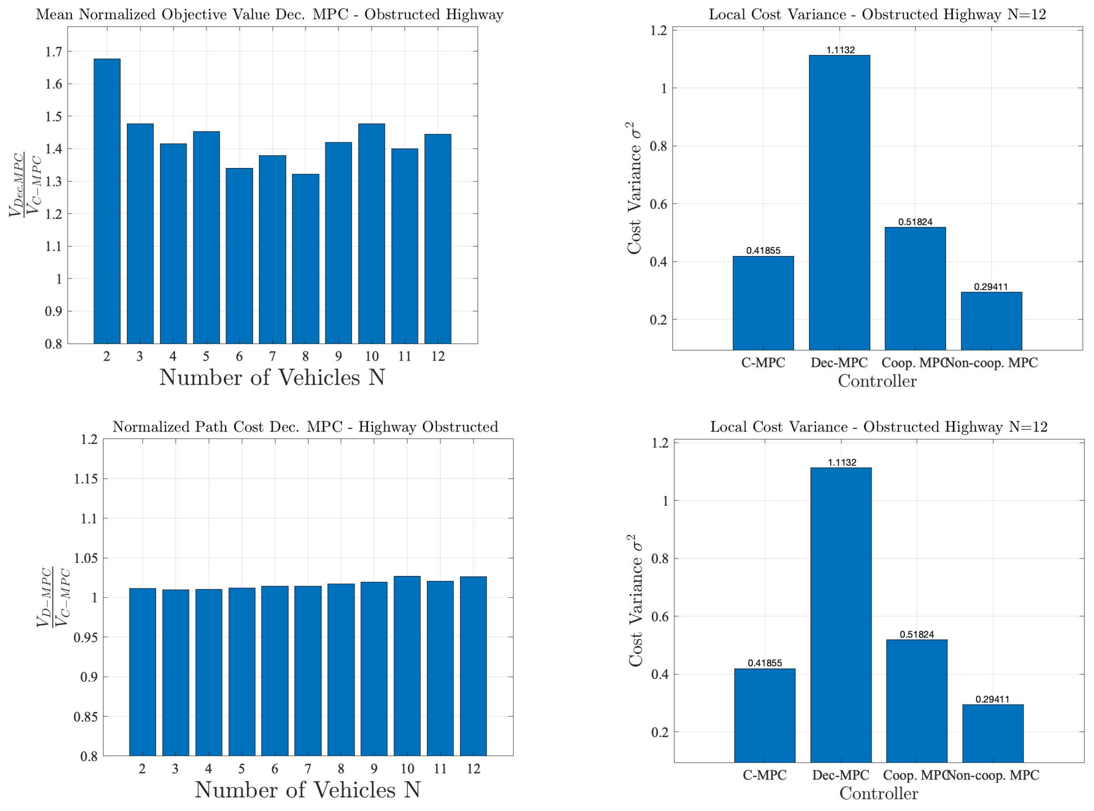


Figure 7-10: Mean Objective, Individual Satisfaction and Path Cost Dec. MPC, normalized with respect to average of C-MPC, Scenario II

For the second scenario, it's clear that the difference between the plantwide optimal solution of C-MPC and the solution of Dec. MPC is larger than compared to the unrestricted highway scenario. With a maximum increase of 67% and an average increase 44%. The rise in complexity requires more cooperation between vehicles to perform close to the Pareto optimal optimal solution. Since in Dec. MPC, the communication between vehicles is minimal, the vehicles are not able to reach the same level of cooperation as in the case of C-MPC. Not only do the objective values increase, the path cost also increases as a result of the lack of cooperation in Dec. MPC.

Computation Time Dec. MPC

Figure 7-17 depicts the mean,max and minimum time required to solve the OCP for a single MPC iteration. Due to the more complex scenario, the mean computation time of Dec. MPC has also risen. The nonlinear controller seems to be sensitive to increases in complexity, as the maximum occurred computational time also increased significantly.

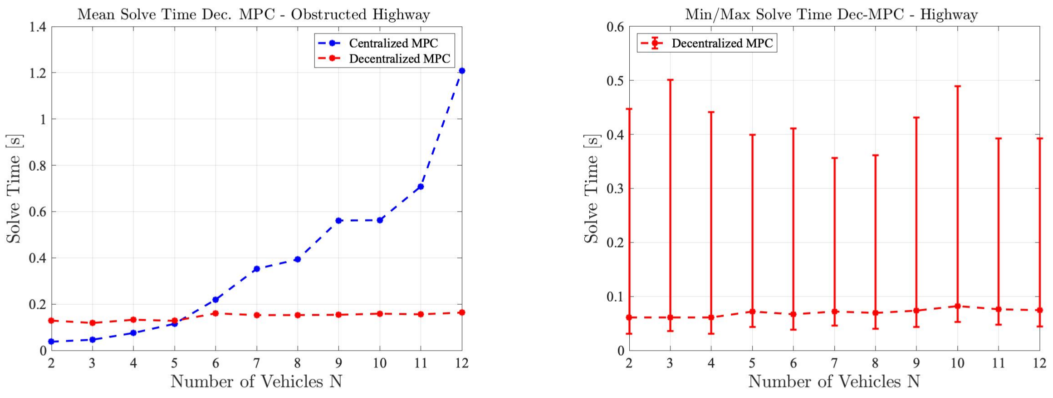


Figure 7-11: Mean, minimal and maximum computation time Dec. MPC, Scenario II

Communication Dec. MPC

For Dec. MPC, communication is fixed to a single time every MPC iteration. Hence, the communication of Dec. MPC for the second scenario is the same as for the first scenario.

7-3 Cooperative Distributed MPC

Figure 7-12 shows the applied trajectories of the Coop. D-MPC controller for N = 12 vehicles, in the second scenario. Applying sub-optimal steps results in a different time of the high-level-controller. Hence, the order in which vehicles merge to the lowest lane is slightly different and the overall trajectories are too.

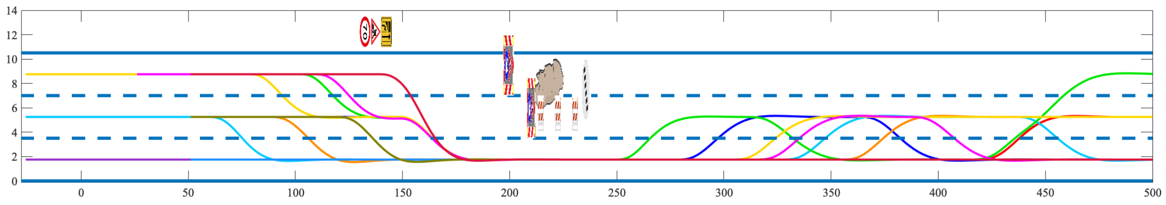


Figure 7-12: Testing scenario Obstructed Highway, Coop. D-MPC for N=12 vehicles, full trajectories.

Trajectory Quality Coop. D-MPC

Similar to the first scenario, all results are achieved using $p = 3$ cooperative iterations.

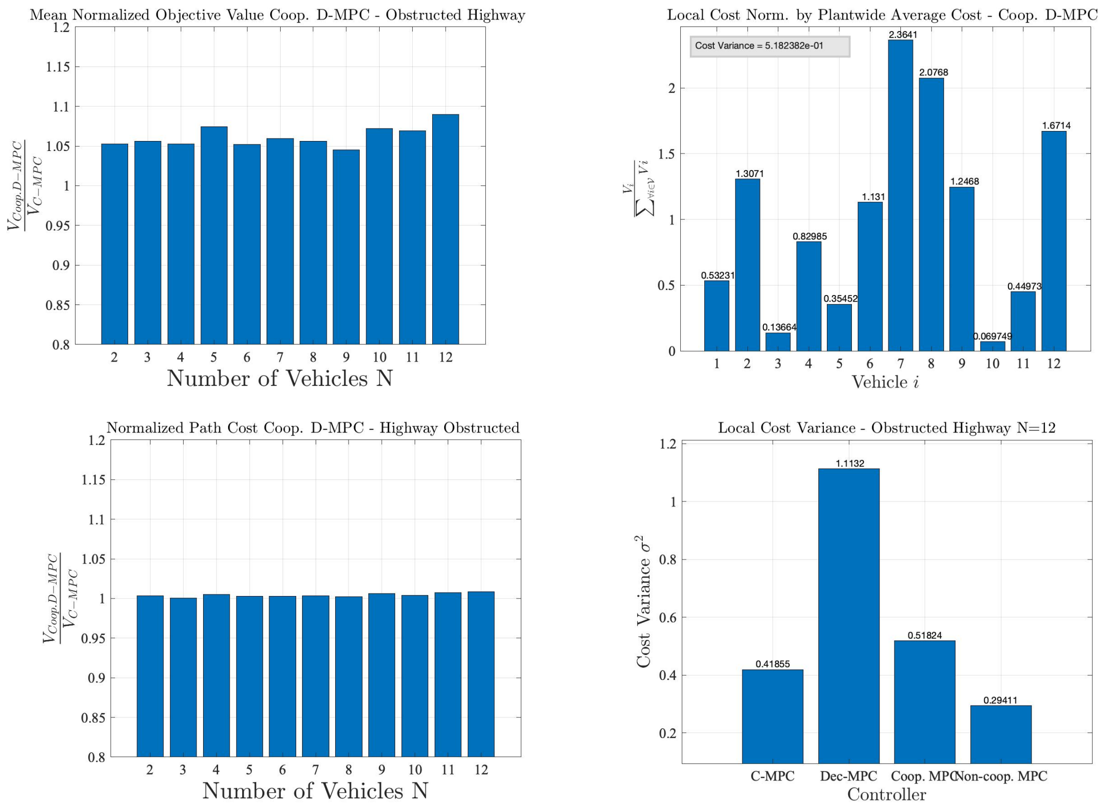


Figure 7-13: Mean Objective, Individual Satisfaction and Path Cost Dec. MPC, normalized with respect to average of C-MPC, Scenario II

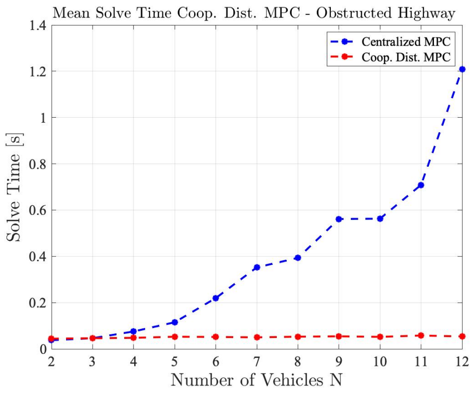
The difference in summed mean objective values of Coop. D-MPC between the first and second scenario is smaller as in the case for Dec. MPC. By communicating $p = 3$ times every MPC iteration, the level of cooperation is increased. As a results, Coop. D-MPC is able to approach the plantwide optimal solution up to a maximum difference of 8% with an average increase of 6% with respect to the centralized solution.

The progress made by the Coop. Dist. controller is similar to the progress made by the centralized controller. Although timing is different, the cost for the overall applied path is equivalent up to an increase of 1%. This offset between the cost for the applied path and objective value can be explained by the fact the controllers apply sub-optimal inputs. For $p = 3$ cooperative iterations, the applied input $\mathbf{u} = \boldsymbol{v}^{p_{max}}$, initialized by the warm start $\tilde{\mathbf{u}}$, is still sub-optimal due to the fact it has not yet converged to the actual computed input $\boldsymbol{v}^{p_{max}}$. As a consequence, the applied inputs result in trajectories in which the sub-optimality is measured in the velocity error, but not significantly seen in the overall progress that is made.

