Thesis for the degree of Licentiate of Engineering

# Computationally efficient exact remodeling of optimization programs with applications to autonomous driving

Johan Karlsson


Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2019

Computationally efficient exact remodeling of optimization programs with applications to autonomous driving
Johan Karlsson



Licentiatavhandlingar vid Chalmers tekniska högskola
Technical report No. R006/2019
ISSN 1403-266X

Department of Electrical Engineering
Chalmers University of Technology
SE–412 96 Göteborg, Sweden
Telephone + 46 (0) 31 – 772 1000

Typeset by the author using LaTeX.

Printed by Chalmers Reproservice
Göteborg, Sweden 2019

*to my family*

# Abstract

The future of autonomous vehicles is rapidly approaching and the published and available research, both from vehicle manufacturers and universities, is abundant. This new technology promises less pollution, lower accident rates, decreased congestion and the possibility to relax or work while a vehicle takes you where you need to go.

In this thesis we use nonlinear model predictive control to control autonomous vehicles overtaking on highways and driving through intersections. One of the main disadvantages of model predictive control is that the optimal control problems can be computationally expensive to solve. This could certainly be the case for the exact temporal formulation of the intersection and highway problem since the modeling of for both applications include binary decisions; and thus, have mixed integer optimization programs as their optimal control problems. To decrease the computational complexity of these optimal control problems this thesis introduces a novel reformulation technique for optimal control problems which removes the integer decisions present due to the collision constraints; which results in a continuous, nonlinear control problem for both applications. The remodeling technique involves changing the independent variable from travel time to traveled distance, introducing travel time and inverse velocity as states and lastly by introducing new input signals. After the remodeling, the continuous, nonlinear optimal control problems are solved using sequential quadratic programming. Further, it is shown that the introduced remodeling technique guarantees that the subproblems of the sequential quadratic programming scheme provides feasible solutions to the original nonlinear program being solved; for both the intersection and overtaking problem. This makes it possible to stop the sequential quadratic programming scheme prematurely and still have access to a solution that is feasible in the nonlinear program; provided, of course, that the subproblems themselves are feasible.

**Keywords:** Model predictive control, Convex optimization, Autonomous driving, Real-time implementation.

# Acknowledgments

I would like to thank my main supervisor prof. Jonas Sjöberg and my co. supervisor Docent Nikolce Murgovski for their help and support throughout my work. They have continuously provided constructive feedback and fruitful discussions which has been immensely helpful.

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

Johan Karlsson
Göteborg, April 2019

# List of Publications

This thesis is based on the following appended papers:

**Paper 1. Johan Karlsson**, Jonas Sjöberg, Nikolce Murgovski, Lowisa Hanning, Susan Luu, Vanessa Olsson, and Alexander Rasch, *Intersection crossing with reduced number of conflicts*, In proceedings of the 21st International Conference on Intelligent Transportation Systems (ITSC), Maui, USA, Nov. 2018.

**Paper 2. Johan Karlsson**, Nikolce Murgovski, and Jonas Sjöberg, *Computationally efficient autonomous overtaking of vehicles*, Submitted to IEEE Journal of Intelligent Transportation Systems in April 2019.

Other relevant publications by the author:

**Johan Karlsson**, Nikolce Murgovski, and Jonas Sjöberg, *Temporal vs. spatial formulation of autonomous overtaking algorithms*, In proceedings of the 19th International Conference on Intelligent Transportation Systems (ITSC), Rio de Janeiro, Brazil, 2016.

**Johan Karlsson**, Nikolce Murgovski, and Jonas Sjöberg, *Comparison between mixed-integer and second order cone programming for autonomous overtaking*, In proceedings of the European Control Conference (ECC), Limassol, Cyprus, 2018.

**Johan Karlsson**, Nikolce Murgovski, and Jonas Sjöberg, *Optimal trajectory planning and decision making in lane change maneuvers near a highway exit*, Accepted for the European Control Conference (ECC), Naples, Italy, 2019.

# List of Acronyms

| | | |
|---|---|---|
| CoG | – | Center of Gravity |
| MIP | – | Mixed Integer Program |
| MINLP | – | Mixed Integer Nonlinar Program |
| MIQP | – | Mixed Integer Quadratic Program |
| MPC | – | Model Predictive Control |
| NLP | – | Nonlinear Program |
| QP | – | Quadratic Program |
| RTI | – | Real-Time Implementation |
| SQP | – | Sequential Quadratic Program |

# Contents

# Part I

# Introductory chapters

# Chapter 1

# Introduction

There are several instances of what seem to be autonomous objects throughout myth and religion. For instance, Odin, the head of the asa gods in Norse mythology, wielded a spear called Gugnir, which was given to him by Loki with the assertion that, "It is a unique spear in that when aimed and thrown it never misses its mark". Further, it is asserted, also by Loki, that the ship Skidbladnir "... never fail to find a breeze, and will sail as well over land as it does over the sea.", [1]. The idea of autonomy is very old and was, as many other things, attributed to magic. A more modern approach to autonomy and automated machines emerged in the 1920s with the first mention of the word robot in Karel Čapek's play Rossum's Universal Robots (however, similarly to autonomy the concept of Robots had been present in ancient myth and mythology for centuries), [2]. One of the first experminents with automated vehicles was also carried out in the twenties when an automated driving system was tested in Milwaukee, [3].

While autonomous driving has been around for almost a century it is in the last few decades that the concept has garnered significant attention from researchers, vehicle manufacturers and media outlets. Some of the notable and highly reported events were the US national automated highway system consortium's proof of technical feasibility of automated driving made in 1997, [4], the DARPA grand challenge in 2005, [5], and the DARPA urban challenge in 2007, [6].

One can ask what an autonomous or self-driving vehicle even is. Is it enough for a vehicle to be not much more than an automated (horizontal) elevator which transports human or goods between two fixed points in a static environment or should it be like the ship Skidbladnir and be able to move anywhere at any time adapting to a dynamic world? In an attempt to answer this question, the SAE international [7] introduced six different levels of autonomy in 2014

**Level 0:** No automation: Manual driving.

**Level 1:** Driver Assistance: Adaptive cruise control, Automatic braking.

**Level 2:** Partial automation.

**Level 3:** Conditional automation: traffic jam chauffeur.

**Level 4:** High automation: Local driverless taxi.

**Level 5:** Complete automation: Autonomous during all conditions.

On Levels 1-2 it is assumed, even when driver support features are engaged, that there is a human who is ready to take over at any time; if necessary. On Level 3 the human driver should be prepared to, at any time, take over the wheel, but should only have to do so when it is requested by the automated feature. Levels 4-5 are truly autonomous in the sense that no human driver is needed.

Levels 1 and 2 are standard in many vehicles today through features such as adaptive cruise control and automated lane keeping. However, most of the predicted benefits, such as stress release for drivers, reduction of pollution, accidents and traffic congestion, require an autonomy of level 4 or 5, [8]. Many automotive companies are working hard to bring vehicles to this new level of automation. For example, BMWs senior vice president say that BMW are on the track to release a vehicle with Level 3 autonomy (or higher) in 2021, Volvo CEO predicts Volvo will have a self-driving car for highways by the same year and Renault-Nissan predicts they will have a driverless car ready for 2025. This leap, from Level 1-2 to 3-5, is significant since the level of autonomy needed is heavily reliant on accurate real-time data collection to describe the surroundings. The data can be gathered via sensors (e.g., cameras and Lidar) attached to the vehicle, or outside sources such as infrastructure. Since no infrastructure to support autonomous vehicles exist today, the main approach used is to mount sensors on the vehicle. In this thesis however, we are not concerned with acquiring and fusing sensor data. Instead, we will assume that this task has already been carried out. The thesis focus on control of the autonomous vehicle given this data.

There are many interesting control problems related to autonomous driving. Some of the most studied problems include platooning, highway driving, driving through intersections and parking. At the higher levels of autonomy, the traffic situations that can occur are so diverse that it will probably not be possible to use one single controller for all situations. Instead, the autonomous vehicle will be controlled via a hierarchical structure which will be capable of making decisions, generate trajectories and detect/avoid danger. This makes it relevant to address isolated control problems such as the ones mentioned above, even in the context of complete automation. There are many different methods that can be used to attack these isolated control problems. For example, multi agent systems, reinforcement learning [9], stochastic optimization and fuzzy control [10].

Some of the isolated control problems can be viewed as collision avoidance problems, and are often addressed using grid/graph search problems or (non)linear model predictive control (MPC), [11]. MPC approaches have been shown increased attention due to their ability to systematically handle system constraints and nonlinearities. In practice, MPC makes it simple to mathematically describe safety and comfort requirements via formulation of optimization constraints in the optimal control problem. In this thesis we will take a closer look at the optimal control problem when applying MPC to two of the most commonly studied problems in autonomous

vehicles research; namely overtaking on a highway and coordination of vehicles driving through an intersection.

In autonomous overtaking on a highway the automated vehicle, often referred to as the ego or host vehicle, is approaching a slower moving leading vehicle from behind. The goal is to overtake the leading vehicle via a double lane change, thus returning to the initial lane at the end of the manoeuvre. This has to be done while: avoiding collision with the leading vehicle; avoiding collision with surrounding traffic; staying within the road boundaries. These demands can be incorporated into the optimal control problem via constraints on the position of the ego vehicle. Graphically, the position constraints can be depicted as in Figure 1.1. To avoid collision with the leading vehicle a rectangular zone, refered to as the critical zone, is placed around the leading vehicle into which it is forbidden for the ego vehicle to enter. For other surrounding vehicles, ramp barriers are used to avoid collision. In Figure 1.1 it seems like the collision avoidance can easily be modeled using continuous, linear constraints. However, this is not the case when time progress since the velocity of the ego vehicle is not known prior to solving the optimal control problem. This introduce two binary decisions into the optimal control problem; at what time should the ego vehicle reach the beginning of the critical zone and at what time should it reach the end of the critical zone. Thus, the optimal control program for the overtaking problem becomes a mixed integer program (MIP).

In the problem of coordinating vehicles driving through an intersection given a predefined order, the goal for a centralized controller is to guide vehicles through an intersection along predefined paths while avoiding collision between the vehicles. In a similar way to the overtaking program the collision avoidance can be modeled by introducing restrictions on the position of the vehicles. However, in this case this cannot be done by completely prohibiting the vehicles from entering a certain area; since they all need to travel through the intersection. Instead critical zones are introduced for each vehicle pair which have intersecting paths, as illustrated by the grey areas in Fig. 1.2. Within these critical zones, collision is possible and therefore collision avoidance must be applied; either by allowing only one vehicle in a particular zone at the time; or by ensuring their distance within the zones are large enough to avoid collision. In contrast to the overtaking problem, the zones in which collision is possible is not attached to any moving vehicles, and are therefore fixed. The velocities of the controlled vehicles are, however, still unknown and therefore, just as in the case of the overtaking problem it is necessary to include binary choices into the program. This time there are two or four binary choices per vehicle pair with intersecting paths; when the vehicles enter the critical set and/or when they exit that same critical set.

