Latecny-aware edge server for safe autonomous driving

MUSTAFA ADNAN

Master in Systems, Control and Robotics
Date: July 9, 2021
Supervisor: Frank Jiang
Examiner: Jonas Mårtensson
School of Electrical Engineering and Computer Science

Abstract

This thesis examines how a smart edge server, that communicates with autonomous vehicles in an intersection, could contribute to the safety of autonomous driving. Safe path planning is possible when an autonomous vehicle is aware of its surroundings such as obstacles, pedestrians and other vehicles. Occlusions in intersections caused by for instance buildings, trees and parked vehicles affect the sensing of autonomous vehicles which in turn lead to unsafe autonomous driving. Having a smart edge server that senses and processes useful information in the intersection such as pedestrians, obstacles and other vehicles and share it with autonomous vehicles will be beneficial for safe autonomous driving. However an important aspect that affects the safety is the delay caused by processing the information as well as the network latency. The path planning of an autonomous vehicle will not be accurate nor safe if the information of its surroundings is delayed and old.
The objective of this thesis is to implement a latency-aware edge server that communicates with the autonomous vehicles and provide them information of their surroundings. For safe autonomous driving an obstacle avoidance Nonlinear Model Predictive Control, NMPC, algorithm is implemented using the information from the edge server. In order to avoid collisions in an intersection the edge server uses reachability analysis to compute the set of future possible states for each autonomous vehicle as zonotopes. These reachable sets along with static obstacles in the intersection are broadcasted as useful information. Each autonomous vehicle in the intersection listens to the edge server and adds those obstacles and other vehicles' reachable sets as unsafe sets in its obstacle avoidance NMPC formulation to plan a safe trajectory.
The implementations are evaluated for different scenarios and it is shown that a latency-aware edge server reinforces safety for autonomous driving. The result shows that autonomous vehicles, that have intersecting paths, enter into unsafe situations if a smart edge server is not considered. When implementing the smart edge server with the obstacle avoidance NMPC the vehicles still enter into unsafe situations if the edge server does not take the delay into account in the reachable sets computation. Including latency-awareness in the sense that the edge server takes the network latency and computational time into account the autonomous vehicles were able to avoid collision and plan a safe trajectory for different scenarios.

Sammanfattning

Contents

Chapter 1

Introduction

Autonomous vehicles are becoming more common in our society and their development introduces many benefits such as smart, energy efficient and sustainable transport. For smarter, energy efficient and safer transport, the autonomous vehicles use and combine many different type of sensors to perceive the world such as LiDAR, camera, GPS, odometry, inertial measurement units. Moreover these huge amount of data from the sensors need to be analyzed and processed to identify obstacles and also interpreted by advanced control systems to find reliable and optimal navigation path. Connectivity is a big contributing factor for safer and more sustainable transport where the autonomous vehicles collect data and information about their surrounding environment as well as sharing data and information between each other. An application of this could be for instance an edge computing server in an intersection. The edge server gathers information about the the infrastructure of the environment and collect sensor information about the vehicles and pedestrians in the intersection. These information can be analyzed and shared to the driving autonomous vehicles in the intersection which can be helpful for planning a safe trajectory. Furthermore, there are other applications that contributes to the safety of autonomous driving for example [1], where a human operator, in a control tower, take over and control the vehicle remotely in an emergency or complex traffic circumstances. However this introduces another challenge where perceiving the environment of the autonomous vehicle will depend on the network reliability. The 5th-generation of wireless technology, 5G, enhances the speed of the network and reduces the network latency, which makes it more reliable than 4G regarding connectivity. The reduced latency and increase of the bandwidth will make it possible for the autonomous vehicles to imitate human reflexes where big amount of data can transfer in little

amount of time. Although the 5G network provides better performance, the wireless signal will still experience some attenuation due to for instance path loss and shadow fading [2]. The network coverage will vary depending on the environment and will in turn lead to the signal being low in some areas and therefore the connection is less reliable. Therefore including network awareness in an edge server and autonomous vehicles will contribute to sustainable and safer transport.