The variance of the local objective cost between the 12 subsystems of Coop. D-MPC is slightly higher than the variance of the plantwide optimal solution, namely 0.51824 and 0.4185, respectively. The main difference with respect to 7-7 is made in the cost of subsystem $i = 8$.

Computation Time Coop. D-MPC

Similar to the first scenario, the computation time shown in figure 7-14 is the time it took a local subsystem to (re-)compute their trajectory $p = 3$ times. Similar to other controllers, the computational time has risen slightly due to the rise in complexity of the scenario. The mean time ranges from 44 to 57 milliseconds, with outliers of maximum 262 milliseconds for $N = 12$.



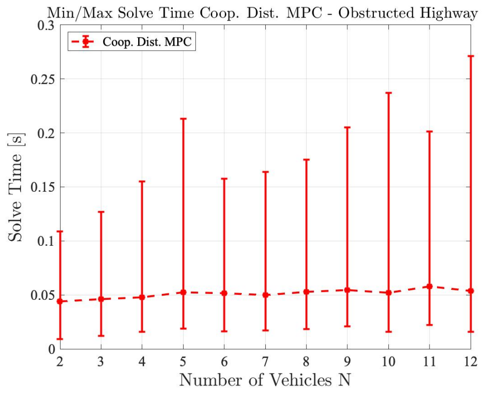


Figure 7-14: Mean computation time Coop. D-MPC, Scenario II

Communication Coop. D-MPC

If not stopped prematurely, the number of communication cycles for Coop. D-MPC is fixed to the number of maximum cooperative iterations p. All simulations are performed for $p = 3$.

7-4 Non-Cooperative Distributed MPC

Figure 7-15 shows the trajectories resulting from the Generalized Potential Game repeatedly solved by the Non-Coop. D-MPC, for N = 12 vehicles. An interesting result is the lane change performed by subsystem $i = 5$, e.g., the purple trajectory, which performs a lane change from the lowest to the middle lane, while other vehicle are all changing lane to lower lanes only. This could be explained as a possible result from the ϵ-Nash Equilibrium found by the Non-Coop. D-MPC controller.

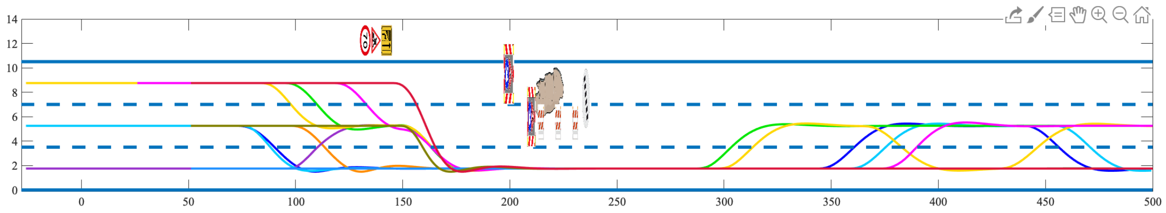


Figure 7-15: Testing scenario Obstructed Highway, Non-Coop. D-MPC for N=12 vehicles, full trajectories.

Trajectory Quality Non-Coop. D-MPC

Again the Non-Coop. D-MPC controller gives the best result in terms of the mean normalized objective values. It approaches the plantwide optimal solution the most, the same can be concluded for the path cost, with a maximum increase of 2.5% and 0.8%, respectively. Interestingly, although the mean normalized objective and normalized path cost are (almost) equivalent to the results of C-MPC, the local cost variance of Non-Coop. D-MPC is lower than that of C-MPC, namely 0.2941 and 0.4185, respectively. So although the plantwide solution is not as effective (but almost as effective), the individual solutions of Non-Coop. D-MPC tend to have a more consensus like property. This results from a property of the Generalized Potential Game, where all vehicles iterate until a collective strategy $\bar{x}$ is found, where each local player has found their *epsilon*-optimal cost decreasing trajectory with respect to the other players trajectories, i.e. an ϵ-Nash Equilibrium:

$$V_v\left(\bar{x}_i, \bar{\mathbf{x}}_{-i}\right) \leq V_v\left(y_i, \bar{\mathbf{x}}_{-i}\right), \quad \forall y_i \in \mathcal{X}_i\left(\bar{\mathbf{x}}_{-i}\right), \quad \forall i \in \mathcal{V}, \tag{7-1}$$

where $\bar{\mathbf{x}}_{-i}$ represents the *epsilon*-optimal set of strategies from all other players then i. Since the summed overall objective values of C-MPC and Non-Coop. D-MPC are equivalent, it can be stated that the average individual satisfaction of the Non-Coop. D-MPC controller is higher. This is considered as a beneficial property of Non-Coop. D-MPC.

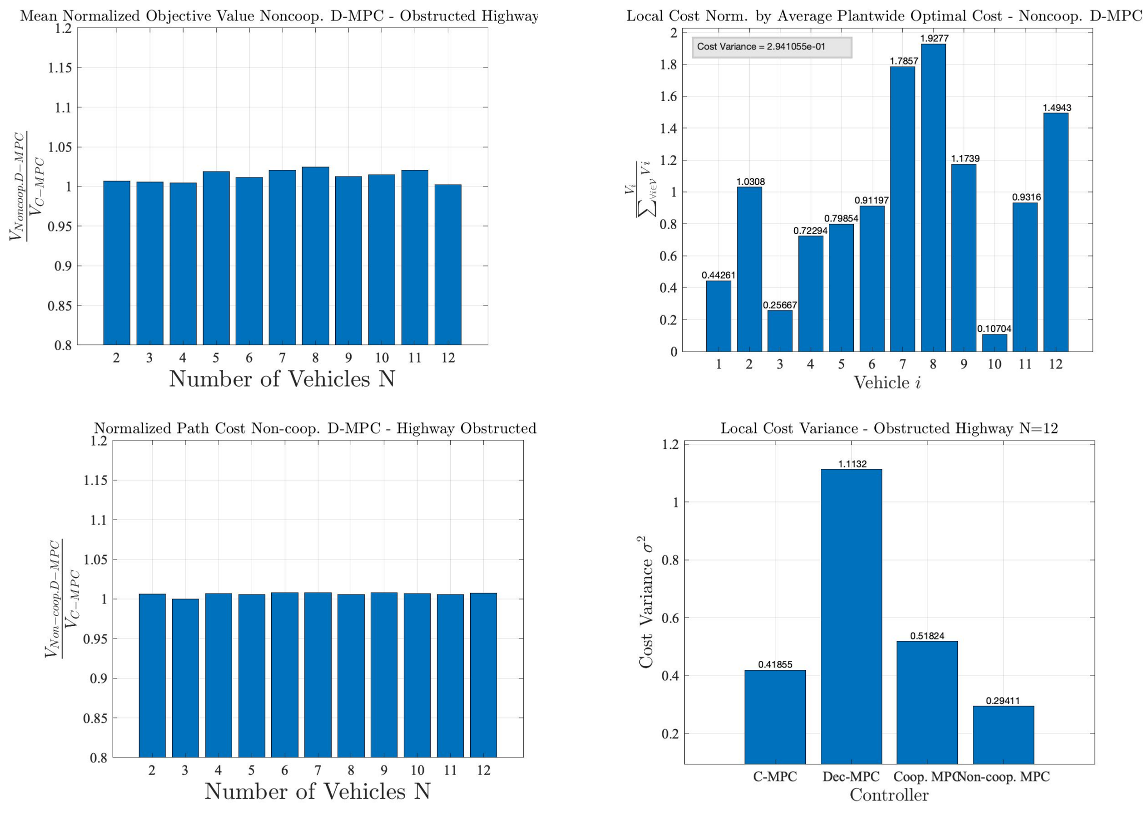


Figure 7-16: Mean Objective, Individual Satisfaction and Path Cost Non-Coop. D-MPC, normalized with respect to average of C-MPC, Scenario II

Computation Time Non-Coop. D-MPC

For the second scenario, the amount again depends on the number of game iterations which where required to solve the Generalized Potential Game. Logging these game iterations, we found that a solution to the game is found in 3 to 4 game iterations, with a median of 3. Similar to the first scenario, Non-coop. D-MPC benefits from faster computational times due to linearizing and convexifying the OCP, compared to the C-MPC and Dec. MPC.

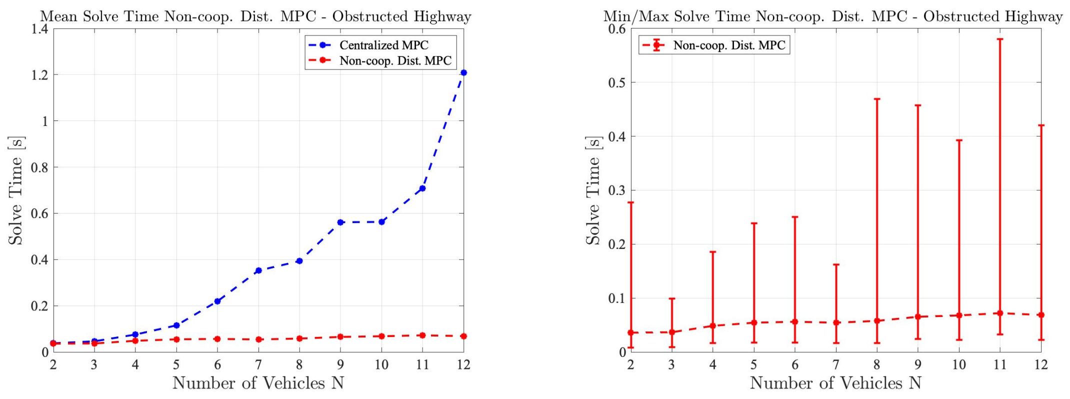


Figure 7-17: Mean, minimal and maximum computation time Non-Coop. D-MPC, Scenario II

The mean computation time ranges from 38 to 64 milliseconds, with occasional outliers up to 420 milliseconds. This is an increase compared to Coop. D-MPC, which is the only other controller using a linearized and convexified OCP. This increase can be explained by the converging/stopping criteria of Non-Coop. D-MPC. Solving the game is not stopped prior to finding a ϵ-Nash Equilibrium.

Communication Non-Coop. D-MPC

Again, the amount of communication depends on the amount of game iterations and the total number of subsystems/players, i.e., (7-18). Logging the amount of game iterations for any configuration of N and using the median as input for $n_{game}(N)$, the following amount of communications is needed for the second scenario:

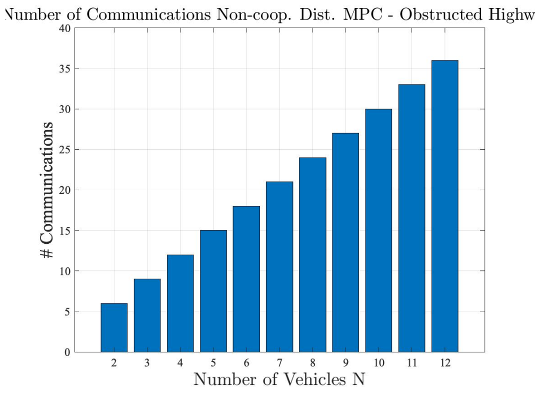


Figure 7-18: Non-Coop. D-MPC communications

7-5 Controller Comparison - Obstructed Highway Scenario

In contrast to the first scenario, the second scenario shows more distinction between controllers. With increasing level of complexity comes the need of more cooperation between vehicles in controllers. Figure 7-19 shows the numerical values corresponding to trajectory quality criteria (5-1), (5-2) and (5-3) for $N = 12$ subsystems in the second scenario.