As was illustrated in the two previous paragraphs the optimal control problem of the MPC might be a complicated optimization program. This highlights the main downside of MPC (and nonlinear MPC in particular); namely that it can be computationally expensive. This might limit its use as a real-time implementable algorithm. Thus, it is important to model the optimal control problem carefully such that it is as computationally tractable as possible. This careful modeling of the optimal control problem is the main focus of this thesis.

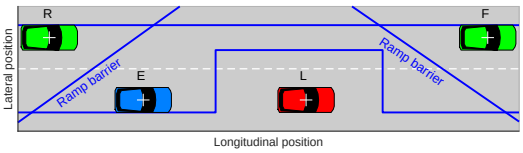



Figure 1.1: Overtaking scenario where an ego vehicle (E) is overtaking a leading vehicle (E) in the presence of two vehicles, (F) and (R), in the adjacent lane.

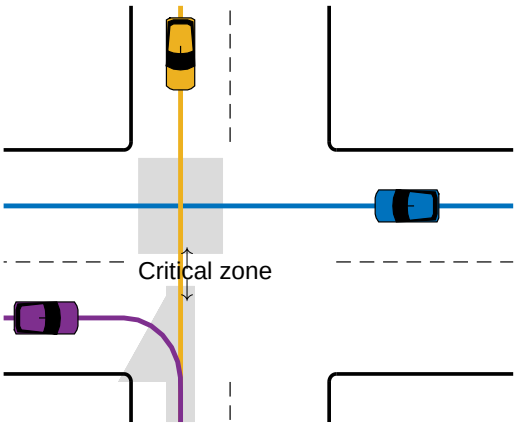



Figure 1.2: Intersection scenario where three vehicles are approaching an intersection. The lines of each vehicle display their predefined paths and the grey areas illustrate the critical zones within which collision between vehicles are possible.

Before addressing the problem of the binary decisions in the optimal control problems for the overtaking and intersection problems, a choice of how to describe the controlled vehicles need to be made. There is an abundance of vehicle models present in the literature. Differences between the vehicle model includes dynamical vs kinetic models, linear vs nonlinear vehicle models and the number of wheels modeled. In this thesis we will use one of the simplest models possible, a dynamical, linear point mass model.

Some common vehicle model used in the literature are: the four-wheels model, the two-wheels model, the one-wheel model and the point mass model; where the vehicle models are listed in order from most to least complex. Naturally, choosing a less complex model will reduce the complexity of the optimal control problem but also affect the accuraccy of the vehicle model. To this end, we will study under which assumptions it is feasible to apply the mentioned vehicle models; in particular in application to the overtaking and intersection problems.

We will show that the optimal control problem of the overtaking and intersection problems can be simplified significantly by application of a novel remodeling technique. The remodeling can be divided into three steps: 1. switching the roles of the travel time and traveled distance; 2. introducing inverse velocity as a state; 3. introducing new control signals. The purpose of step 1 is to eliminate the binary modes of the overtaking and intersection problems. As we shall see exchanging time for velocity achieves this, since the decisions that are binary when considering position as an optimization variable, is actually linear when considering time as an optimization variable instead. However, assuming that a linear point mass model has been used

to describe the controlled vehicles, this change of variables introduces nonlinearities into the dynamical model describing the vehicles. This is remedied in steps 2-3 in which the vehicle model is transformed, through exact remodeling, into an alternative linear vehicle model by the introduction of new states and controls. The result of this remodeling is that the optimal control problems, for both applications, are continuous nonlinear programs. It is then discussed how these nonlinear programs can be solved using the common sequential quadratic programming scheme, [12], [13].

## Thesis outline and main contribution

The main contributions of this thesis are as follows

**Contribution 1:** New remodeling scheme for computationally efficient MPC for vehicular scenarios with intersection paths.

**Contribution 2:** Application of this remodeling scheme to intersection control and highway driving.

**Contribution 3:** A heuristic that reduces the number of crossing orders that needs to be considered when determining the optimal order for vehicles to cross an intersection.

The thesis is divided into the typical two parts. In the first part, context and explanation is given for the papers that are appended in part 2.
The thesis has the following structure

**Chapter 2**

This chapter contains introduced the linear point mass model which is used to describe the vehicle dynamics throughout the intersection. Further, it contains a mixed integer formulation of the intersection program, which will later be used to showcase the applicability of the remodeling scheme introduced in Chapter 3. Lastly, the mathematical methods for solving optimziation programs that are used throughout the Kappa is introduced; sequential quadratic programming and real-time iterations.

**Chapter 3**

A remodeling scheme is presented which reduces the computational complexity of the optimal control problems of the MPC. The remodeling scheme includes change of reference frame and careful choices of state and control constraints. The remodeling is introduced step-wise; starting with the simple case of one vehicle being controlled longitudinally, followed by multiple vehicles being controlled longitudinally to vehicles being controlled both longitudinally and laterally. Along the way, the remodeling scheme will be applied to the intersection and highway driving problem

to showcase its applicability.

**Chapter 4**

In Chapter 3, the problem of longitudinally controlling vehicles traveling through an intersection, in a given order, was addressed. In this chapter we briefly address the problem of determining the crossing order, by iteratively solving the optimal control problem introduced in Chapter 3. Further, a heuristic is introduced which removes redundant crossing orders.

**Chapter 5**

This chapter contains a short summary of each of the appended papers.

**Chapter 6**

The final chapter of the Kappa outline directions for future work.

# Chapter 2

# Background

The scope of this chapter is to give a brief background to concepts used throughout the thesis and also to provide context and motivation to assumptions and arguments made in both the introductory chapters and the appended papers. In Section 2.2 the optimal control problem of coordinating vehicles driving through an intersection in a predefined order is formulated as an MIP; using the point mass model to describe the dynamics of the controlled vehicles. This problem will then be remodeled into an NLP, using a novel remodeling technique presented in Chapter 3.

Section 2.3 introduces the notation of mixed integer nonlinear programming and nonlinear programming, and it is discussed why, in general, a nonlinear program is preferable over a mixed integer one; in terms of solver efficiency.

Finally, in Section 2.4, it is discussed how continuous nonlinear programs (NLP) can be solved using sequential quadratic programming (SQP) and when this iterative procedure produces feasible point to the original NLP in each iteration.

## 2.1 Vehicle dynamics

In this thesis we will use a simple linear point mass model for the vehicle dynamics. This model given by the equations of motion

$$m\ddot{x} = F_{\mathrm{x}} - F_{\mathrm{z}} \sin\theta - F_{\mathrm{d}}(\dot{x}), \tag{2.1a}$$
$$m\ddot{y} = F_{\mathrm{y}}, \tag{2.1b}$$
$$v_{\mathrm{x}} = \dot{x}, \tag{2.1c}$$
$$v_{\mathrm{y}} = \dot{y}, \tag{2.1d}$$
$$F_{\mathrm{z}} = mg. \tag{2.1e}$$

where the notation is given in Table 2.1.

However, this point mass model is holonomic since the longitudinal and lateral dynamics are independent of each other. Since, vehicles are wellknown to be non-holonomic, i.e., its position in a cartesian coordinate system is determined by the followed path, we recapture this property by restricting the slip angle of the vehicle $\beta = \arctan(\dot{y}/\dot{x})$ and then introducing the inequality

$$\dot{y} \in [s_{\min}, s_{\max}]\dot{x}, \tag{2.2}$$

Table 2.1: Nomenclature for vehicle modeling. The entry 1 in the unit column means that the variable is unitless.

| Symbol | Description | Units |
|---|---|---|
| $x, y, z$ | Longitudinal, lateral and vertical coordinates | m |
| $v$ | Linear velocity | $\mathrm{ms}^{-1}$ |
| $m$ | Vehicle mass | kg |
| $F$ | Force | N |
| $\theta$ | Angle of road inclination | rad |

where $s_{\min} = -\arctan(\beta)$ and $s_{\max} = \arctan(\beta)$, as was suggested in [14].

The point mass model can be simplified even further, by assuming that the slope of the road is zero, $\theta = 0$, and assuming that the drag is zero, i.e., $F_{\mathrm{d}}(\dot{x}) = 0$. These simplifications lead to the linear point mass model

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}, \tag{2.3a}$$
$$\dot{y} \in [s_{\min}, s_{\max}]\dot{x}, \tag{2.3b}$$

where the state and input vectors are

$$\mathbf{x}(t) = [x,\ v_{\mathrm{x}},\ y,\ v_{\mathrm{y}}]^T,\ \mathbf{u}(t) = [F_{\mathrm{x}},\ F_{\mathrm{y}}]^T$$

and the matrices are given by

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix},\ B = \begin{bmatrix} 0 & 0 \\ \frac{1}{m} & 0 \\ 0 & 0 \\ 0 & \frac{1}{m} \end{bmatrix}. \tag{2.4}$$

In the next section this point mass model is used when modeling the problem of coordinating vehicles driving through an intersection.

## 2.2 Intersection problem

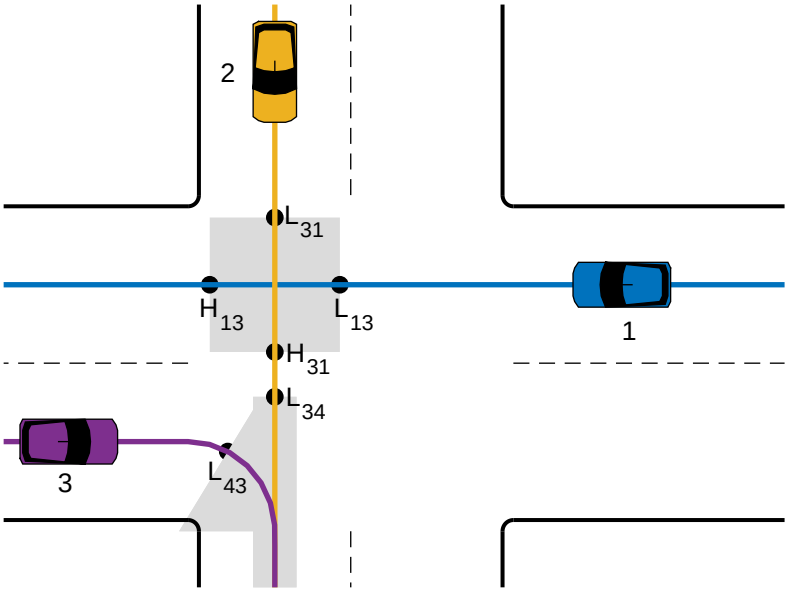



Figure 2.1: Illustration of a scenario where several autonomous vehicles approach an intersection, defined by a range of positions over a predefined paths. Collision avoidance is modeled via critical sets, here depicted in gray.