1.1 Problem formulation

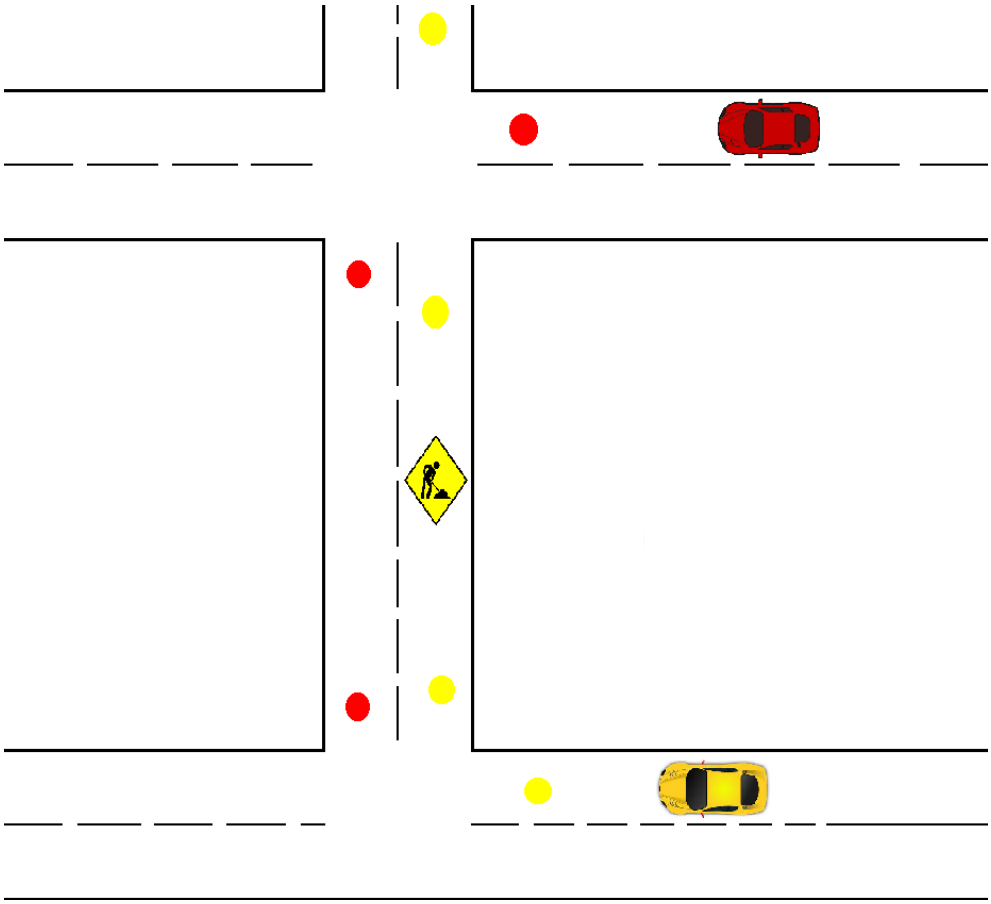


Figure 1.1: Intersection scenario

Consider the following scenario shown in Figure 1.1, where there are two autonomous vehicles, a red vehicle which is following the red trajectory waypoints and a yellow vehicle which is following the yellow trajectory waypoints.

There is a construction work that is out of the vehicles' sight and blocks the road for the yellow vehicle. Given only the sensors of the yellow vehicle, then the yellow vehicle will try to avoid the construction obstacle once it detects it. This may be dangerous since the yellow vehicle needs to change its lane and may encounter other vehicles, for instance the red vehicle in this scenario. Even if the yellow vehicle detects the red vehicle with its sensors, it may not be able to break in time and can cause a collision. If the red vehicle knows the future position of the yellow vehicle then it can plan accordingly and take actions that will prevent the collision. The same applies for the yellow vehicle. Suppose there is an edge server in the intersection that can gather information with its sensors about the environment. The edge serve will then broadcast these information for the vehicles to plan a safe path. The edge serve should also handle the network latency.The objective of the thesis is to:

- Implement a reachability based edge server that is network aware.The edge server provides and reinforces safety for the autonomous vehicles.
- Implement an obstacle avoidance NMPC based algorithm for the autonomous vehicles. The autonomous vehicles should be able to communicate with the edge server.