Especially Dec. MPC shows a significant increase in mean objective value. Clearly in a more complex scenario, the controller is not able to achieve close to plantwide optimal control. By only communicating a single time and restricting each local feasible set by (4-9f), the controller performs up to 45% less efficient as C-MPC (in the case of $N = 12$ subsystems). Although reference tracking is less efficient, the overall cost for the progress that is made is only increased by 2.6%. This shows that although efficiency is significantly lower, the controller is still able to meet its target position.

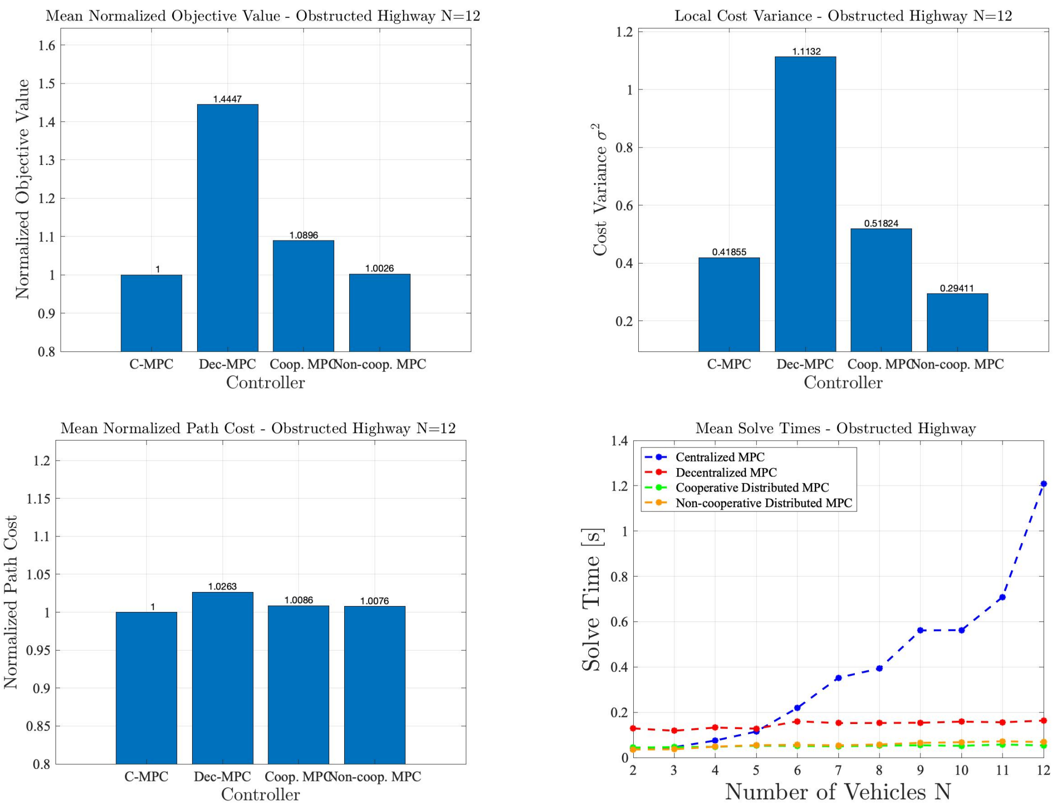


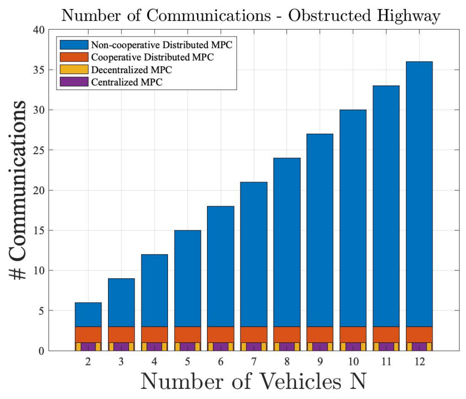
Figure 7-19: Trajectory Quality: Objectives, Costs, Cost Variance and Solve Time, Scenario I

Both cooperative and non-cooperative distributed controllers perform significantly better with a mean normalized objective value of 1.089 and 1.003, respectively. Communicating $p = 3$ times seems to increase the level of cooperation enough to approach the centralized solution up to 9%, while the non-cooperative approaches the centralized solution up to an increase of just 1%. Please note that this does not necessarily hold for other configurations of N. Of course, performance of Coop. D-MPC can be improved by increasing the number of cooperative iterations. Both controllers show virtually equivalent cost as C-MPC for the progress in path made in the simulation.

Furthermore, probably the most interesting result can be derived from the variance between the local mean objective cost. The game theoretic controllers shows that although the plantwide objective cost is slightly higher, the individual solutions of Non-Coop. D-MPC tend to have a more consensus like property. Where C-MPC controls all subsystems the most efficiently, at the cost of some subsystem's individual satisfaction, Non-Coop. D-MPC is able to find a higher averaged individual satisfaction. This is derived from the fact Non-Coop. D-MPC has a lower variance between individual cost values. Since both C-MPC and Non-Coop. D-MPC have an almost equivalent summed overall cost, the variance can be used to state that the average individual satisfaction is higher for Non-Coop. D-MPC, showing a more consensus-like solution.

In general, due to the increased complexity of the second scenario, all controllers require longer computational times than compared to the first scenario. C-MPC shows a significant increase in mean and maximum computational time, as shown in figure 7-19 and 7-20. Second comes Dec. MPC, followed by Coop. and Non-Coop. D-MPC.

Assessing the communicational load of our controllers, shown in figure 6-20. We see similar behaviour as for the first scenario. The communicational load of Non-Coop. D-MPC is the highest, due to the sequential solving scheme. Second comes Coop. D-MPC, with a fixed number of iterations, $p = 3$ (assuming the cooperative iterations are not stopped prematurely). C-MPC and Dec. MPC again only require a single communication every MPC iteration.



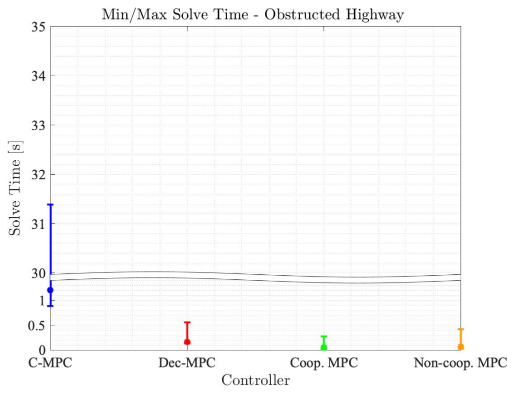


Figure 7-20: Communicational load - Unrestricted Highway Scenario

7-6 Conclusion Obstructed Highway Scenario

In a more challenging scenario, e.g, an obstructed highway, a bigger distinction among controllers emerges in our simulations. With increasing level of complexity comes the need of more cooperation between vehicles in controllers. Although all trajectories of all controller are able to avoid collision, the difference in cooperation and hence efficiency is more evident.

In C-MPC, when merging to the lowest lane, one vehicle slows down while the other speeds up in order to be able to merge to one lane, see figure 7-4. In Dec. MPC for example, the vehicle already in the lower lane, does not alter his own velocity to enable the other vehicle to merge. This results in a less plantwide optimal solution, as is expected.

Non-Coop. D-MPC performs the best in sense of the mean normalized objective value, followed closely after by Coop. D-MPC. Both controllers show similar progress when it comes to the applied path. Dec. MPC shows a significant increase in mean objective. In a more complex scenario, the controller is not able to achieve close to plantwide optimal control.

Perhaps the most interesting result shown, is achieved by the game theoretic controller. This shows that although the mean plantwide objective cost is close too, but slightly higher than C-MPC, the individual solutions of Non-Coop. D-MPC tend to have a more consensus like property, while C-MPC controls all subsystems the most efficiently, at the cost of some subsystem's individual satisfaction, Non-Coop. D-MPC is able to find a higher averaged individual satisfaction.

Combining the plantwide objective, individual objective and the progress made in the path, Non-Coop. D-MPC performs the best. This performance however comes at a cost of an extremely high communicational load, due to the fact all local subsystems solve their local OCP sequentially. Second in lines comes Coop. D-MPC, where all local subsystems compute their own trajectory $p = 3$ times. Although no increased average individual satisfaction, the controller is still able to achieve performance similar to the plantwide optimal solution, while only communicating up to maximum 3 times, with the possibility of stopping the cooperative iterations before convergence.

Chapter 8

Conclusion and Recommendation

Although C-MPC gives a plantwide optimal solution, implementation of plantwide control is often not practical due to computational and organizational complexity. Distributing the OCP over local subsystems solves this issue.

In this thesis, we compared the performance of three different distributed MPC controllers for multi-vehicle autonomous driving. All distributed controllers are able to meet their reference and avoid collisions, in our simulations. The non-Coop. D-MPC controller was able to approach the centralized solution by sequentially solving a single local OCP according to a predefined order and communicating its solution to vehicles with lower order. This advantage comes at the cost of high communication effort and having to follow strict ordering among subsystems. As a consequence, it is difficult to guarantee that all vehicles are able to compute their final solution within the predefined sample time of the MPC controller. This strategy relies on fast communication and the ability of each local subsystem to quickly compute a solution to their OCP.

With Coop. D-MPC however, all local subsystems compute their solution synchronously and in parallel to each other and iteratively apply a sub-optimal cooperative step for a fixed number of iterations, p. The performance of Coop. D-MPC improves for an increasing number of p, but due to the non-convexity of the collision avoidance constraints, Coop. D-MPC dues not guarantee to converge to the centralized optimal solution, while simulations show that it is able to approach to this centralized solution up to acceptable accuracies. Obviously, communication and computation time increase for an increasing number of p. In this thesis, all simulation are performed with $p = 3$. This gives a good trade-off between performance and communicational and computational load. Another benefit of Coop. D-MPC is the ability to stop the iterative sub-optimal algorithm prior to meeting its stopping criteria. This adds extra flexibility and robustness to the controller.

In simple scenarios, Dec. MPC has shown the ability to give similar performance as C-MPC, while limiting communication and computation to a single iteration. For increasing complexity, Dec. MPC has shown to have degrading performance and shows a lack of cooperation due to limited communication.

It can be concluded, that for the three assessed distributed controllers, Coop. D-MPC has the best trade-off between feasibility in sense of communication, computation and robustness, while still showing good performance. If the total number of subsystems is low and communication is fast and cheap, Non-Coop. D-MPC can be considered too. In fact, our simulations show that only Non-Coop. D-MPC can ensure the best performance among our distributed controllers, but also has the advantage of providing solutions that are more fair, where the cost of cooperation is shared among vehicles.