Consider three vehicles approaching a four-way intersection as in Fig. 2.1. Each vehicle, $i \in \{1, 2, 3\}$, plan on passing through the intersection along its predefined path, $p_i$. In this section, the crossing order of the vehicles is assumed to be fixed to $\mathcal{O} = [1,\ 2,\ 3]$.

The roads connecting to the area where the roads meet in Fig. 2.1, from here on referred to as the area of intersections, are called legs. The leg from which a vehicle approaches an intersection is the entry leg of that vehicle. For instance, vehicle 1 in Fig. 2.1 has entry leg 1, while vehicle 2 has entry leg 2. Further, a departure leg is the leg by which a vehicle leaves the area of intersection, e.g. vehicle 2 and 3 have the same departure leg, namely leg 4.

For each vehicle of the three vehicles, it is assumed that

**Assumption 1** a path is given and known;

**Assumption 2** the assigned path is perfectly followed;

**Assumption 3** the acceleration along the path can be varied;

**Assumption 4** its clock is synchronized with the other vehicles and is located within a certain control radius centered at the intersection.

### 2.2.1 Longitudinal dynamics

Let $\mathbf{x}_i(t) = [p_i(t), v_i(t)]^T$ denote the state vector of vehicle $i$, consisting of position $p_i(t)$ and velocity $v_i(t) = \dot{p}_i(t)$ along its path. A full measurement of the state $\mathbf{x}_i(t)$

is available at all times. The dynamical model for each vehicle is then given by the longitudinal part of the point mass model (2.3a), namely

$$\dot{\mathbf{x}}_i(t) = A\mathbf{x}_i(t) + Bu_i(t),$$

with

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix},\ B = \begin{bmatrix} 0 \\ \frac{1}{m_i} \end{bmatrix}, \tag{2.5}$$

where the longitudinal force is chosen as the control signal, i.e., $u_i(t) = F_i(t) = \ddot{p}_i(t)/m_i$, where $m_i$ is the mass of the $i$th vehicle.

### 2.2.2 State and control constraints

Let

$$\mathbf{x}_{i\min}(t) = [0, v_{i\min}(t)]^T,\ \mathbf{x}_{i\max}(t) = [\text{free}, v_{i\max}(t)]^T,$$
$$u_{i\min}(t) = F_{i\text{xmin}}(t) = \frac{a_{i\text{xmin}}(t)}{m_i},\ u_{i\max}(t) = F_{i\text{xmax}}(t) = \frac{a_{i\text{xmax}}(t)}{m_i},$$

where $v_{i\min}(t)$, $v_{i\max}(t)$, $a_{i\min}(t), a_{i\max}(t)$, with $0 < v_{i\min} \leq v_{i\max}(t)$, $a_{i\min} < 0$, $a_{i\max}(t) \geq 0$, for all $t \in [0, t_{i\text{f}}]$, denote the minimum and maximum speed and acceleration limits, respectively. Each vehicle $i$ is subject to state and control constraints

$$\mathbf{x}_i(t) = [\mathbf{x}_{i\min}(t), \mathbf{x}_{i\max}(t)],\ \forall i \in \mathcal{N},$$
$$u_i(t) = [u_{i\min}(t), u_{i\max}(t)],\ \forall i \in \mathcal{N},$$

where each inequality is imposed for all time instances $t \in [0, t_{i\text{f}}]$, until vehicle $i$ reaches its final destination $p_{i\text{f}}$. The final time, $t_{i\text{f}}$ is free and the largest of these times will be the time it takes to carry out the full solution.

### 2.2.3 Critical set

For each vehicle, $i \in \{1, 2, 3\}$ let $\mathcal{C}_i(j)$ denote the set of all positions along the path where collision with vehicle $j \neq i$, is possible. Studying Fig. 2.1 it is seen that the sets are given by

$$\mathcal{C}_1(2) = \{p_1(t) \in [0, p_{1\text{f}}] : p_1(t) \in [L_{1,2}, H_{1,2}]\}$$
$$\mathcal{C}_1(3) = \emptyset,$$
$$\mathcal{C}_2(1) = \{p_2(t) \in [0, p_{2\text{f}}] : p_2(t) \in [L_{2,1}, H_{2,1}]\},$$
$$\mathcal{C}_2(3) = \{p_2(t) \in [0, p_{3\text{f}}] : p_2(t) \in [L_{2,3}, p_{2\text{f}}]\}$$
$$\mathcal{C}_3(1) = \emptyset,$$
$$\mathcal{C}_3(2) = \{p_3(t) \in [0, p_{3\text{f}}] : p_3(t) \in [L_{3,2}, H_{3,2}]\},$$

where the critical sets of vehicles 1 and 3 are empty since their paths do not intersect within the intersection. The parameters $L_{i,j}$ and $H_{i,j}$ are time-invariant, their values depend on the geometry of the workspace and the relation between the paths of vehicles $i$ and $j$.

### 2.2.4 Occupancy interval

For each critical set there is a corresponding occupancy interval, which contains the times corresponding to the positions of the critical set, i.e., it is the set of time instances when vehicle $i$ resides within the critical set $\mathcal{C}_i(j)$:

$$\begin{aligned}
&\mathcal{G}_1(2) = \{t \in [0, t_{1\mathrm{f}}] : p_1(t) \in \mathcal{C}_1(2)\},\ \mathcal{G}_1(3) = \emptyset,\\
&\mathcal{G}_2(3) = \{t \in [0, t_{2\mathrm{f}}] : p_2(t) \in \mathcal{C}_2(3)\},\ \mathcal{G}_2(1) = \{t \in [0, t_{2\mathrm{f}}] : p_2(t) \in \mathcal{C}_2(1)\},\\
&\mathcal{G}_3(1) = \emptyset, \qquad\qquad\qquad\qquad\quad \mathcal{G}_3(2) = \{t \in [0, t_{3\mathrm{f}}] : p_3(t) \in \mathcal{C}_3(2)\}.
\end{aligned}$$

### 2.2.5 Collision avoidance

Collision avoidance is guaranteed if vehicles keep a long enough distance between each other inside their nonempty critical zone. This is ensured by the following constraints

$$t_1 - t_{\mathrm{hw}} \leq t_2 \text{ for all } (t_1, t_2) \in \mathcal{G}_1(2) \times \mathcal{G}_2(1), \tag{2.6}$$

$$t_2 - t_{\mathrm{hw}} \leq t_3 \text{ for all } (t_2, t_3) \in \mathcal{G}_2(3) \times \mathcal{G}_3(2) \tag{2.7}$$

where $t_{\mathrm{hw}}$ is a time headway which introduces an extra safety distance between the vehicles. The first constraint implies that, when both vehicles 1 and 2 reside within their critical set, vehicle 1 is in front of vehicle 2. Similarly, the second constraint say that vehicle 2 should be in front of vehicle 3 within their critical set.

### 2.2.6 Full intersection problem

The performance is evaluated by a sum of cost functions

$$\sum_{i=1}^{3} J_i(\mathbf{x}_i(t), u_i(t), \dot{u}_i(t), \mathbf{x}_i(t_{i\mathrm{f}}), t_{i\mathrm{f}}),$$

where the individual cost functions may differ between vehicles. The functions $J_i(\cdot)$ may include penalties on the final state, penalties for deviation from the reference velocity $v_{i\mathrm{r}}(t)$, penalties for the control actions, penalties for changes in control actions (e.g., discomfort penalties associated to longitudinal jerk), or penalties for the total travel time, etc. The optimization program for the intersection problem can now be formulated as

$$\underset{u_i(t)}{\text{minimize}} \sum_{i=1}^{N} J_i(\mathbf{x}_i(t), u_i(t), \dot{u}_i(t), \mathbf{x}_i(t_{i\mathrm{f}}), t_{i\mathrm{f}}), \tag{2.8a}$$

subject to

$$\dot{\mathbf{x}}_i(t) = A\mathbf{x}_i(t) + Bu_i(t), \quad \forall i \in \{1, 2, 3\}, \tag{2.8b}$$

$$\mathbf{x}_i(t) \in [\mathbf{x}_{i\min}(t), \mathbf{x}_{i\max}(t)], \ \ \forall i \in \{1, 2, 3\}, \tag{2.8c}$$

$$u_i(t) \in [u_{i\min}(t), u_{i\max}(t)], \ \ \forall i \in \{1, 2, 3\}, \tag{2.8d}$$

$$\mathbf{x}_i(0) = \mathbf{x}_{i0}, \mathbf{x}_i(t_{i\mathrm{f}}) = \mathbf{x}_{i\mathrm{f}}, \quad \forall i \in \{1, 2, 3\}, \tag{2.8e}$$

$$t_1 - t_{\mathrm{hw}} \leq t_2, \forall (t_1, t_2) \in \mathcal{G}_1(2) \times \mathcal{G}_2(1), \tag{2.8f}$$

$$t_2 - t_{\mathrm{hw}} \leq t_3, \forall (t_2, t_3) \in \mathcal{G}_2(3) \times \mathcal{G}_3(2), \tag{2.8g}$$

where all constraints are imposed for $t \in [0, t_{if}]$. The initial and final state values are denoted by $\mathbf{x}_{i0}$ and $\mathbf{x}_{if}$, respectively and satisfy $\mathbf{x}_{i0} \in [\mathbf{x}_{i\min}(0), \mathbf{x}_{i\max}(0)]$, $\mathbf{x}_{if} \in [\mathbf{x}_{i\min}(t_{if}), \mathbf{x}_{i\max}(t_{if})]$. The final state values $\mathbf{x}_{if}$ are included both in the objective function and as a hard constraint in program (2.8). This is useful when only part of the states are constrained to a target state (e.g. the final position $p_{if}$), while other states (such as the final velocity) are not and can thus be penalized in the objective.

### 2.2.7 Objective function

A commonly used cost function, penalizing the deviation from a reference velocity, is given by

$$J_{i1}(\cdot) = \int_0^{t_f} q_i(\dot{p}_i(t) - v_{ir})^2 \mathrm{d}t, \tag{2.9}$$

where $v_{ir}$ is the reference velocity for vehicle $i$ and $q_i$ is a nonnegative penalty parameter that may have assigned a certain unit.