1.2 Scope

The main focus in this thesis is to study how to handle uncertainties when an autonomous vehicle is communicating through the network with an edge server to plan a safe path with the help of Model Predictive Control and Reachability Analysis. Therefore the following assumptions and limitations are chosen:

- The edge server knows the network latency function $\tau(X, Y)$.
- The edge server knows the position of the obstacles in the intersection.
- The autonomous vehicle knows its state.
- The original trajectory for each autonomous vehicle is known.

1.3 Outline

Chapter 2

Background

2.1 Model Predictive Control

2.1.1 Nonlinear Model Predictive Control

Nonlinear Model Predictive Control, NMPC, is generally formulated as solving online an optimization problem for a system with nonlinear dynamics while satisfying a set of constraints. The feedback strategy, based on receding horizon, is that only the first control input from the optimal control policy is applied. The next time step, the optimization problem is solved again and a new optimal control policy is obtained based on new measurements of the system. This can be summarized as the following steps [3]:

1. Measure the state of the system at the current time t.
2. Solve the optimization problem over a time horizon T and find the optimal control policy $u(\cdot)^*$.
3. Apply the first control from the optimal control policy.
4. Wait for one time step and start again from step 1.

To formulate the optimization problem mathematically, consider a continuous-time system given by the nonlinear dynamics

$$\dot{x}(t) = f(x(t), u(t)) \tag{2.1}$$

, where $x(t) \in \mathbb{R}^n$ are the states of the system, $u(t) \in \mathbb{R}^m$ are the control inputs of the system and $f : \mathbb{R}^n \times \mathbb{R}^m \longrightarrow \mathbb{R}^n$ is a globally Lipschitz continuous function. The nonlinear system satisfies the state and input constraints

$$x(t) \in \mathcal{X}, \quad \forall t \geq 0 \tag{2.2}$$

$$u(t) \in \mathcal{U}, \quad \forall t \geq 0 \tag{2.3}$$

, where $\mathcal{X} \subseteq \mathbb{R}^n$ is connected, and $\mathcal{U} \subseteq \mathbb{R}^m$ is compact. Note that for simplicity (2.3) can be denoted as $u(\cdot) \in \mathcal{U}$ where $u(\cdot)$ is referred as the input trajectory. The objective of the optimization problem is to find an optimal control policy $u(\cdot)^*$ while minimizing a cost function, that describes the desired performance of the system, over a time horizon T

$$J(x(t_0), u(\cdot)) = \int_{t_0}^{t_0+T} L(x(t), u(t))dt \tag{2.4}$$

Given equations (2.1), (2.2), (2.3), (2.4), the optimization problem that is solved online in NMPC is generally formulated as:

$$\min_{u(\cdot)} \quad J(x(t_0), u(\cdot)) = \int_{t_0}^{t_0+T} L(x(t), u(t))dt \tag{2.5a}$$

subject to

$$\dot{x}(t) = f(x(t), u(t)), \quad \forall t \in [t_0, t_0 + T] \tag{2.5b}$$

$$x(t_0) = x_0 \tag{2.5c}$$

$$u(t) \in \mathcal{U}, \quad \forall t \in [t_0, t_0 + T] \tag{2.5d}$$

$$x(t) \in \mathcal{X}, \quad \forall t \in [t_0, t_0 + T] \tag{2.5e}$$

Most systems have nonlinear dynamics and it is desirable to use a nonlinear model of the system for the prediction in NMPC, because it is more accurate than the linear model. Therefore NMPC, may be appealing because it allows using a nonlinear model and also allows considering the state and input constraints explicitly as shown in the optimization problem above. However to guarantee stability, as mentioned in [4], the cost function (2.4) should have an infinite time horizon, $T = \infty$. This causes some issues since it is very difficult if not impossible to solve such optimization problem for nonlinear systems. To find a numerical solution for the optimization problem that is solved online in NMPC the time horizon should be finite. In [4] different approaches are discussed on how to achieve closed loop stability for NMPC while having a finite time horizon. One of the approaches that is mostly used because of its simplicity is the terminal equality constraint $x(t_0 + T) = 0$. The closed loop stability is achieved under the assumption of a quadratic objective functions