For future studies, it is recommended too further investigate the communicational load of the distributed controllers and the corresponding feasibility for embedded control. Since the load of communication is not studied in simulations or experiments, and only assumptions are made according to other studies. Also, the stability of the distributed controllers is not studied in this thesis. In the case of Dec. MPC and Coop. D-MPC, this is still an an active field of research, due to the non-convexity of the collision avoidance constraints. Furthermore, stability for game theoretic controllers, even without the presence of non-convex constraints, is also an active field of research.

Appendix A

Stability Proof LTI MPC

With a MPC controller one can impose state and/or input constraints on the system. Below one can find an overview of the constraint types and the constraints itself. There are four types of constraints; State-, Input-, Terminal- and Stability constraints. Just as an example, we refer to possible constraints for a vehicle in a highway scenario, using the bicycle model in (3-1).

Constraint type	
State constraint	$\mathbb{X} = \left\{ \begin{bmatrix} \lvert x \rvert \\ \lvert y \rvert \\ \lvert \psi \rvert \\ \lvert V \rvert \end{bmatrix} \leq \begin{bmatrix} x_{max} \\ y_{max} \\ \psi_{max} \\ V_{max} \end{bmatrix} \right\}$
Input constraint	$\mathbb{U} = \left\{ \lvert \delta \rvert \leq \delta_{max} \right\}$
Terminal constraint	$\mathbb{X}_\text{f} = \left\{ \begin{bmatrix} \lvert x - x_\text{ref} \rvert \\ \lvert y - y_\text{ref} \rvert \\ \lvert \psi - \psi_\text{ref} \rvert \\ \lvert V - V_\text{ref} \rvert \end{bmatrix} \leq \begin{bmatrix} x_\text{ref. error} \\ y_\text{ref. error} \\ \psi_\text{ref. error} \\ V_\text{ref. error} \end{bmatrix} \right\}$
Stability constraint	$\forall x \in \mathbb{X}_f, \exists u \in \mathbb{U}$ s.t. $f(x,u) \in \mathbb{X}_f$ and $V_\text{f}(f(x,u)) \leq V_\text{f}(x) - \ell(x,u)$

Table A-1: All constraint types and the constraint values itself for the MPC controller are described

An example for state constraints could be the road space-time, the vehicle should not be allowed to drive of the road and higher than the allowed speed limit. The maximum steering angle poses a constraint on the input and the terminal constraints ensure reference tracking.

To obtain asymptotic stability we are looking for an input such that $\forall x, \exists u \in \mathbb{U}$, the following two conditions from [26, Section 2.4] hold [1]:

1. $V_f(x(t+1)) \leq V_f(x(t)) - \ell(x(t), u(t))$,
2. $x(t+1) \in \mathbb{X}_f$, where $\mathbb{X}_f$ is a positively invariant terminal set.

The first condition guarantees a Lyapunov decrease, such that the terminal cost is less or equal than the terminal cost in the previous time step. To prove 1, it is first necessary to assume that the Lyapunov function equals the optimal value function, $V_\infty^{\mathrm{u}} = \frac{1}{2}x^TPx$. The unconstrained infinite-horizon optimal control problem is given below.

$$\mathbb{P}_\infty^{\mathrm{uc}}(x) : \min_{\boldsymbol{u}} \sum_{k=0}^{\infty} \ell(x(k), u(k)) \tag{A-1}$$

where,

$$x(k) = \phi\left(k; x_0, \boldsymbol{u}_k\right) = A^k x_0 + \mathcal{C}_k \boldsymbol{u}_k \tag{A-2}$$

and,

$$\mathcal{C}_k = \left[\begin{array}{ccccc} A^{k-1}B & \dots & A^2B & AB & B \end{array}\right] \tag{A-3}$$

From equation A-1 the optimal steady-state LQR gain, K, is found as well as the direct solution P. The Discrete Algebraic Riccati Equation (DARE) and the results from the unconstrained infinite-horizon optimal control problem are given below.

$$\begin{aligned} 0 < P &= A_K^\top P A_K + Q_K \\ K &= -\left(B^TPB + R\right)^{-1} B^TPA^T \end{aligned} \tag{A-4}$$

where,

$$\begin{aligned} A_K &= A + BK \\ Q_K &= Q + K^TRK \end{aligned} \tag{A-5}$$

Now we define the optimal value function as our Lyapunov function.

$$V_f(x) := V_\infty^{\mathrm{uc}}(x) = \frac{1}{2}x^TPx \tag{A-6}$$

By doing this and choosing the input u as Kx, the following result is obtained.

$$\begin{aligned} V_f(x(t+1)) &= V_f(A_kx(t)) \\ &= \frac{1}{2}x^T(t+1)Px(t+1) \\ &= \frac{1}{2}x^T(t)A_k^TPA_kx(t) \\ &= \frac{1}{2}x^T(t)Px(t) - \frac{1}{2}x^T(t)Q_Kx(t) \\ &= V_f(x(t)) - \frac{1}{2}x^T(t)Qx(t) - \frac{1}{2}x^T(t)K^TRKx(t) \\ &= V_f(x(t)) - \frac{1}{2}x^T(t)Qx(t) - \frac{1}{2}u^T(t)Ru(t) \\ &= V_f(x(t)) - \ell(x(t), u(t)) \end{aligned} \tag{A-7}$$

[1]Please note that not all assumptions needed to prove asymptotic stability are explained. The ones stated are though the ones that in many cases are not met a-priori. Please check [26] for the remaining conditions, e.g., the state-solution and cost function must be continuous and the terminal constraint set compact.

This concludes the proof for condition 1. To ensure that $\forall x, u \in \mathbb{U}$ and $x(t+1) \in \mathbb{X}_f$, we have to guarantee a solution, where all constraints are met and end up in the final terminal invariant set. This is done by creating different terminal sets, whom are subjected to a set of constraints.

$$\begin{aligned} \mathbb{X}_f^{\mathbb{U}} &= \{x \in \mathbb{X} | A_{\mathbb{U}} K x \leq b_{\mathbb{U}}\} \\ \mathbb{X}_f^K &= \left\{x \in \mathbb{X} | A_{\mathbb{X}_f} A_K x \leq b_{\mathbb{X}_f}\right\} \end{aligned} \tag{A-8}$$

Here the combination between $A_{\mathbb{U}}$ and $b_{\mathbb{U}}$ describe the input constraints of the system and $A_{\mathbb{X}_f}$ and $b_{\mathbb{X}_f}$ characterize the initial terminal set. By taking the intersection of $\mathbb{X}_f$ and the terminal sets as described above we obtain the following final terminal set.

$$\mathbb{X}_f^{\text{fin}} := \mathbb{X}_f \cap \mathbb{X}_f^{\mathbb{U}} \cap \mathbb{X}_f^K \tag{A-9}$$

After creating the final terminal set it can be expressed as follows.

$$\mathbb{X}_f^{\text{fin}} = A_{\mathbb{X}_f^{\text{fin}}} x \leq b_{\mathbb{X}_f^{\text{fin}}} \tag{A-10}$$

here $A_{\mathbb{X}_f^{\text{fin}}}$ and $b_{\mathbb{X}_{\text{f}}^{\text{fin}}}$ is made up out of stacking the matrices and vectors that create the final constraint.

$$A_{\mathbb{X}_f^{\text{fin}}} = \begin{bmatrix} A_{\mathbb{X}_f} \\ A_{\mathbb{U}} K \\ A_{\mathbb{X}_f} A_K \end{bmatrix} \tag{A-11}$$

$$b_{\mathbb{X}_f^{\text{fin}}} = \begin{bmatrix} b_{\mathbb{X}_f} \\ b_{\mathbb{U}} \\ b_{\mathbb{X}_f} \end{bmatrix} \tag{A-12}$$

$$\mathbb{X}_f^{\text{fin}} = \begin{bmatrix} A_{\mathbb{X}_f} \\ A_{\mathbb{U}} K \\ A_{\mathbb{X}_f} A_K \end{bmatrix} x \leq \begin{bmatrix} b_{\mathbb{X}_f} \\ b_{\mathbb{U}} \\ b_{\mathbb{X}_f} \end{bmatrix} \tag{A-13}$$

For implementation purposes it is needed to create a conversion from state constraints to input constraints. This conversion is demonstrated below as an example for the terminal constraints.

$$\begin{aligned} A_{\mathbb{X}_f} x_N &\leq b_{\mathbb{X}_f} \\ A_{\mathbb{X}_f}(T x_0 + S u_N) &\leq b_{\mathbb{X}_f} \\ A_{\mathbb{X}_f} T x_0 + A S u_N &\leq b_{\mathbb{X}_f} \\ A_{\mathbb{X}_f} S u_N &\leq b_{\mathbb{X}_f} - A_{\mathbb{X}_f} T x_0 \end{aligned} \tag{A-14}$$

Clearly, this aggressively augmenting is not optimal in the sense that it always produces a relatively ”small” set. There are of course more elaborate ways to augment the terminal set differently, creating a larger augmented terminal set that still proves both. This can be of great difference in many applications since the MPC optimization could easily render infeasible by having a ”too small” terminal constraint set. Especially when a short horizon is desired (e.g. for computation time and complexity purpose.

Appendix B

Proofs for Sub-Optimal Cooperative Distributed Control for Vehicle Collision Avoidance

In chapter 4-4, 3 Lemmas are given describing properties of sub-optimal cooperative distributed MPC applied to a vehicle collision avoidance problem. First feasibility of the sub-optimal trajectories is shown by inductive reasoning (Lemma 4.1), next it is shown that the local cost functions of each subsystem converge to a fixed value (Lemma 4.2), but not to the Pareto optimal solution (Lemma 4.3).

B-1 Proof of Lemma 4.1

By assumption, initial guess $\tilde{\boldsymbol{u}}(0)$ is feasible. Since the input constraint sets $\mathcal{U}_i, \forall i \in \mathcal{V}$ are convex and since the cooperative control convex step of (4-16) with $p = 0$, in combination with compatibility constraint (4-15h), implies feasibility of $(\boldsymbol{v}_1^1, \boldsymbol{v}_2^1)$. Feasibility for $p > 1$ follows by induction.

B-2 Proof of Lemma 4.2

Since each local subsystem makes use of (mixed integer) linear constraints for collision avoidance (see section (3-3)) and has a linear vehicle model, $V_i(\mathbf{x}_i(0), \boldsymbol{v}_i^p)$ is convex. Hence, the plantwide objective, e.g., the sum of local cost functions is also convex. For every $p \geq 0$ the plantwide cost function satisfies the following:

$$\begin{aligned}
&V\left(x(0), \boldsymbol{v}^{p+1}\right) = V\left(x(0), w_1\left(\boldsymbol{v}_1^*, \boldsymbol{v}_2^p\right) + w_2\left(\boldsymbol{v}_1^p, \boldsymbol{v}_2^*\right)\right) \\
&\leq w_1 V\left(x(0), \left(\boldsymbol{v}_1^*, \boldsymbol{v}_2^p\right)\right) + w_2 V\left(x(0), \left(\boldsymbol{v}_1^p, \boldsymbol{v}_2^*\right)\right) \\
&\leq w_1 V\left(x(0), \left(\boldsymbol{v}_1^p, \boldsymbol{v}_2^p\right)\right) + w_2 V\left(x(0), \left(\boldsymbol{v}_1^p, \boldsymbol{v}_2^p\right)\right) \\
&\leq V\left(x(0), \boldsymbol{v}^p\right)
\end{aligned} \tag{B-1}$$

where $(\boldsymbol{v}_1^*, \boldsymbol{v}_2^p)$ denotes the optimal input sequence with respect to $\boldsymbol{v}_2^p$. Note that (B-1) shows the plantwide cost function for $N = 2$ vehicles, but the result applies to any number of vehicles. The first inequality, results from (4-16), the second from the convexity of the plantwide cost function, the third inequality follows from the optimality of $\boldsymbol{v}_i^*, \forall i \in \mathcal{V}$, the last inequality follows from the fact that the summed weight equals one, e.g. $\sum_{i\in\mathcal{V}} w_i = 1, \forall w_i > 0$. Since the cost is bounded, the plantwide cost converges.