A simple model used for obtaining a comfortable drive and limiting actuator usage, is to penalize high accelerations and jerks

$$J_{i2}(\cdot) = \int_0^{t_{if}} r_i u_i^2(t) \mathrm{d}t + \int_0^{t_{if}} s_i \dot{u}_i^2(t) \mathrm{d}t, \tag{2.10}$$

where $r_i$ and $s_i$ are nonnegative penalty parameters that may have a certain unit.

Adding the two objective functions above into $J_i(\cdot) = J_{i1}(\cdot) + J_{i2}(\cdot)$, results in an objective function that seeks to minimize deviation from a given reference position while keeping the ride comfortable for the occupants of vehicle $i$. The complete objective function used in (2.8) is then obtained by adding the objective functions for the three vehicles, as is done in (2.8a).

## 2.3 Optimization programs

The intersection program introduced in Section 2.2, is a mixed integer nonlinear program (MINLP) of the form

$$\underset{\mathbf{x},\mathbf{y}}{\text{minimize}}\ f(\mathbf{x}) \tag{2.11a}$$

subject to

$$h_i(\mathbf{x}) = 0, \quad \text{for all } i \in E \tag{2.11b}$$

$$g_i(\mathbf{x}, \mathbf{y}) \leq 0, \quad \text{for all } i \in I \tag{2.11c}$$

$$\mathbf{y} \in Y \subseteq \{0, 1\}^m, \tag{2.11d}$$

$$\mathbf{x} \in X \subseteq \mathbb{R}^n, \tag{2.11e}$$

where $\mathbf{y}$ is a column vector of $m$ binary variables and $\mathbf{x}$ is a vector of $n$ real variables. The binary variables $\mathbf{y}$ only affects the inequality constraints (2.12b). The MINLP is one of the most general modeling paradigms in optimization and solving MINLP's

typically involves searching through large search trees. Therefor, solving these types of problems have proven to be extremely challenging [15]. However, in the last fifteen-twenty years there have been a big development in software for MINLP solvers, and there exist several commercial and open source solvers. For a survey on MINLP, see [16].

In this thesis we will also be consered with solving a certain subset of MINLP problems; namely continuous nonlinear programs (NLP) of the form

$$\begin{aligned}
&\underset{\Delta\mathbf{x}}{\text{minimize}}\ f(\mathbf{x}) && \text{(2.12a)}\\
&\text{subject to}\\
&g_i(\mathbf{x}) \leq 0, \quad \text{for all } i \in I && \text{(2.12b)}\\
&h_i(\mathbf{x}) = 0, \quad \text{for all } i \in E && \text{(2.12c)}\\
&\mathbf{x} \in X \subseteq \mathbb{R}^n. && \text{(2.12d)}
\end{aligned}$$

where $f,\ g_1, \ldots, g_k,\ h_1, \ldots, h_l$ are functions on $\mathbb{R}^n$ and $\mathbf{x} \in \mathbb{R}^n$. While several commercial and open-source solvers for MINLP's have emerged in the last few decades the methods and solvers for solving NLP's are more mature. For instance, the algorithm we will utilize in this paper is the very efficient sequential quadratic programming method, which has been around since the 60's. Since, computational efficiency is key when solving optimal control programs in real-time applications, reducing optimization programs into a smaller class of optimization programs can be key when evaluating their viability.

## 2.4 Sequential quadratic programming

Consider again the general program (2.12) but with the additional assumptions that $f(\mathbf{x})$, $g_i(\mathbf{x})$ and $h_i(\mathbf{x})$ are smooth nonlinear functions. To obtain a quadratic version of this problem the inequality constraints (2.31) and the equality constraints (2.12b) are linearized. Naturally, solving this new problem will, in general, give an approximate solution to the nonlinear program, (2.12). However, one can amend the objective function with the gradient of the original objective function and the Lagrangian of the constraints. This yields the convex program

$$\begin{aligned}
&\underset{\Delta\mathbf{x}}{\text{minimize}}\ f(\mathbf{x}_k) + \nabla f_k^T(\mathbf{x}_k, \boldsymbol{\lambda}_{k-1})\Delta\mathbf{x}\\
&\qquad\qquad + \frac{1}{2}\Delta\mathbf{x}^T\nabla^2_{\mathbf{xx}}\mathcal{L}(\mathbf{x}_k, \boldsymbol{\lambda}_{k-1})\Delta\mathbf{x} && \text{(2.13a)}\\
&\text{subject to}\\
&g_i(\mathbf{x}_k) + \nabla g_i(\mathbf{x}_k)\Delta\mathbf{x} \leq 0, \quad \text{for all } i \in I && \text{(2.13b)}\\
&h_i(\mathbf{x}_k) + \nabla h_i(\mathbf{x}_k)\Delta\mathbf{x} = 0, \quad \text{for all } i \in E && \text{(2.13c)}
\end{aligned}$$

where $\boldsymbol{\lambda}_k$ are the Lagrange multipliers of the relaxed constraints, $\nabla$ is the gradient, $\nabla^2_{\mathbf{xx}}$ denotes the hessian with respect to $\mathbf{x}$ and $\mathcal{L}$ is the lagrangian. This new program, (2.13), is referred to as a subproblem and is solved iteratively where the current subproblem is linearized around the previous solution. If $(\mathbf{x}_k, \boldsymbol{\lambda}_{k-1})$ is the k:th iterate

of the program (2.13), the new iterate is given by $(\mathbf{x}_{k+1}, \lambda_{k+1}) = (\mathbf{x}_k + \alpha\Delta\mathbf{x}, \boldsymbol{\lambda}_k)$ where $\alpha \in [0, 1]$ is the step length. The iterations continue until a stop criterion is reached, e.g., until the normed difference between consecutive solutions are below some tolerance. A simple version of the SQP algorithm is given in Algorithm 2.4.1. For a more thorough introduction to SQP schemes, see for instance, [12] or [13].

**Algorithm 2.4.1.**

---

**Step 1:** Choose an initial linearization point $(\mathbf{x}_k, \boldsymbol{\lambda}_{k-1})$ and set $k = 1$.

**Step 2:** Solve (2.13) with $(\mathbf{x}_k, \boldsymbol{\lambda}_{k-1})$ to obtain $\Delta\mathbf{x}$ and $\boldsymbol{\lambda}_k$.

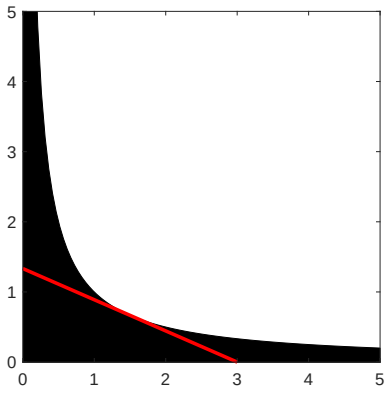
**Step 3:** Update the iterate $(\mathbf{x}_{k+1}, \boldsymbol{\lambda}_k) = (\mathbf{x}_k + \alpha\Delta\mathbf{x}, \boldsymbol{\lambda}_k)$

**Step 4:** Stop if the stopping criterion is fulfilled, otherwise set $k := k + 1$ and go to step 2.

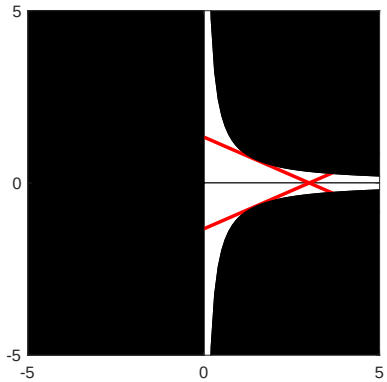
---

### 2.4.1 Real-Time iterations

Even though convex quadratic programs are fast to solve, there is no guarantee for how many subproblems of the type (2.13) that needs to be solved before convergence. When solving a problem offline without time restrictions this might not be an issue, however, when applying this method in a real-time scenario where the consequence of not obtaining a solution fast enough might lead to a fatal accident, it becomes vital. Therefore, it is an advantage if each of the solutions to the subproblems (2.13) are feasible in the original nonlinear program (2.12). It is, however, not true in general, that the subproblems yield solutions that are feasible in the nonlinear program; as the next toy example illustrates.




(a) The white area represents the points that fullfil the constraint $x \geq 1/z$, while the area above the red line represent the feasible points for the linearization.




(b) The white area represents the points that fullfil the constraints $-1/v \leq y \leq 1/v$, while the area between the red lines represent the feasible points for the linearization.

Figure 2.2: Illustration of constraints, with their feasible area depicted in white, and their linearizations by the thick red lines. It can be seen that the linear approximation in (2.2a) is not an inner approximation, but that the linearization of the constraints in (2.2b) is.

**Example 2.4.1.** Consider the minimization program

$$\underset{x,y,z}{\text{minimize}}\ x^2 + y^2 \tag{2.14a}$$

subject to

$$x \geq \frac{1}{z} \tag{2.14b}$$

$$-z \leq y \leq z \tag{2.14c}$$

$$1 \leq z \leq 2 \tag{2.14d}$$

The solution to this problem can be determined using the following argument: Since the only restriction on $y$ is that it should lay in the interval $[-z, z]$, where $z$ is a positive number, $y$ can be set to zero independently of the choices of $x$ and $z$. Thus, it remains to minimize the $x$ part of the objective, which is achieved by choosing $x$ as small as possible. This corresponds to choosing $z$ as large as possible, i.e., the solution is $(x, y, z) = (1/2, 0, 2)$. This gives the objective function value $1/4$.

If the nonlinear constraint (2.14b) is linearized around some feasible point, say $(3/4, 0, 1)$ and the initial Lagrangian multiplier is set to zero, we get from (2.13) that the first subproblem in the SQP scheme can be written as

$$\underset{x,y,z}{\text{minimize}}\ 2(x - \frac{3}{8})^2 + 2y^2 \tag{2.15a}$$

subject to

$$x \geq 2 - z, \tag{2.15b}$$

$$-z \leq y \leq z, \tag{2.15c}$$

$$1 \leq z \leq 2. \tag{2.15d}$$

With a similar reasoning as before, the solution to this subproblem is given by $(x, y, z) = (3/8, 0, 2)$. However, this solution is not feasible in the original program, since the constraint (2.14b) yields

$$\frac{3}{8} \geq \frac{1}{2}.$$

△

As we shall see in the next example, it is possible to, sometimes, remodel the optimization program in order to transform it into a program for which all subproblems have feasible solutions.