and the system must be in the origin in finite time, which may result in long time horizon. Another approach that were discussed is Quasi Infinite Horizon NMPC, where the idea is to split up the infinite horizon cost function into a finite horizon cost function and a terminal penalty as following:

$$\begin{aligned} J(x(t_0), u(\cdot)) &= \int_{t_0}^{\infty} L(x(t), u(t))dt \\ &= \int_{t_0}^{t_0+T} L(x(t), u(t))dt + \int_{t_0+T}^{\infty} L(x(t), u(t))dt \\ &\leq \int_{t_=}^{t_0+T} L(x(t), u(t))dt + L_f(x(t_0+T)) \end{aligned} \tag{2.6}$$

For the inequality $\int_{t_0+T}^{\infty} L(x(t), u(t))dt \leq L_f(x(t_0+T))$ to hold, the trajectories of the closed loop system should remain in the terminal region, Ω, for the time interval $[t_0+T, \infty]$. To achieve this the terminal region, Ω, is created so that a local feedback control law $u = k(x)$ asymptotically stabilizes the system in the terminal region for the time interval $[t_0+T, \infty]$. To ensure that the trajectories of the closed loop system remain in the terminal region a terminal constraint is added to the NMPC optimization problem: $x(t_0+T) \in \Omega$. However finding the terminal region, Ω, and adding the terminal constraint is not an easy task, where Ω can be found for instance by solving an optimization problem offline as explained in [4]. Nevertheless, in [5] it is shown that closed loop stability is achieved without terminal constraints where the terminal region is chosen as $\Omega = \mathcal{X}$. The closed loop system for NMPC is semiglobally asymptotically stable given that the terminal cost function $L_f(x(t_0+T))$ is a local control Lyapunov function of the system, see chapter 7 in [5] for more details.

2.1.2 Obstacle Avoidance

Obstacle avoidance is fundamental for autonomous systems, where they move or navigate in a complex environment with many different obstacles. For example unmanned air vehicles, UAVs, or autonomous vehicles try to move in their environment while not colliding with trees, buildings, people or other autonomous vehicles. Using MPC for obstacle avoidance is appealing for autonomous systems because the obstacle avoidance can be expressed as state constraint in for instance equation (2.2) with the system dynamics in the MPC formulation. There are many implementations for obstacle avoidance using MPC for instance [6] proposes navigation and obstacle avoidance for UAV

using NMPC. In [6] the obstacles are modeled as spheres and the obstacle avoidance is included as state constraint where the distance between UAV's position and the obstacle's position should satisfy some safety distance margin. Another implementation is described in [7], where the authors propose obstacle avoidance based on vehicle's sensor data using MPC. The obstacle avoidance in [7] is formulated as state constraint where a safe area is defined based on LIDAR sensor data, where the vehicle should lie within it to avoid the obstacle. The authors in [8] presents a novel method for optimization based collision avoidance. Their method will be used in this thesis because it applies to general obstacles that can be represented as the union of convex sets. Using strong duality of convex optimization, the non-differentiable obstacle avoidance constraints are reformulated into smooth nonlinear constraints.

Firstly the following are introduced:

- For two vectors $a, b \in \mathbb{R}^l$ and a proper cone $\mathcal{K} \subset \mathbb{R}^l$ then the following holds $a \preceq_{\mathcal{K}} b \iff (b-a) \in \mathcal{K}$. If $\mathcal{K} = \mathbb{R}^l{}_+$ then $\preceq_{\mathcal{K}}$ is the standard element-wise inequality $\leq$. For more details see section 2.4 in [9].
- Let $||\cdot||_*$ be the dual norm of $||\cdot||$, and $\mathcal{K}^*$ is the dual cone of $\mathcal{K}$.
- The Minkowski difference of two sets $\mathcal{A}$ and $\mathcal{B}$ is defined as $\mathcal{A} \ominus \mathcal{B} = \{a - b | a \in \mathcal{A}, b \in \mathcal{B}\}$.