B-3 Observation 4.1 Explained

Although the plantwide cost converges, it does not guarantee to converge to the plantwide Pareto optimal solution. This is shown by proving how the solution converges to the plantwide optimal solution, in case that the overall plantwide OCP is convex and only has simple bounds on the input as constraints. Using this as an contradicting example, it is shown why this is not possible in presence of non-convex (collision avoidance) constraints. This proof is adopted from [43].

Optimality proof in case of convex OCP

From Lemma 4.2, it is known that the algorithm converges, say to $\underline{V}$. Since V is quadratic and strongly convex, its sublevel sets $\text{lev}_{\leq a}(V)$ are compact and bounded for all a. In the case of only having bounds on the input, all iterates of the distributed optimization algorithm belong to the compact set $\text{lev}_{\leq V(v^0)}(V) \cap \mathbb{U}$. Since all sets are convex, the intersection of all sets is convex too. Hence, there is at least one accumulation point. Let $\overline{\boldsymbol{v}}$ be such accumulation point and choose any integer value $p \in \mathcal{I}$ for which $\{\boldsymbol{v}^p\}_{p\in\mathcal{I}}$ converges to $\overline{\boldsymbol{v}}$. By noticing that $V(x(0), \overline{\boldsymbol{v}}) = \underline{V}$ [1], and moreover that

$$\lim_{p\in\mathcal{I}, p\to\infty} V\left(x(0), \boldsymbol{v}^p\right) = \lim_{p\in\mathcal{I}, p\to\infty} V\left(x(0), \boldsymbol{v}^{p+1}\right) = \underline{V}. \tag{B-2}$$

By strict convexity of V and compactness of $\mathbb{U}_i, \forall i \in \mathcal{V}$, the minimizer of $V(x(0), \cdot)$ is attained at a unique point $\mathbf{u}^0 = \left(\mathbf{u}_1^0, \mathbf{u}_2^0\right)$ By taking limits in (B-1) as $p \to \infty$ for $p \in \mathcal{I}$, and using $w_1 > 0$, $w_2 > 0$ we deduce directly that

$$\lim_{p\in\mathcal{I}, p\to\infty} V\left(x(0), \left(\boldsymbol{v}_1^*\left(\boldsymbol{v}_2^p\right), \boldsymbol{v}_2^p\right)\right) = \underline{V} \tag{B-3a}$$

$$\lim_{p\in\mathcal{I}, p\to\infty} V\left(x(0), \left(\boldsymbol{v}_1^p, \boldsymbol{v}_2^*\left(\boldsymbol{v}_1^p\right)\right)\right) = \underline{V} \tag{B-3b}$$

We can conclude that $\overline{\boldsymbol{v}} = \mathbf{u}^0$ (since $\underline{V} = V\left(x(0), \mathbf{u}^0\right)$. Since $\overline{\boldsymbol{v}}$ was an arbitrary accumulation point of the sequence $\boldsymbol{v}^p$, and since this sequence is confined to a compact set, we conclude that the whole sequence converges to $\mathbf{u}^0$.

Optimality in case of non-convex OCP

In the case of an OCP that has to ensure vehicle collision avoidance, non-convexity arises as explained in chapter 3-3. In the case of sub-optimal cooperative distributed MPC, this unfortunately comes at the cost of not having the guarantee the distributed algorithm converges to the plantwide Pareto optimal solution.

[1]Proof by contradiction is given in [43, Appendix A],the interested reader is referred to here

In the case of a convex OCP with bounded and compact constraint sets, all iterates of the distributed algorithm belong to the intersection of these sets, which in its self is a convex, compact and bounded level set, e.g., $\text{lev}_{\leq V(v^0)}(V) \cap \mathbb{U}$. As explained above, since this level set is convex, there exists at least one accumulation point. In the case of a non-convex OCP due to non-convex collision avoidance constraints, this intersection of constrained sets is non-convex too, e.g. $\text{lev}_{\leq V(v^0)}(V) \cap \mathbb{U} \cap \mathcal{X} \cap \mathcal{C}$. Here, $\mathcal{X}$ and $\mathcal{C}$ represent the sets resulting from additional state constraints and (non-convex) collision avoidance constraints, respectively. Hence, such a accumulation point does not necessarily exist and convergence to the plantwide Pareto optimal solution is not guaranteed.

An empirical experiment is done to see if the distributed algorithm would approach the plantwide Pareto optimal solution. The sub-optimal cooperative distributed MPC controller is ran for different number of cooperative iterations, e.g. $p = 1, 2, 3, 5, 10, 20, 30, 50$. The objective values of the distributed controller are normalized with respect to the centralized solution, i.e., a Pareto optimal solution would correspond to an objective value of 1.

The experiment is conducted in the Unrestricted Highway Scenario for $N = 12$ vehicles. As can be seen in the figure below, the solutions converge to a constant value, but do not converge to the centralized, Pareto optimal solution.

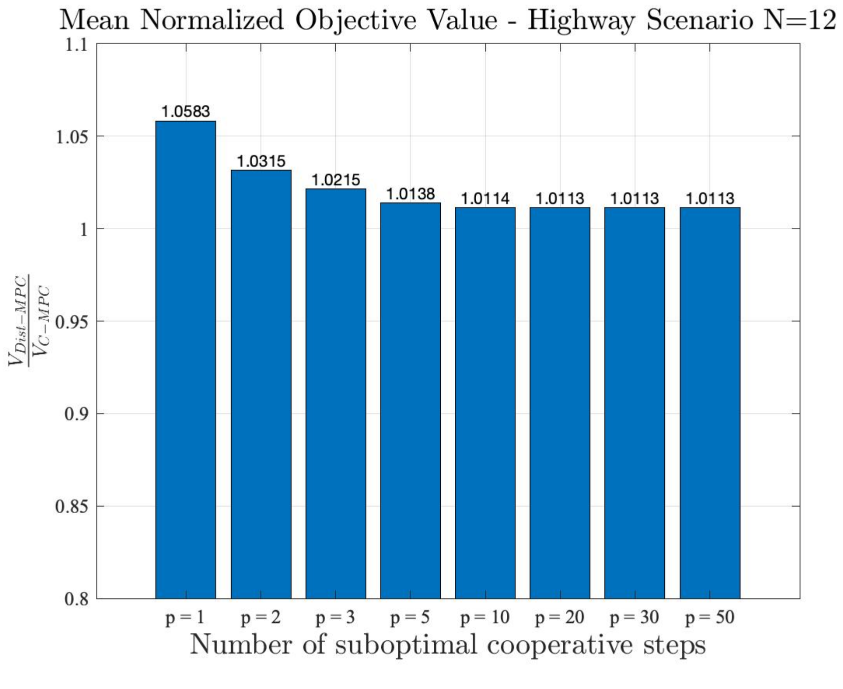


Figure B-1: Convergence and optimality experiment Coop. D-MPC

Appendix C

Trajectories Obstructed Highway Scenario

Since, in comparison to the first scenario, the trajectories of the distributed controllers for the second scenario are slightly different compared to the centralized controller, they are shown here.

C-1 Decentralized MPC

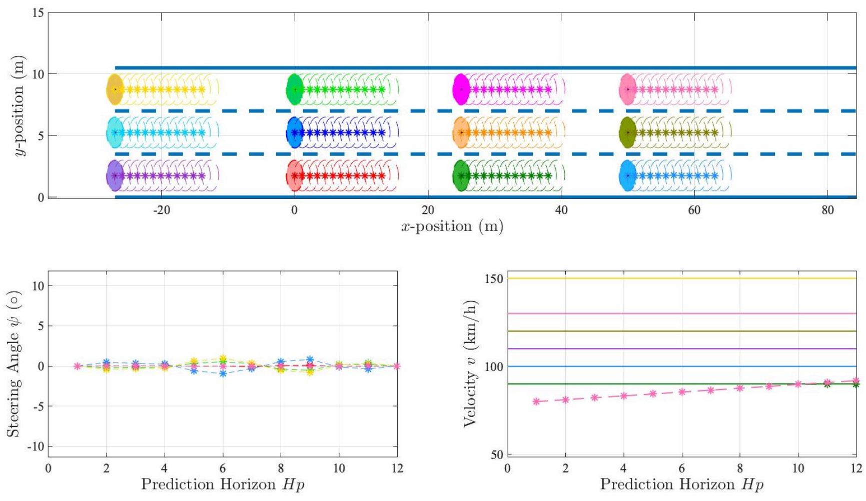


Figure C-1: MPC iteration = 1.

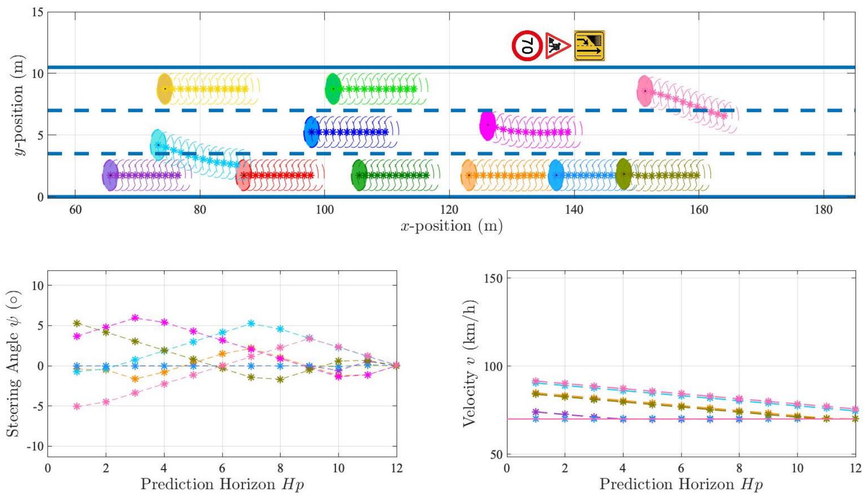


Figure C-2: MPC iteration = 70.