**Example 2.4.2.** Recall the minimization program from Example 2.4.1, and apply the variable change $v = 1/z$. The program (2.15) can then be written as

$$\underset{x,y,v}{\text{minimize}}\ x^2 + y^2 \tag{2.16a}$$

subject to

$$x \geq v \tag{2.16b}$$

$$-1/v \leq y \leq 1/v \tag{2.16c}$$

$$\frac{1}{2} \leq v \leq 1 \tag{2.16d}$$

Clearly, this program has the solution $(x, y, v) = (1/2, 0, 1/2)$ with the objective function value $1/4$. Linearizing (2.16c) around the point $(3/4, 0, 1)$ (which is the corresponding point to the one that was linearized around in Example 2.4.1) and set the initial Lagrangian multiplier to zero, we get the subproblem

$$\underset{x,y,v}{\text{minimize}}\ 2(x - \frac{3}{8})^2 + 2y^2 \tag{2.17a}$$

subject to

$$x \geq v \tag{2.17b}$$

$$-(1 - v) \leq y \leq 1 - v \tag{2.17c}$$

$$\frac{1}{2} \leq v \leq 1 \tag{2.17d}$$

which clearly has the solution $(x, y, v) = (1/2, 0, 1/2)$. Note that this solution is feasible in the program (2.16), which after reversing the variable change gives the solution to the original program (2.14), i.e., $(x, y, z) = (1/2, 0, 2)$. △

One can also realize that the solutions to the subproblem of (2.15) might yield solutions that are infeasible in the nonlinear program by studying the graph of its nonlinear constraint and linearization, see Fig. 2.2a, in which it is clear that the linearization makes the search-space larger. On the other hand, the subproblem of (2.16) is an inner approximation as seen in Fig. 2.2b, and thus makes the search-space smaller; which guarantees solutions will be feasible in the original program. Further, we can realize from the figures that the linearization of the inequality constraints (2.12b), will be an inner approximation if $g_i(x)$ is a concave function.

# Chapter 3

# Remodeling for computational efficiency

In this Chapter the main method of the thesis is presented. It consists of remodeling the point mass vehicle model (2.3a) to a vehicle model that is more appropriate when modeling collision avoidance of vehicles with intersecting paths. The reformulation consists of two main steps: 1. to exchange the independent variable, traveling time, for the traveled distance of the vehicle; 2. to remove the nonlinearities in the vehicle model introduced by the change of independent variables, by introducing new state and control signals. In Section 3.1 the transformation is made for the longitudinal dynamics; and it is then shown that applying this vehicle model to the MIP intersection program (2.8) transforms it into a general NLP, which can be solved using SQP. In Section 3.3 the remodeling is extended to lateral dynamics and it is then illustrated how it is possible to use this point mass model to construct an NLP, optimal control program for autonomous highway driving.

## 3.1 Remodeling: Longitudinal dynamics

Integrating the first row of the vehicle model (2.3a) yields

$$x(t) = \int_0^t v_{\mathrm{x}}(\tau)\mathrm{d}\tau.$$

To exchange position for travelling time we want to invert the position function $x = \tilde{x}(t)$ to find $t = \tilde{t}(x)$. For them to be inverses, it must hold that

$$x = \tilde{x}(\tilde{t}(x)) = \int_0^{\tilde{t}(x)} v_{\mathrm{x}}(\tau)\mathrm{d}\tau.$$

Taking the derivative with respect to $x$ gives

$$1 = \tilde{x}'(\tilde{t}(x))\tilde{t}'(x) = v_{\mathrm{x}}(\tilde{t}(x))\tilde{t}'(x)$$

Thus, the inverse exists if

$$\begin{aligned} &v_{\mathrm{x}}(\tilde{t}(x)) = \tilde{x}'(\tilde{t}(x)) \neq 0, \\ &\tilde{t}'(x) = \frac{1}{v_{\mathrm{x}}(\tilde{t}(x))} := \frac{1}{\tilde{v}_{\mathrm{x}}(x)} := z_{\mathrm{x}}(x). \end{aligned} \tag{3.1}$$

Here, the variable $z_{\mathrm{x}}(x)$, with units s/m can be thought of as the inverse velocity of the ego vehicle, evaluated at position $x$. From now on, it will be referred to as the *lethargy* of the ego vehicle.

Now, since the dynamical equation for the travel time is expressed using lethargy. Lethargy will be chosen as a state and its dynamical equation is given by

$$z'_{\mathrm{x}}(x) = \Big(\frac{1}{\tilde{v}_{\mathrm{x}}(x)}\Big)' = -\frac{\tilde{v}'_{\mathrm{x}}(x)}{\tilde{v}^2_{\mathrm{x}}(x)} = -\tilde{v}'_{\mathrm{x}}(x) z^2_{\mathrm{x}}(x) = \tilde{u}_{\mathrm{x}}(x). \tag{3.2}$$

Thus, using equations (3.1) and (3.2), we arrive at a second linear vehicle model

$$\tilde{\mathbf{x}}'(x) = A\tilde{\mathbf{x}}(x) + B\tilde{u}_{\mathrm{x}}(x), \tag{3.3}$$

where

$$\tilde{\mathbf{x}}(x) = \begin{bmatrix} \tilde{t}(x) \\ z_{\mathrm{x}}(x) \end{bmatrix}, \; A = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \; B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Further, it is possible to translate between the states and inputs using the equations

$$v_{\mathrm{x}}(\tilde{t}(x)) = z_{\mathrm{x}}(x), \tag{3.4}$$

$$\tilde{u}_{\mathrm{x}}(x) = -\frac{\mathrm{d}}{\mathrm{d}x}\tilde{v}_{\mathrm{x}}(x) z^2_{\mathrm{x}}(x) = -\frac{dt}{dx}\dot{\tilde{v}}_{\mathrm{x}}(x) z^2_{\mathrm{x}}(x) = -F_{\mathrm{x}}(x) z^3_{\mathrm{x}}(x). \tag{3.5}$$

In the next section it is shown how these remodeling steps can be applied to the constraints and objective function of the intersection program (2.8), to yield a NLP formulation of the control problem for the intersection problem.

## 3.2 Application to intersection problem

Assume three vehicles are approaching a four leg intersection, as is depicted in Fig. 2.1.

To apply the vehicle model above to the intersection scenario in Fig. 2.1 it is necessary to generalize the approach to a situation where there are 3 vehicles, labeled $i = \{1, 2, 3\}$, travelling along paths $p_i(t)$. Since there are three different position functions, it is first necessary to construct a generic position vector $p(t)$ which can describe the position of any vehicle. This can be achieved by applying the transformation

$$x(t) = \frac{p_i(t) - p_{i0}(t)}{p_{i\mathrm{f}} - p_{i0}}.$$

This transformation will project all the paths onto a path starting at 0 and ending at $x_{\mathrm{f}} = 1$. Thus, using this generic position vector, each vehicle is described by a vehicle model of the form (3.3), i.e.,

$$\tilde{\mathbf{x}}'_i(x) = A\tilde{\mathbf{x}}_i(x) + B\tilde{u}_i(x).$$

We will now see that applying this remodeling to the constraints of the intersection problem (2.8) will remove the binary variables while maintaining a linear vehicle model. For the state bounds, initial and final state, we simply change the parameters

$$\tilde{\mathbf{x}}_{i\min} = \begin{bmatrix} 0 & \frac{1}{v_{i\max}(x)} \end{bmatrix}^T, \ \tilde{\mathbf{x}}_{i\max} = \begin{bmatrix} \text{free} & \int_0^x \frac{\mathrm{ds}}{v_{i\min}(s)} \end{bmatrix}^T,$$
$$\tilde{\mathbf{x}}_{i0} = \begin{bmatrix} 0 & \frac{1}{v_{i0}} \end{bmatrix}^T, \qquad \tilde{\mathbf{x}}_{i\mathrm{f}} = \begin{bmatrix} \text{free} & \frac{1}{v_{i\mathrm{f}}} \end{bmatrix}^T.$$

Using (3.5) on the input constraints (2.8d) yield the remodeled input constraints

$$\tilde{u}_i(x) \in z_{i\mathrm{x}}^3(x)[-u_{i\max}(x), -u_{i\min}(x)].$$

Lastly, the collision constraints (2.8f) and (2.8g) can be simplified to linear constraints on the time states

$$t_1(x) + t_{\mathrm{hw}} < t_2(x),$$
$$t_2(x) + t_{\mathrm{hw}} < t_3(x).$$

The full reformulation of the intersection problem (2.8) can then be written as

$$\underset{\tilde{u}_i(x)}{\text{minimize}} \ \sum_{i=1}^{N} \hat{J}_i(\hat{\mathbf{x}}_i(x), \tilde{u}_i(x), \dot{\tilde{u}}_i(x), \mathbf{x}_i(x_\mathrm{f}), x_\mathrm{f}), \tag{3.6a}$$

subject to

$$\tilde{\mathbf{x}}_i'(x) = A\tilde{\mathbf{x}}_i(x) + B\tilde{u}_i(x), \qquad \forall i \in \{1,2,3\}, \tag{3.6b}$$
$$\tilde{\mathbf{x}}_i(x) \in [\tilde{\mathbf{x}}_{i\min}(x), \tilde{\mathbf{x}}_{i\max}(x)], \qquad \forall i \in \{1,2,3\}, \tag{3.6c}$$
$$\tilde{u}_i(x) \in z_{i\mathrm{x}}^3(x)[-u_{i\max}(x), -u_{i\min}(x)], \quad \forall i \in \{1,2,3\}, \tag{3.6d}$$
$$\tilde{\mathbf{x}}_i(0) = \tilde{\mathbf{x}}_{i0}, \tilde{\mathbf{x}}_i(x_{i\mathrm{f}}) = \tilde{\mathbf{x}}_{i\mathrm{f}}, \qquad \forall i \in \{1,2,3\}, \tag{3.6e}$$
$$t_1(x) + t_{\mathrm{hw}} < t_2(x), \qquad \forall x \in [L_{2,1}, H_{1,2}], \tag{3.6f}$$
$$t_2(x) + t_{\mathrm{hw}} < t_3(x), \qquad \forall x \in [L_{3,2}, H_{2,3}], \tag{3.6g}$$

This problem is an NLP with a quadratic objective function and one nonlinear constraint (3.6d). Since this is an NLP it can be solved using the SQP scheme discussed in Section 2.4, by linearizing $z_{i\mathrm{x}}^3(x)$ around, for example, the reference trajectory $z_{i\mathrm{r}}(x)$

$$z_{i\mathrm{x}}^3(x) \approx z_{i\mathrm{r}}^3(x) + 3z_{i\mathrm{r}}^2(x)(z_{i\mathrm{x}}(x) - z_{i\mathrm{r}}(x)) = -2z_{i\mathrm{r}}^3(x) + 3z_{i\mathrm{r}}^2 z_{i\mathrm{x}}(x).$$

Further, $z_{i\mathrm{x}}^3(x)$ is convex in the domain of interest, namely $z_{i\mathrm{x}}(x) > 0$. This means that the constraints (3.6d) are concave and thus that the linearization will be an inner approximation of the search-space. Therefore, whenever the subproblems of the SQP scheme are feasible, the solution to that subproblem will be a feasible point to the nonlinear overtaking program (3.6). As was discussed in Section 2.4.1, this opens up the possibility to utilize RTI or other stopping criterions. So far, we have studied the effect of the remodeling have on the dynamics and the inequality constraints. It remains to transform the objective functions. This is done in the next section.

### 3.2.1 Objective function

Applying the remodeling steps on the reference tracking objective $J_{i1}(\cdot)$, (2.9) yield the objective function

$$\hat{J}_{i1}(\cdot) = q_i \int_0^{x_\mathrm{f}} \left( \frac{1}{\sqrt{z}_{i\mathrm{x}}(x)} - v_{i\mathrm{r}}(x)\sqrt{z_{i\mathrm{x}}(x)} \right) \mathrm{d}x$$

which can be formulated as a convex second order cone function. However, if one wants to keep the quadratic shape of the objective function the term in parenthesis can be linearized around the reference $v_{i\mathrm{r}}$ which results in

$$\begin{aligned}\hat{J}_{i1}(\cdot) &\approx q_i \int_0^{x_\mathrm{f}} v_{i\mathrm{r}}^3(x) \left( z_{i\mathrm{x}}(x) - \frac{1}{v_{i\mathrm{r}}(x)} \right)^2 \mathrm{d}x \\ &\approx q_i \bar{v}_{i\mathrm{r}}^3(x) \int_0^{x_\mathrm{f}} \left( z_i(p) - \frac{1}{v_{i\mathrm{r}}(x)} \right)^2 \mathrm{d}x\end{aligned}$$

where $\bar{v}_{i\mathrm{r}}$ is the mean value of the reference velocity of vehicle $i$.

An exact translation of the second part, $J_{i2}(\cdot)$, (2.10) results in the non-convex objective

$$\begin{aligned}\hat{J}_{i2}(\cdot) = r_i &\int_0^{x_\mathrm{f}} z_{i\mathrm{x}}(x) \left( \frac{\tilde{u}_i(x)}{z_{i\mathrm{x}}^3(x)} \right)^2 \mathrm{d}x \\ &+ s_i \int_0^{x_\mathrm{f}} z_{i\mathrm{x}}(x) \left( \frac{-\tilde{u}_i(x)}{z_{i\mathrm{x}}^4(x)} + 3\frac{\tilde{u}_i^2(x)}{z_{i\mathrm{x}}^5(x)} \right)^2 \mathrm{d}x.\end{aligned}$$

Since this objective function is not convex or concave (which, for instance, can be seen by checking the hessian matrix of the integrand), we suggest to use the quadratic approximation

$$\begin{aligned}\hat{J}_{i2}(\cdot) &\approx r_i \int_0^{x_\mathrm{f}} v_{i\mathrm{r}}^5(x)\tilde{u}_{i\mathrm{x}}^2(x)\mathrm{d}x + s_i \int_0^{x_\mathrm{f}} v_{i\mathrm{r}}(x)^7 u_{i\mathrm{x}}'^2(x)\mathrm{d}p, \\ &\approx r_i \bar{v}_{i\mathrm{r}}^5(x) \int_0^{x_\mathrm{f}} \tilde{u}_{i\mathrm{x}}^2(x)\mathrm{d}x + s_i \bar{v}_{i\mathrm{r}}(x)^7 \int_0^{x_\mathrm{f}} u_{i\mathrm{x}}'^2(x)\mathrm{d}p.\end{aligned}$$

As before, the objective for vehicle $i$ is given by

$$\hat{J}_i = \hat{J}_{i1} + \hat{J}_{i2}.$$

This remodeling technique can be applied to a more general class of intersection programs as is discussed in the next section.

### 3.2.2 Extention of intersection problem

For a more general treatment of the intersection program, see Paper 1 in which the intersection problem for a four-way intersection is formulated for an arbitrary configuration of vehicles with predefined paths but with no vehicles sharing an entry and/or exit leg. It is also straightforward to extend the the approach to an arbitrary

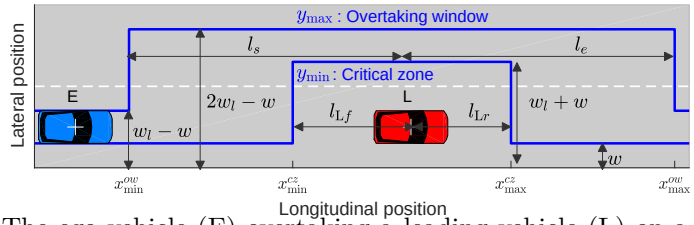



Figure 3.1: The ego vehicle (E) overtaking a leading vehicle (L) on a road with two lanes. The center of gravity of the ego vehicle is allowed to reside between the limits $y_{\min}$ and $y_{\max}$, depicted by the thick, blue, solid lines.

configuration which allows vehicles to travel along the same path (as in the example described in this section) but this work has not been submitted for publication yet. In both these cases it is also possible to handle more scenarios where vehicles turn left, go straight or turn right within the intersection. Applying the presented remodeling scheme to these generalized intersection programs yield an NLP which can be solved using sequential quadratic programming, in the same way as for the example discussed here.

## 3.3 Remodeling: Lateral dynamics

To add lateral control to the dynamical model (3.3), we exchange the state $y(t)$ to $\tilde{y}(x) = y(t(x))$ which has the derivative

$$\tilde{y}'(x) = \frac{\mathrm{d}}{\mathrm{d}x}\tilde{y}(x) = \frac{\mathrm{d}t}{\mathrm{d}x}\frac{\mathrm{d}}{\mathrm{d}t}y(t(x)) = v_{\mathrm{y}}(x)z_{\mathrm{x}}(x),$$
$$\tilde{y}''(x) = \frac{\mathrm{d}}{\mathrm{d}x}\tilde{y}'(x) = \frac{\mathrm{d}}{\mathrm{d}x}(v_{\mathrm{y}}(x)z_{\mathrm{x}}(x)) = v'_{\mathrm{y}}(x)z_{\mathrm{x}}(x) + v_y(x)z'_{\mathrm{x}}(x) = \tilde{u}_{\mathrm{y}}(x).$$

Thus, the dynamical model amended with lateral dynamics read

$$\frac{\mathrm{d}}{\mathrm{d}x}\begin{bmatrix} t(x) \\ z_{\mathrm{x}}(x) \\ y(x) \\ y'(x) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} t(x) \\ z_{\mathrm{x}}(x) \\ y(x) \\ y'(x) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} \tilde{u}_{\mathrm{x}}(x) \\ \tilde{u}_{\mathrm{y}}(x) \end{bmatrix},$$
$$y'_{\mathrm{E}}(\hat{x}) \in [s_{\min}, s_{\max}],$$

where $\tilde{(\cdot)}$ has been set to $(\cdot)$ for ease of notation.

## 3.4 Application to highway driving

In this subsection we will show how the remodeling presented in Section 3.3 can be applied to highway driving scenarios, where the aim is to overtake a slow moving leading vehicle. Collision avoidance with the leading vehicle can be modeled via a rectangular critical zone around the vehicle, as is illustrated in Fig. 3.1. However, just as was the case for the intersection application it is not possible to know when the ego vehicle reaches the critical zone, since the velocity of the ego vehicle is an

optimization variable. Thus, it is, as it was when solving the intersection problem, again advantageous to apply the alternative vehicle model. However, this time the critical zone is also moving (with the speed of the leading vehicle), therefore a change of reference frame will be performed into a relative frame where the critical zone is fixed; before the alternative vehicle model is applied.

Assume that the longitudinal position and velocity of the leading vehicle are given by $x_{\mathrm{L}}(t), v_{\mathrm{L}}(t)$ and that its lateral position is constant. Then, we introduce the relative state vector,

$$\begin{bmatrix} \hat{x}_{\mathrm{E}}(t) \\ \hat{v}_{\mathrm{Ex}}(t) \\ y_{\mathrm{E}}(t) \\ v_{\mathrm{Ey}}(t) \end{bmatrix} = \begin{bmatrix} x_{\mathrm{E}}(t) \\ v_{\mathrm{Ex}}(t) \\ y_{\mathrm{E}}(t) \\ v_{\mathrm{Ey}}(t) \end{bmatrix} - \begin{bmatrix} x_{\mathrm{L}}(t) \\ v_{\mathrm{Lx}}(t) \\ 0 \\ 0 \end{bmatrix},$$
$$y'_{\mathrm{E}}(t) \in [s_{\min}, s_{\max}] v_{\mathrm{Ex}}(t)$$

where $\hat{x}_{\mathrm{E}}(t)$ and $\hat{v}_{\mathrm{Ex}}(t)$ denote the longitudinal position and velocity of the ego vehicle with respect to the leading vehicle. A vehicle model of the form (2.3a) can be formed, but with the longitudinal states exchanged for their relative counter part

$$\begin{bmatrix} \dot{\hat{x}}_{\mathrm{E}}(t) \\ \dot{\hat{v}}_{\mathrm{Ex}}(t) \\ \dot{y}_{\mathrm{E}}(t) \\ \dot{v}_{\mathrm{Ey}}(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \hat{x}_{\mathrm{E}}(t) \\ \hat{v}_{\mathrm{Ex}}(t) \\ y_{\mathrm{E}}(t) \\ v_{\mathrm{Ey}}(t) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ \frac{1}{m} & 0 \\ 0 & 0 \\ 0 & \frac{1}{m} \end{bmatrix} \begin{bmatrix} F_{\mathrm{Ex}}(t) \\ F_{\mathrm{Ey}}(t) \end{bmatrix}.$$
$$y'_{\mathrm{E}}(t) \in [s_{\min}, s_{\max}](v_{\mathrm{Ex}}(t) - v_{\mathrm{L}}(t)).$$