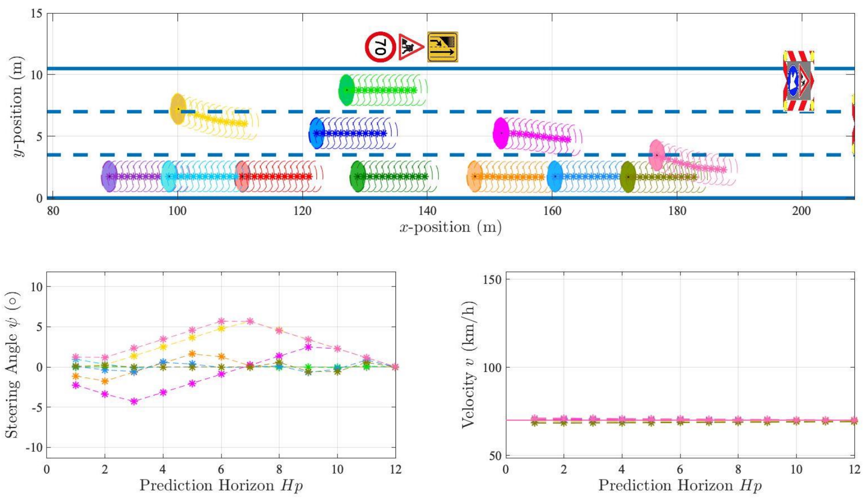


Figure C-3: MPC iteration = 94.

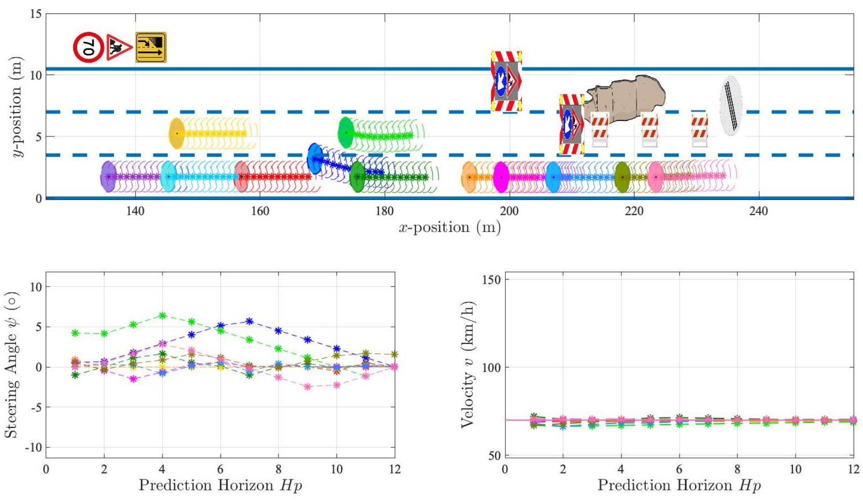


Figure C-4: MPC iteration = 142.

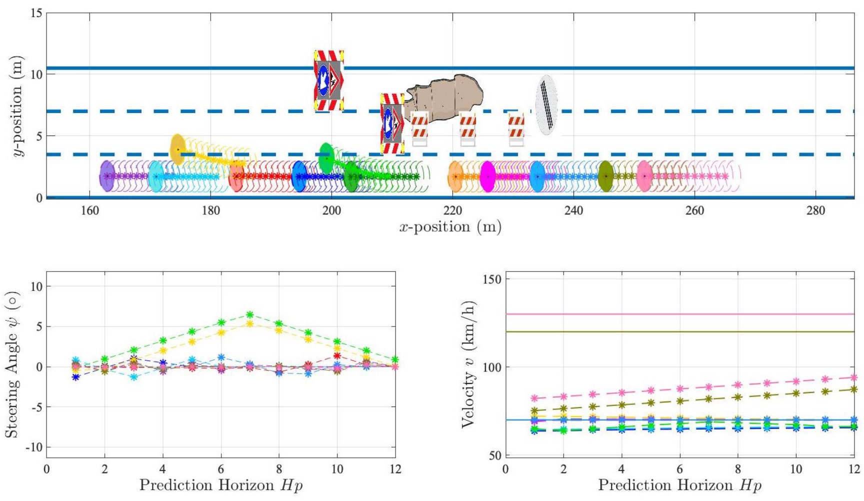


Figure C-5: MPC iteration = 170.

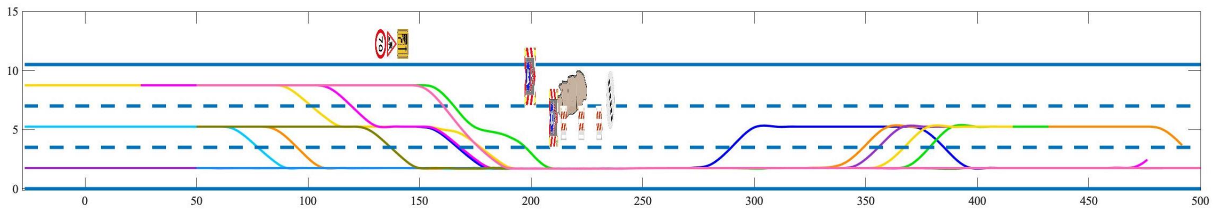


Figure C-6: Testing Scenario Obstructed Highway, Dec-MPC for N=12 vehicles, full trajectories.

C-2 Cooperative Distributed MPC

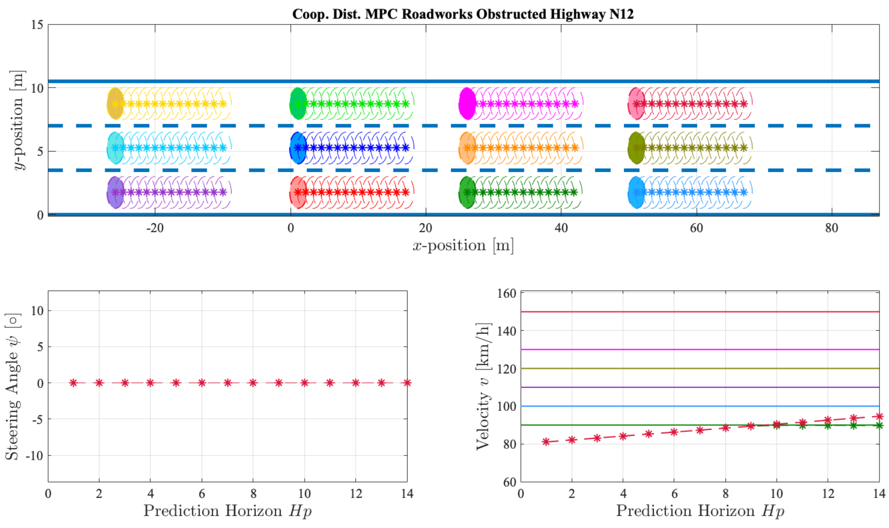


Figure C-7: MPC iteration = 1.

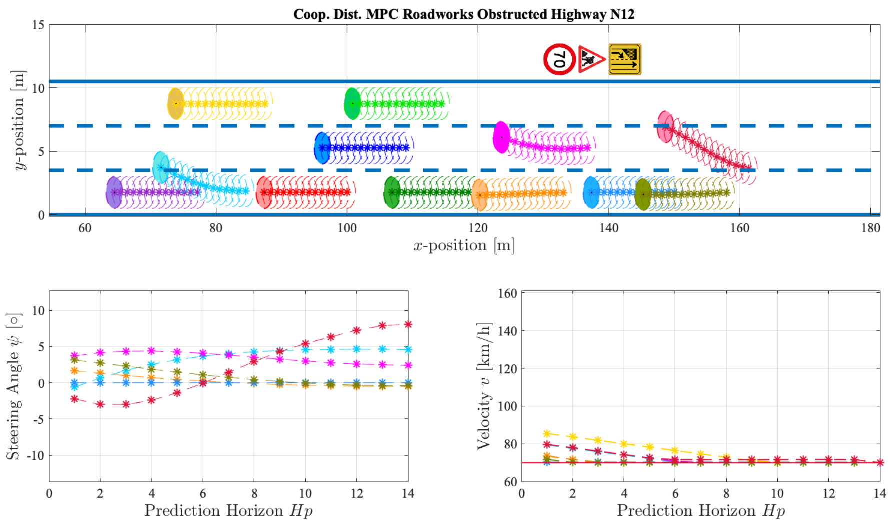


Figure C-8: MPC iteration = 70.

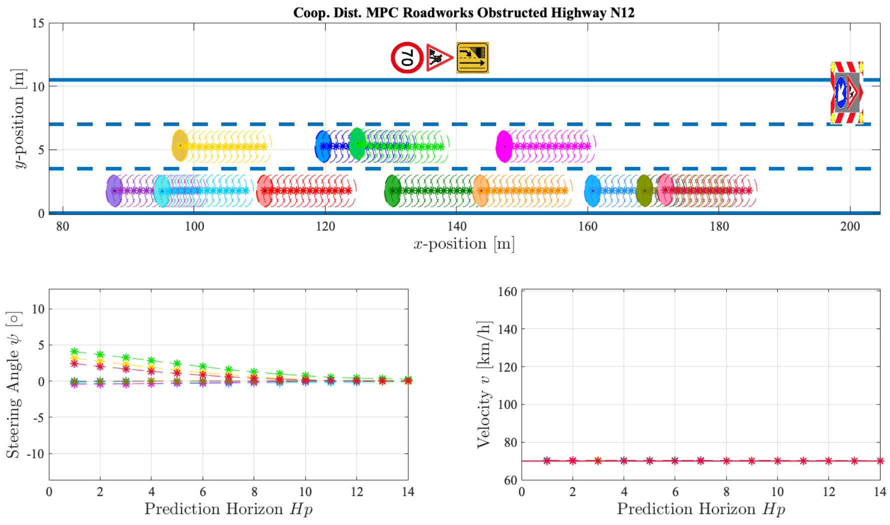


Figure C-9: MPC iteration = 94.

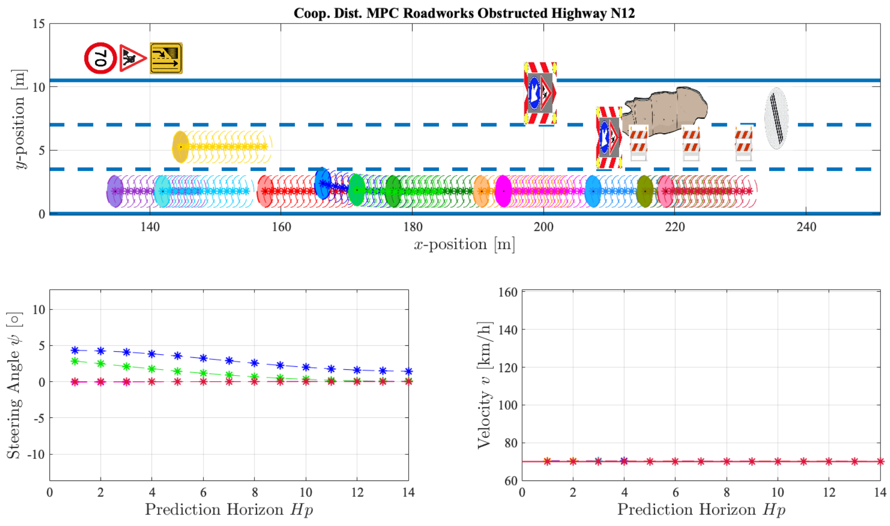


Figure C-10: MPC iteration = 142.