If the steps from Section 3.1 and Section 3.3 are applied to this new vehicle model, but with relative position $\hat{x} = x - x_{\mathrm{L}}$ as the new independent variables instead of the absolute position, the alternative vehicle model is given by

$$\frac{d}{d\hat{x}} \begin{bmatrix} \hat{t}_{\mathrm{E}}(\hat{x}) \\ \hat{z}_{\mathrm{Ex}}(\hat{x}) \\ y_{\mathrm{E}}(\hat{x}) \\ v_{\mathrm{Ey}}(\hat{x}) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \hat{t}_{\mathrm{E}}(\hat{x}) \\ \hat{z}_{\mathrm{Ex}}(\hat{x}) \\ y_{\mathrm{E}}(\hat{x}) \\ y'_{\mathrm{E}}(\hat{x}) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ \frac{1}{m} & 0 \\ 0 & 0 \\ 0 & \frac{1}{m} \end{bmatrix} \begin{bmatrix} \hat{u}_{\mathrm{Ex}}(\hat{x}) \\ \hat{u}_{\mathrm{Ey}}(\hat{x}) \end{bmatrix}, \tag{3.7}$$
$$y'_{\mathrm{E}}(\hat{x}) \in [s_{\min}, s_{\max}](1 - v_{\mathrm{L}}(t)) z_{\mathrm{E}}(\hat{x}). \tag{3.8}$$

where

$$\begin{aligned} \hat{u}_{\mathrm{Ex}}(\hat{x}) &= -\hat{F}_{\mathrm{Ex}}(\hat{x}) \hat{z}^3_{\mathrm{Ex}}(\hat{x}), \\ \hat{u}_{\mathrm{Ey}}(\hat{x}) &= \hat{z}_{\mathrm{Ex}}(\hat{x}) v_{\mathrm{Ey}}(\hat{x}) + v'_{\mathrm{Ey}}(\hat{x}) \hat{z}_{\mathrm{Ex}}(\hat{x}), \\ \hat{z}_{\mathrm{Ex}}(\hat{x}) &= \frac{1}{v_{\mathrm{Ex}}(\hat{x}) - v_{\mathrm{L}}(\hat{x})}, \\ \hat{F}_{\mathrm{Ex}}(\hat{x}) &= -m \hat{v}_{\mathrm{Ex}}(\hat{x}) \hat{v}'_{\mathrm{Ex}}(\hat{x}), \end{aligned}$$

i.e., identical to the transformations made in Sections 3.1 and 3.3, but with the states exchanged for their relative counter parts. To simplify the presentation of the lane

change program, we modify the vehicle model (3.7) slightly by removing the lateral force and using the lateral acceleration as input. This yields

$$\hat{\mathbf{x}}'_{\mathrm{E}}(\hat{x}) = \hat{A}\hat{\mathbf{x}}_{\mathrm{E}}(\hat{x}) + \hat{B}\hat{\mathbf{u}}_{\mathrm{E}}(\hat{x}),$$

where

$$\hat{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix},\ \hat{B} = \begin{bmatrix} 0 & 0 \\ \frac{1}{m} & 0 \\ 0 & 1 \end{bmatrix},\ \hat{\mathbf{x}}_{\mathrm{E}}(\hat{x}) = \begin{bmatrix} \hat{t}_{\mathrm{E}}(\hat{x}) \\ \hat{z}_{\mathrm{Ex}}(\hat{x}) \\ y_{\mathrm{E}}(\hat{x}) \end{bmatrix},\ \hat{\mathbf{u}}_{\mathrm{E}}(\hat{x}) = \begin{bmatrix} \hat{u}_{\mathrm{Ex}}(\hat{x}) \\ y'_{\mathrm{E}}(\hat{x}) \end{bmatrix}.$$

We are now ready to apply this remodeling technique to overtaking program.

### 3.4.1 Full overtaking program

Similarly, to the case of the intersection program, box constraints for the minimum and maximum bounds of the state variables are introduced

$$\hat{\mathbf{x}}_{\min}(\cdot) = [0, \frac{1}{v_{\mathrm{xmax}}(\hat{x}) - v_{\mathrm{L}}}, y_{\min}(\hat{x})]^T,$$
$$\hat{\mathbf{x}}_{\max}(\cdot) = [\mathrm{free}, \frac{1}{v_{\mathrm{xmin}}(\hat{x}) - v_{\mathrm{L}}}, y_{\max}(\hat{x})]^T.$$

The minimum and maximum lateral position $y_{\min}(\hat{x})$ and $y_{\max}(\hat{x})$ are not constants as can be seen in Fig. 3.1. Instead, they should take on one of two values; where the switch between the two values occur when the ego vehicle enters/leaves the critical zone/overtaking window. Since, sampling is done in relative longitudinal position it is known when these changes occur, and thus the minimum and maximum lateral position is given by

$$y_{\min}(\hat{x}) = \begin{cases} w + w_{\mathrm{l}}, & \hat{x} \in x_{\mathrm{L0}} + [-l_{\mathrm{lf}}, l_{\mathrm{lr}}], \\ w, & \text{otherwise}, \end{cases}$$
$$y_{\max}(\hat{x}) = \begin{cases} w_{\mathrm{l}} - w, & \hat{x} \in x_{\mathrm{L0}} + [-l_{\mathrm{s}}, l_{\mathrm{e}}], \\ 2w_{\mathrm{l}} - w, & \text{otherwise}. \end{cases}$$

Similarly, the constraints on the longitudinal input is similar to the intersection case, with the difference that all variables have been switched to their relative counter parts, which gives

$$\hat{u}_{\mathrm{Ex}}(\hat{x}) \in -[F_{\mathrm{xmax}}, F_{\mathrm{xmin}}]\hat{z}^3_{\mathrm{Ex}}(\hat{x}).$$

Lastly, initial states and a final lateral limit is set

$$\hat{\mathbf{x}}^T_{\mathrm{E}}(\hat{x}_0) = \hat{\mathbf{x}}^T_0 = [0,\ \frac{1}{v_{\mathrm{E0}}},\ 0,\ 0],$$
$$y_{\mathrm{E}}(\hat{x}_{\mathrm{f}}) = y_{\mathrm{f}},$$

where the final constraint on the lateral velocity can, for instance, be used to force the overtaking to be completed within the prediction horizon. In summary, we get the optimization program

$$\underset{\hat{\mathbf{u}}_{\mathrm{E}}(\hat{x})}{\text{minimize}}\ \hat{J}(\hat{\mathbf{x}}_{\mathrm{E}}(\hat{x}), \hat{\mathbf{u}}_{\mathrm{E}}(\hat{x}), \hat{\mathbf{u}}'_{\mathrm{E}}(\hat{x}))\mathrm{d}\hat{x} \tag{3.9a}$$

subject to

$$\hat{\mathbf{x}}'_{\mathrm{E}}(\hat{x}) = \hat{A}\hat{\mathbf{x}}_{\mathrm{E}}(\hat{x}) + \hat{B}\hat{\mathbf{u}}_{\mathrm{E}}(\hat{x}), \tag{3.9b}$$

$$\hat{\mathbf{x}}_{\mathrm{E}}(\hat{x}) \in [\hat{\mathbf{x}}_{\min}(\hat{x}), \hat{\mathbf{x}}_{\max}(\hat{x})], \tag{3.9c}$$

$$\hat{u}_{\mathrm{Ex}}(\hat{x}) \in -[F_{\mathrm{xmax}}, F_{\mathrm{xmin}}]\hat{z}^3_{\mathrm{Ex}}(\hat{x}), \tag{3.9d}$$

$$y'_{\mathrm{E}}(\hat{x}) \in [s_{\min}, s_{\max}](1 + v_{\mathrm{L}}\hat{z}_{\mathrm{Ex}}(\hat{x})), \tag{3.9e}$$

$$y_{\mathrm{E}}(\hat{x}_{\mathrm{f}}) = y_{\mathrm{f}}, \tag{3.9f}$$

$$\hat{\mathbf{x}}(0) = \hat{\mathbf{x}}_{\mathrm{E}0}. \tag{3.9g}$$

As in the case of the intersection program, this is an NLP with a quadratic objective function and one nonlinear constraint (3.9d). Just like in the case of the intersection program the nonlinear constraint can be linearized around a reference trajectory $\hat{z}_{\mathrm{r}}(\hat{x})$. Since, this constraint is the same as in the intersection case the linearization will once again yield an inner approximation. Thus, the solution of each subproblem of the SQP algorithm will result in a point which is feasible in the original nonlinear program (3.9).

### 3.4.2 Objective function

For the objective function we suggest to use an approximate quadratic objective function similar to the one used for the intersection program but reduced to only one vehicle and with lateral dynamics added

$$\hat{J}(\cdot) = \left\| \begin{bmatrix} \hat{z}_{\mathrm{E}}(\hat{x}) \\ y_{\mathrm{E}}(\hat{x}) \end{bmatrix} - \begin{bmatrix} 1/\hat{v}_{\mathrm{r}}(\hat{x}) \\ y_{\mathrm{r}}(\hat{x}) \end{bmatrix} \right\|^2_Q + \|\hat{\mathbf{u}}_{\mathrm{E}}(\hat{x})\|^2_R + \|\hat{\mathbf{u}}'_{\mathrm{E}}(\hat{x})\|^2_S, \tag{3.10}$$

where $Q$, $R$ and $S$ are positive semidefinite weighting matrices and $\hat{v}_{\mathrm{r}}$ is the relative reference velocity of the ego vehicle and $y_{\mathrm{r}}$ is the relative lateral position of the ego vehicle. To see how the weighting matrices $Q$, $R$ and $S$ can be found using the corresponding temporal formulation as was done in the intersection case, see Paper 2.

### 3.4.3 Generalized overtaking program

A more extensive treatment of the lane change problem can be found in Paper 2, extended with the scenario of oncoming and adjacent vehicles as is depicted in Fig. 3.2. Collision avoidance with traffic in the adjacent lane is handled using ramp barriers instead of critical zones. Further, the paper also considers nonconstant, but deterministic speed of the ego vehicle. Lastly, a more realistic bound on the longitudinal force, dependent on the velocity of the ego vehicle, is considered in the paper.

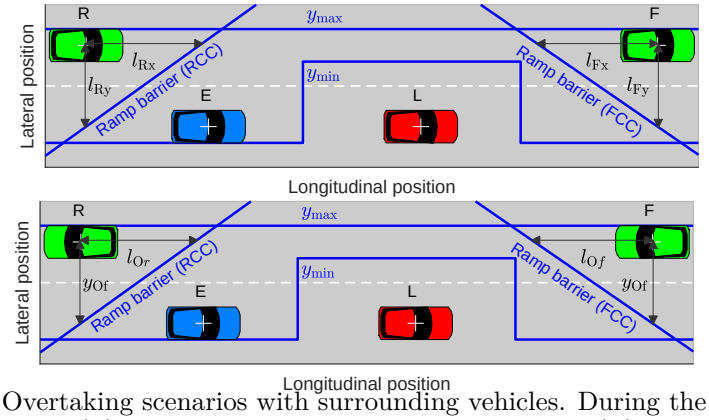



Figure 3.2: Overtaking scenarios with surrounding vehicles. During the overtaking of the leading vehicle (L), the center of gravity of the ego vehicle (E) is allowed to reside within the limits $y_{\min}$, $y_{\max}$. The forward collision constraint (FCC) and rear collision constraint (RCC) make sure the ego vehicle does not collide with the vehicle at the front (F) or rear (R), respectively. The surrounding vehicles in the top plot are referred to as adjacent vehicles while the surrounding vehicles in the bottom are referred to as oncoming vehicles.