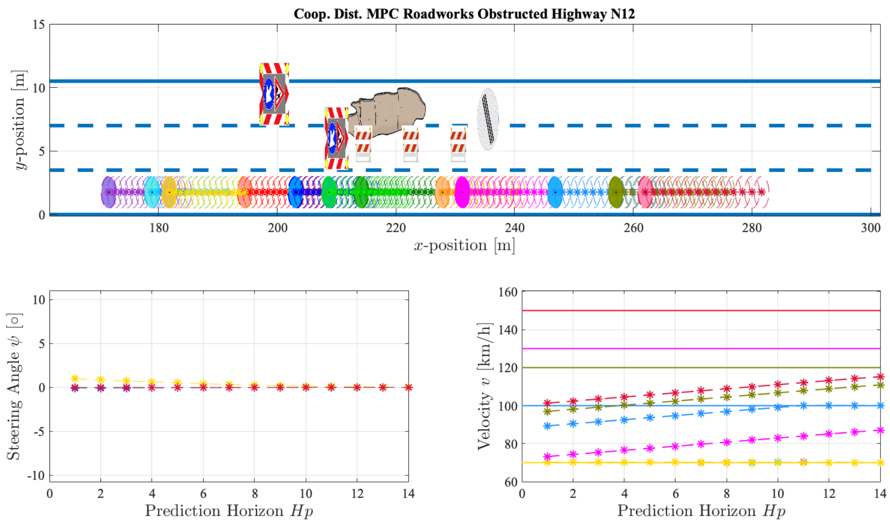


Figure C-11: MPC iteration = 180.

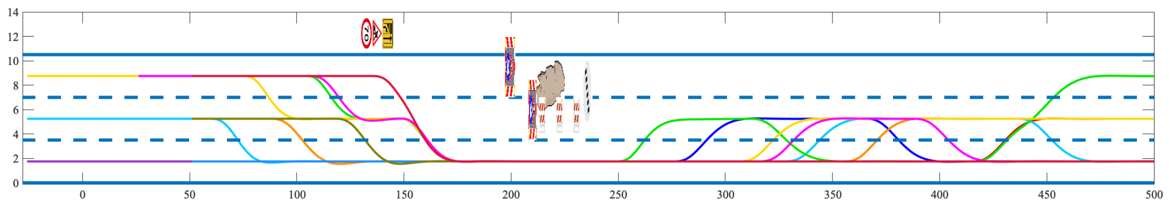


Figure C-12: Testing scenario Obstructed Highway, Coop. D-MPC for N=12 vehicles, full trajectories.

C-3 Non-Cooperative Distributed MPC

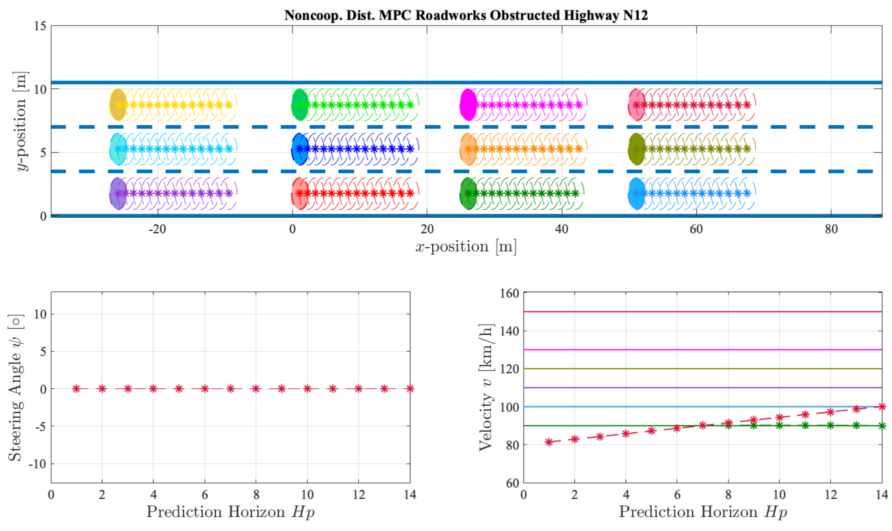


Figure C-13: MPC iteration = 1.

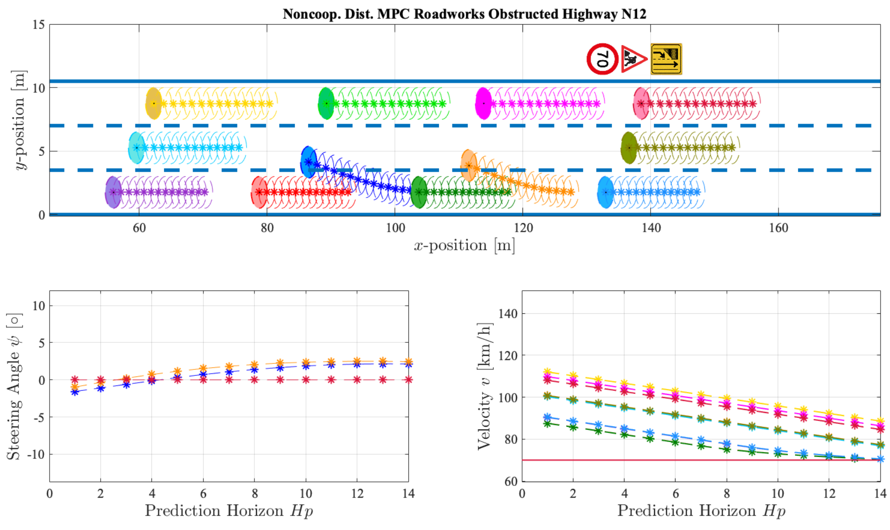


Figure C-14: MPC iteration = 70.

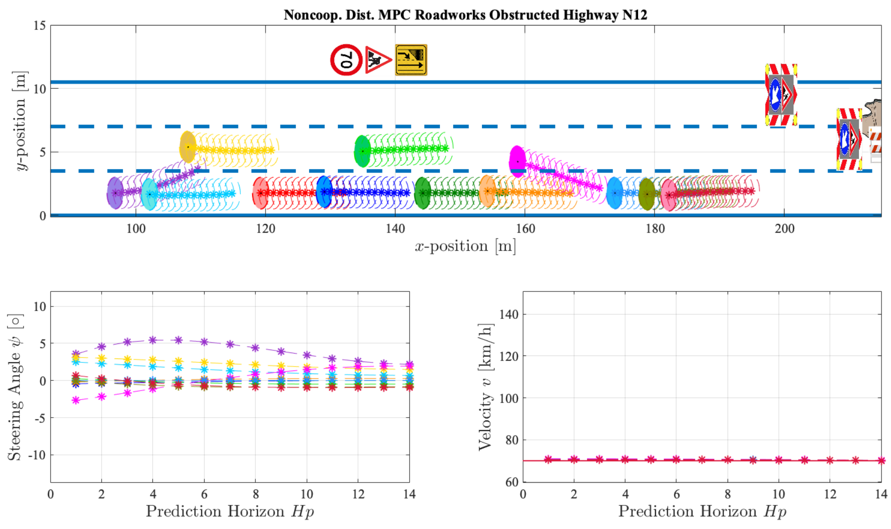


Figure C-15: MPC iteration = 94.

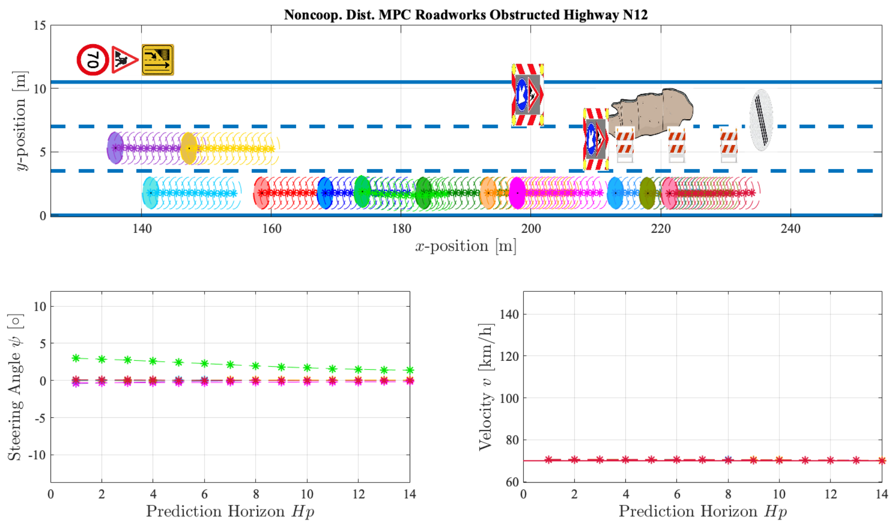


Figure C-16: MPC iteration = 142.

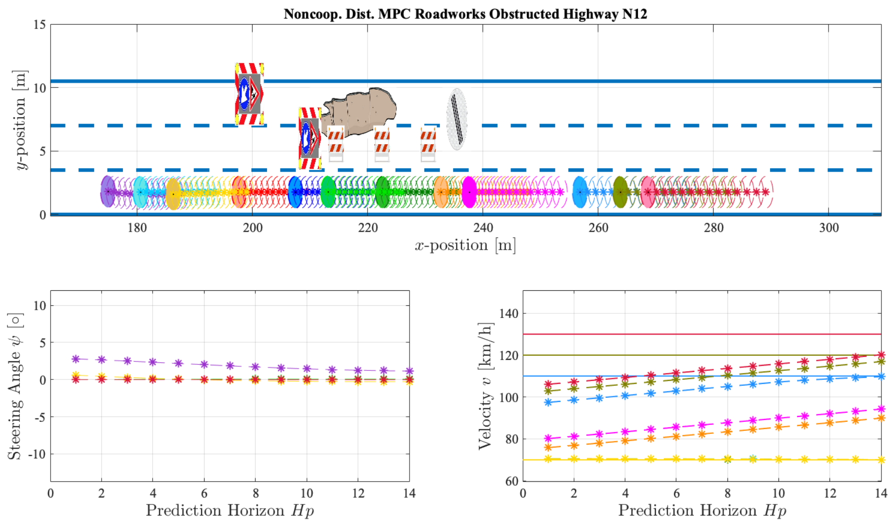


Figure C-17: MPC iteration = 180.

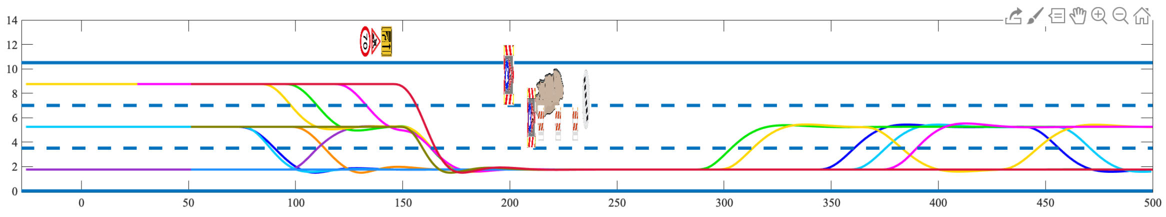


Figure C-18: Testing scenario Obstructed Highway, Non-Coop. D-MPC for N=12 vehicles, full trajectories.

Bibliography

[1] Bassam Alrifaee. *Networked Model Predictive Control for Vehicle Collision Avoidance.* PhD thesis, 05 2017.

[2] Bassam Alrifaee, Masoumeh Ghanbarpour, and Dirk Abel. Centralized non-convex model predictive control for cooperative collision avoidance of networked vehicles. 10 2014.

[3] Bassam Alrifaee, Janis Maczijewski, and Dirk Abel. Sequential convex programming mpc for dynamic vehicle collision avoidance. 08 2017.

[4] Joel Andersson, Joris Gillis, Greg Horn, James Rawlings, and Moritz Diehl. Casadi: a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11, 07 2018.

[5] J. Barreiro-Gomez, C. Ocampo-Martinez, and N. Quijano. Partitioning for large-scale systems: A sequential distributed mpc design **this work has been partially supported by the project deocs (ref. dpi2016-76493-c3-3-r). j. barreiro-gomez is partially supported by colciencias and agaur. *IFAC-PapersOnLine*, 50(1):8838 – 8843, 2017. 20th IFAC World Congress.

[6] S. Boyd. Sequential convex programming, 2015.

[7] Stephen Boyd and Lieven Vandenberghe. Semi definite programming relaxations of non-convex problems in control and combinatorial optimization, 1997.

[8] E Camponogara, Dong Jia, BH Krogh, and S Talukdar. Distributed model predictive control. *Control Systems Magazine, IEEE*, 22:44–52, 02 2002.

[9] Rafael P. Costa, Joao M. Lemos, Joao F. C. Mota, and Joao M. F. Xavier. D-admm based distributed mpc with input-output models*.

[10] R. Diestel. *Graph Theory.* Electronic library of mathematics. Springer, 2006.

[11] Minh Dang Doan, Moritz Diehl, Tamas Keviczky, and Bart De Schutter. A jacobi decomposition algorithm for distributed convex optimization in distributed model predictive control**support by the alexander von humboldt foundation, by eu via erc-highwind (259 166), itn-tempo (607 957), and itn-awesco (642 682) and by dfg via the project numerische methoden zur optimierungsbasierten regelung zyklischer prozesse is gratefully acknowledged. *IFAC-PapersOnLine*, 50(1):4905 – 4911, 2017. 20th IFAC World Congress.

[12] William Dunbar. Distributed receding horizon control of multiagent systems. 01 2004.

[13] William Dunbar and Richard Murray. Receding horizon control of multi-vehicle formations: A distributed implementation. volume 2, pages 1995 – 2002 Vol.2, 01 2005.

[14] M.G. Earl and R. D'Andrea. Iterative milp methods for vehicle control problems. volume 21, pages 4369 – 4374 Vol.4, 01 2005.

[15] Technische Universiteit Eindhoven. Smart mobility, 2019.

[16] Filippo Fabiani and Sergio Grammatico. A mixed-logical-dynamical model for automated driving on highways. pages 1011–1015, 12 2018.

[17] Filippo Fabiani and Sergio Grammatico. Multi-vehicle automated driving as a generalized mixed-integer potential game. *IEEE Transactions on Intelligent Transportation Systems*, PP:1–10, 03 2019.

[18] Francisco Facchinei and Christian Kanzow. Generalized nash equilibrium problems. *Annals of Operations Research*, 175(1):177–211, Mar 2010.

[19] Francisco Facchinei and Jong-Shi Pang. Nash equilibria: The variational approach. *Convex Optimization in Signal Processing and Communications*, 01 2009.

[20] Francisco Facchinei, Veronica Piccialli, and Marco Sciandrone. Decomposition algorithms for generalized potential games. *Computational Optimization and Applications*, 50(2):237–262, Oct 2011.

[21] Giacomo Como G. Giordano. Lecture notes in networked and distributed control, technische universiteit delft, April 2019.

[22] Saul I. Gass and Michael C. Fu, editors. *A* Algorithm*, pages 1–1. Springer US, Boston, MA, 2013.

[23] Saul I. Gass and Michael C. Fu, editors. *Dijkstra's Algorithm*, pages 428–428. Springer US, Boston, MA, 2013.

[24] LLC Gurobi Optimization. Gurobi optimizer reference manual, 2020.

[25] Peter Hart. Shakey: the world's first mobile, intelligent robot, 2015.

[26] D.Q. Mayne J.B. Rawlings and M. M. Diehl. Model predictive control: Theory, computation, and design 2nd edition, 2017.

[27] T. Keviczky, F. Borrelli, K. Fregene, D. Godbole, and G. J. Balas. Decentralized receding horizon control and coordination of autonomous vehicle formations. *IEEE Transactions on Control Systems Technology*, 16(1):19–33, Jan 2008.

[28] Steven M. Lavalle. Rapidly-exploring random trees: A new tool for path planning. Technical report, 1998.

[29] Kevin Leyton-Brown and Yoav Shoham. *Essentials of Game Theory : a Concise, Multidisciplinary Introduction.* Morgan Claypool,, 2008.

[30] J.M. Maestre. *Distributed Model Predictive Control Based on Game Theory.* PhD thesis, 10 2010.

[31] Matlab optimization toolbox, 2019a.

[32] Fatemeh Mohseni, Erik Frisk, Jan Åslund, and Lars Nielsen. Distributed model predictive control for highway maneuvers. *IFAC-PapersOnLine*, 50:8531–8536, 07 2017.

[33] R.B. Myerson. *Game theory: analysis of conflict.* Harvard University Press, 1991.

[34] Gabriele Pannocchia. *Distributed Model Predictive Control*, pages 301–308. Springer London, London, 2015.

[35] Anthony N Michel Panos J Antsaklis. *Linear Systems.* Springer Science Business Media, 2006.

[36] Pablo A. Parrilo and Sanjay Lall. Semidefinite programming relaxations and algebraic optimization in control. *European Journal of Control*, 9(2):307 – 321, 2003.

[37] Ionela Prodan, Sorin Olaru, Cristina STOICA MANIU, and Silviu Niculescu. Path following with collision avoidance and velocity constraints for multi-agent systems. *Annals of the University of Craiova, Automation, Computers, Electronics and Mechatronics*, 7:33–38, 01 2010.

[38] Xiangjun Qian, Sébastien Diemer, Jean Gregoire, Fabien Moutarde, Silvère Bonnabel, Ignacio Llatser, Andreas Festag, Katrin Sjöberg, Arnaud de La Fortelle, A. Martinoli, and Ali Marjovi. Network of automated vehicles: The autonet2030 vision. 09 2014.

[39] Arthur Richards and Jonathan How. Aircraft trajectory planning with collision avoidance using mixed integer linear programming. volume 3, pages 1936 – 1941 vol.3, 02 2002.

[40] Simone Sagratella. Algorithms for generalized potential games with mixed-integer variables. *Computational Optimization and Applications*, 68, 07 2017.

[41] Tom Schouwenaars, Bart De Moor, Eric Feron, and Jonathan How. Mixed integer programming for multi-vehicle path-planning. *European Control Conf.*, 01 2001.

[42] P. O. M. Scokaert, D. Q. Mayne, and J. B. Rawlings. Suboptimal model predictive control (feasibility implies stability. *IEEE Trans. Automat Control*, pages 648–654, 1999.

[43] Brett Stewart, Aswin Venkat, James Rawlings, Stephen Wright, and Gabriele Pannocchia. Cooperative distributed model predictive control. *Systems Control Letters*, 59:460–469, 08 2010.

[44] Brett T. Stewart, Stephen J. Wright, and James B. Rawlings. Cooperative distributed model predictive control for nonlinear systems. *Journal of Process Control*, 21(5):698 – 704, 2011. Special Issue on Hierarchical and Distributed Model Predictive Control.

[45] Felipe Valencia, Jairo Espinosa Oviedo, Bart De Schutter, and Kateřina Staňková. Feasible-cooperation distributed model predictive control scheme based on game theory . volume 18, 08 2011.

[46] Felipe Valencia, Julian Patiño, José D. López, and Jairo Espinosa. Game theory based distributed model predictive control for a hydro-power valley control. *IFAC Proceedings Volumes*, 46(13):538 – 544, 2013. 13th IFAC Symposium on Large Scale Complex Systems: Theory and Applications.

[47] Ruben Van Parys and Goele Pipeleers. Distributed model predictive formation control with inter-vehicle collision avoidance. 12 2017.

[48] H.P. Williams. Self-driving car, 2018.

[49] Joerg Wolf, P. Robinson, and J. Davies. Vector field path planning and control of an autonomous robot in a dynamic environment. 01 2004.

[50] Xue Yang, Jie Liu, Feng Zhao, and Nitin Vaidya. A vehicle-to-vehicle communication protocol for cooperative collision warning. pages 114–123, 01 2004.

[51] Jiang Zhao, Siyue Zhou, and Rui Zhou. Distributed time-constrained guidance using nonlinear model predictive control. *Nonlinear Dynamics*, 84:2016, 05 2016.

[52] Huarong Zheng, Rudy R. Negenborn, and Gabriel Lodewijks. Cooperative distributed collision avoidance based on admm for waterborne agvs. In Francesco Corman, Stefan Voß, and Rudy R. Negenborn, editors, *Computational Logistics*, pages 181–194, Cham, 2015. Springer International Publishing.

[53] Huarong Zheng, Rudy R. Negenborn, and Gabriel Lodewijks. Cooperative distributed collision avoidance based on admm for waterborne agvs. In Francesco Corman, Stefan Voß, and Rudy R. Negenborn, editors, *Computational Logistics*, pages 181–194, Cham, 2015. Springer International Publishing.

Glossary

List of Acronyms

NE	Nash Equilibrium
GNEP	Generalized Nash Equilibrium Problem
GNE	Generalized Nash Equilibrium
GPG	Generalized Potential Game
QP	Quadratic Programming
NLP	Non-Linear Programming
MPC	Model Predictive Control
C-MPC	Centralized Model Predictive Control
D-MPC	Distributed Model Predictive Control
CoG	Center of Gravity
MIP	Mixed Integer Programming
SCP	Sequential Convex Programming
SDPR	Semi-Definite Programming Relaxations
QCQP	Quadratically Constraint Quadratic Programming
OCP	Optimal Control Problem
ADMM	Alternating Direction Method of Multipliers
Coop. D-MPC	Cooperative Distributed MPC
Noncoop. D-MPC	Non-cooperative Distributed MPC
Dec. MPC	Decentralized MPC
DSRC	Dedicated Short Range Communication
V2V	Vehicle-to-Vehicle
LLC	Low Level Controller
MLC	Mid Level Controller
HLC	High Level Controller
LIDAR	LIght Detection And Ranging
RADAR	RAdio Detection And Ranging
ADAS	Advanced Driving Assistence Systems

List of Symbols

ϵ	Equilibrium approximation treshold
γ	Collision avoidance potential function weight
$\mathbb{R}^n$	Algebraic set with dimension n
$\mathcal{E}$	Set of edges, a.k.a. communication links
$\mathcal{G}$	Connected and undirected graph
$\mathcal{K}$	Constant
$\mathcal{N}_i$	Set of neighboring nodes of agent i
$\mathcal{O}$	Vehicle ordering
$\mathcal{S}$	Solution of a game
$\mathcal{U}$	Constrained set of the inputs of subsystem i
$\mathcal{V}$	Set of vertices/nodes, a.k.a. vehicles
$\mathcal{W}$	Weight matrix
$\mathcal{X}_i$	Constrained set of the states of agent,player,subsystem i
ω	Steering rate
ψ	Steering angle
σ	Forcing Function
τ	Undefined time instance
θ	Vehicle body angle in global frame
h	Sampling time
H_p	Prediction Horizon
H_u	Control Horizon
P	Potential Function of a game
V	Objective/Cost function