# Chapter 4

# Intersection: Crossing sequence

In this section we will briefly address the problem of choosing the crossing order. The most straightforward way to attack this problem would be to solve the program (2.8), for each possible crossing order. However, the number of combinations grow with the factorial of the number of vehicles. Thus, this method will become intractable with growing $n$, e.g., if we have five vehicles we would have to solve $5! = 120$ programs of the form (2.8). However, if one studies the Fig. 2.1, one realizes that it is actually not necessary to study all possible combinations. Clearly, the order of vehicle 2 and 3 is unimportant for the solution of (2.8). In this section we will present a heuristic which provides us with a minimum set of crossing orders, which needs to be examined. To keep it simple, the heuristic will be demonstrated on an example in this section, for a proof of the validity of the heuristic, see Paper 1.

The idea behind the heuristic is simple. Given the matrix $\mathcal{O}$ of all possible crossing orders, we find the orders which will give identical solutions to the program (2.8) and remove all those rows but one from the matrix.

**Example 4.0.1.** The scenario depicted in Fig. 2.1, has the following crossing order matrix

$$\mathcal{O} = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \\ 2 & 1 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \\ 3 & 2 & 1 \end{bmatrix}.$$

depending on the crossing order, a different set of collision avoidance constraints will be applied. Calculating the collision avoidance constraints for each crossing order

and sorting them yield

$$
\begin{aligned}
&[3,\ 1,\ 2]:\ t_1(p) + t_{\text{hw}} < t_2(p)\ t_3(p) + t_{\text{hw}} < t_2(p),\\
&[1,\ 3,\ 2]:\ t_1(p) + t_{\text{hw}} < t_2(p),\ t_3(p) + t_{\text{hw}} < t_2(p),\\
&\\
&[1,\ 2,\ 3]:\ t_1(p) + t_{\text{hw}} < t_2(p),\ t_2(p) + t_{\text{hw}} < t_3(p),\\
&\\
&[2,\ 1,\ 3]:\ t_2(p) + t_{\text{hw}} < t_1(p),\ t_2(p) + t_{\text{hw}} < t_3(p),\\
&[2,\ 3,\ 1]:\ t_2(p) + t_{\text{hw}} < t_1(p),\ t_2(p) + t_{\text{hw}} < t_3(p),\\
&\\
&[3,\ 2,\ 1]:\ t_3(p) + t_{\text{hw}} < t_2(p),\ t_2(p) + t_{\text{hw}} < t_1(p).
\end{aligned}
$$

Thus, crossing orders [3, 1, 2] and [1, 3, 2] yield identical crossing orders. So does [2, 1, 3] and [2, 3, 1]. Therefore, it is enough to consider four crossing orders, one from each of the groups above. Notice that in the identical crossing orders, the only difference in the crossing order is the order of 1 and 3. This is expected, since vehicles 1 and 3 have an empty critical set.

The complete version of the heuristic is given in Algorithm 4.0.1, where $\mathcal{I}$ is the set of all vehicle pairs $(i, j)$ which have an empty critical set. Let us illustrate how it works on the example scenario.

**Example 4.0.2.** In this case we have the sets

$$
\begin{aligned}
&\mathcal{I} = \{(1,3),(3,1)\},\\
&\mathcal{I}_1 = \{(1,3)\},\\
&\mathcal{I}_2 = \emptyset.
\end{aligned}
$$

Take $(1, 3)$. Reduce the crossing order matrix to

$$
\tilde{\mathcal{O}} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \\ 3 & 2 & 1 \end{bmatrix}.
$$

Now, remove $(1, 3)$ from $\mathcal{I}_1$, but this makes $\mathcal{I}_1 = \emptyset$ and thus the algorithm is finished already after one iteration. Notice that the reduced crossing matrix contains exactly one crossing sequence from the groups calculated in Example 4.0.1.

In the Paper 1, it is proved that this heuristic will provide a minimum number of crossing orders to check for any vehicle configuration within a four-way intersection.

**Algorithm 4.0.1.**

---

**Step 0:** Find the set $\mathcal{I}_1 = \{(i, j) \in \mathcal{I} : i < j\}$, sort it on the first and then second entry. Further, create the empty set $\mathcal{I}_2$.

**Step 1:** Take the first element $(i, j)$ in $\mathcal{I}_1$.

**Step 2:** Go through the rows of the crossing order matrix $\mathcal{O}$ and remove all orders (rows) containing the pair $(i, j)$ in sequence. Also, set $\mathcal{I}_2$=$\mathcal{I}_2 \cup \{j\}$.

**Step 3:** Remove $(i, j)$ from $\mathcal{I}_1$, i.e, $\mathcal{I}_1 = \mathcal{I}_1 \setminus (i, j)$.

**Step 4:** If $\mathcal{I}_1 = \emptyset$ stop, otherwise take the next element $(k, l)$ in $\mathcal{I}$ and do the first of the following that apply

1. If $k \in \mathcal{I}_2$ set $(i, j) = (l, k)$ and go to step 2,
2. Set $(i, j) = (k, l)$ go to step 2.

# Chapter 5

# Summary of papers

In this chapter a brief summary is given for each of the appended papers. The Paper 1 deal with the intersection problem and the Paper 2 deals with the problem of highway driving in an overtaking scenario.

## 5.1 Summary of Paper 1

This paper address the problem of deciding the crossing order of vehicles in an intersection scenario. One way is to solve the intersection program presented in Paper 1 for each possible crossing order to find the best one. In this paper, however, it is shown that this is not necessary, since it can be known a-priori that two crossing orders will give identical results. A heuristic is constructed which takes the predefined paths of the vehicles as inputs and returns a minimum set of crossing orders that need to be checked in order to guarantee finding one of the optimal crossing orders.

## 5.2 Summary of Paper 2

This paper deals with solving the overtaking/lane change problem on highways using model predictive control. It is shown that there are several ways to formulate the optimal control problem. The four formulations used are: temporal integer formulation; spatial formulation, spatial formulation using kinetic energy, spatial formulation using inverse speed. The accuraccy of the solutions for the approaches are compared, as well as their computationally efficiency. It turns out that the accuracy is about the same for all four algorithms, but the spatial formulation using inverse speed is significantly faster in an MPC setting since it allows for the use of real-time iterations.

# Chapter 6

# Future work

There are multiple directions in which the work can be expanded, both in practical and more theoretical directions.

One direction is to examine what happens when disturbance is introduced on the inputs and/or states. In reality these disturbances would correspond to sensor inaccuracies, both external and internal. Such disturbances, if small enough, might be handeled simply by updating the MPC fast enough. However, if this is not enough one could try to apply, for instance, stochastic model predictive control.

A second practical direction would be to implement the algorithms in a real vehicle and run real-time tests. This would further showcase the viability of the algorithms by showing that it is, indeed, real-time implementable with regard to the computer power available in a car as well as having the potential to give rise to new interesting research questions.

More theoretical directions include studying the stability and robustness of the introduced model predictive controllers or apply the remodeling technique to more advanced vehicle models (such as the one- or two-wheel model) to see if that has any benefits over the temporal formulations.

# Bibliography


[1] Heilan Yvette Grimes. *The Norse Myths*. Hollow Earth Publishing, 2010 (cit. on p. 3).

[2] Thomas R. Kurfess. *Robotics and Automation Handbook*. 1st ed. CRC Press, 2004 (cit. on p. 3).

[3] Keshav Bimbraw. "Autonomous cars: Past, present and future a review of the development in the last century, the resent scenario and the expected future of autonomous vehicle technology". In: *12th International Conference on Informatics and Control, Automation and Robotics (ICINCO)*. 2015 (cit. on p. 3).

[4] Todd Jochem Chuck Thorpe and Dean Pomerleau. "The 1997 Automated Highway Free Agent Demonstration". In: *In proceedings of Conference on Intelligent Transportation Systems*. 1997 (cit. on p. 3).

[5] Karl Iagnemma Martin Buehler and Sanjiv Singh eds. *The 2005 DARPA Grand Challenge: the great robot race*. Vol. 36. Springer, 2007 (cit. on p. 3).

[6] Karl Iagnemma Martin Buehler and Sanjiv Singh eds. *The Darpa Urban Challenge: Autonomous vehicles in City Traffic*. Vol. 56. Springer, 2009 (cit. on p. 3).

[7] *Taxonomy and definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*. standard. SAE international, June 2018 (cit. on p. 3).

[8] Todd Litman. "Autonomous vehicle implementation predictions". In: *Victoria Transport Policy Institute* 28 (2014) (cit. on p. 4).

[9] Arnaud de La Fortelle Pin Wang Ching-Yao Chan. "A reinforcement based approach for automated lane change maneuvers". In: *IEEE Intelligent Vehicles Symposoium*. 2018 (cit. on p. 4).

[10] Changzhu Zhang, Jinfei Hu, Jianbin Qiu, Wilin Yang, Hong Sun, and Qijun Chen. "A novel Based Observer-Based Steering Control Approach for Path Tracking in Autonomous Vehicles". In: *IEEE Transactions on Fuzzy Systems* 27.2 (2019), pp. 278–290 (cit. on p. 4).

[11] Sebastian Gros Robert Hult Mario Zanon and Paolo Falcone. "Optimal Coordination of Automated Vehicles at Intersections with Turns". In: *European Control Conference*. 2019 (cit. on p. 4).

[12] Jorge Nocedal and Steven J. Wright. *Numerical optimization*. 2nd ed. Springer, 2000 (cit. on pp. 7, 16).

[13] Paul T. Boggs. "Sequential Quadratic Programming". In: *Acta Numerica* 4 (Jan. 1995) (cit. on pp. 7, 16).

[14] Julia Nilsson, Mohammed Ali, Paolo Falcone, and Jonas Sjöberg. "Predictive manoeuvre generation for automated driving". In: *Conference on intelligent transport system (ITSC)*. The Hague, Netherlands, 2013 (cit. on p. 10).

[15] Sebastian Sager. "Numerical methods for mixed-integer optimal control problems". PhD thesis. University of Heidelberg, 2005 (cit. on p. 15).

[16] Pietro Belotti, Christian Kirches, Sven Leyffer, Jeff Linderoth, James Luedtke, and Ashutosh Mahajan. "Mixed-Integer nonlinear optimization". In: *Acta Numerica* 22 (2013), pp. 1–131 (cit. on p. 15).