Consider the space occupied by an obstacle is described by the a convex set with no relative interior

$$\mathcal{Z} = \left\{ y \in \mathbb{R}^i : Cy \preceq_{\mathcal{K}} d \right\} \tag{2.7}$$

, where $C \in \mathbb{R}^{j \times i}$, $d \in \mathbb{R}^j$ and $\mathcal{K} \subset \mathbb{R}^i$ is a closed convex pointed cone with non-empty interior.
The space which the controlled object occupies at time t with a state $x(t) \in \mathbb{R}^n$ is denoted by

$$\mathcal{S}(x(t)) = R(x(t))\mathcal{X}_0 + T(x(t)) \tag{2.8}$$

, where $R(x(t)) : \mathbb{R}^n \longrightarrow \mathbb{R}^{i \times i}$ is the rotation matrix for the controlled object and $T(x(t) : \mathbb{R}^n \longrightarrow \mathbb{R}^i$ is the translation vector. The initial set of the controlled object, $\mathcal{X}_0$, is given by

$$\mathcal{X}_0 = \left\{ y \in \mathbb{R}^i : Ay \preceq_{\bar{\mathcal{K}}} b \right\} \tag{2.9}$$

where $A \in \mathbb{R}^{h \times i}$, $b \in \mathbb{R}^h$ and $\bar{\mathcal{K}} \subset \mathbb{R}^i$ is a closed convex pointed cone with non-empty interior. The obstacle avoidance constraint can be formulated as

$\mathcal{S}(x(t)) \cap \mathcal{Z} = \emptyset$. However it is non-convex and non-differetntiable, which cannot be included directly in the MPC formulation. This obstacle avoidance constraint be reformulated using the notion of signed distance which is given by

$$sd(\mathcal{S}(x), \mathcal{Z}) = dist(\mathcal{S}(x), \mathcal{Z}) - pen(\mathcal{S}(x), \mathcal{Z}) \tag{2.10}$$

, where the distance function $dist(\cdot, \cdot)$ and the penetration function $pen(\cdot, \cdot)$ are defined as

$$dist(\mathcal{S}(x), \mathcal{Z}) = \min_{\Delta} \left\{ ||\Delta|| : (\mathcal{S}(x) + \Delta) \cap \mathcal{Z} \neq \emptyset \right\} \tag{2.11}$$

$$pen(\mathcal{S}(x), \mathcal{Z}) = \min_{\Delta} \left\{ ||\Delta|| : (\mathcal{S}(x) + \Delta) \cap \mathcal{Z} = \emptyset \right\} \tag{2.12}$$

If the signed distance (2.10) is positive then the two sets $\mathcal{S}(x)$ and $\mathcal{Z}$ do not intersect. On the other hand if the two sets $\mathcal{S}(x)$ and $\mathcal{Z}$ overlap with each other then the signed distance (2.10) is negative.

The definition in (2.11) can be rewritten as

$$dist(\mathcal{S}(x), \mathcal{Z}) = \min_{s,z} \{ ||s - z|| : s \in \mathcal{S}, z \in \mathcal{Z} \}$$

Using equations (2.7), (2.8) and (2.9) the distance between the controlled set $\mathcal{S}$ and the obstacle set $\mathcal{Z}$ can be found by solving the optimization problem

$$\begin{aligned} \min_{s,z} \quad & \left|\left| R(x)s + T(x) - z \right|\right| \\ \text{subject to} \quad & Cz \preceq_{\mathcal{K}} d \\ & As \preceq_{\bar{\mathcal{K}}} b \end{aligned}$$

The dual of this minimization problem is given by the following (see section 8.2 in [9] for more details)

$$\begin{aligned} \max_{\lambda,\mu} \quad & -b^T\mu + (CT(x) - d)^T\lambda \\ \text{subject to} \quad & A^T\mu + R(x)^T C^T \lambda = 0 \\ & \left|\left| C^T\lambda \right|\right|_* \leq 1 \\ & \lambda \succeq_{\mathcal{K}^*} 0, \quad \mu \succeq_{\bar{\mathcal{K}}^*} 0 \end{aligned}$$

Assuming the two sets $\mathcal{S}(x)$ and $\mathcal{Z}$ have non-empty relative interior, strong duality holds. The condition where the distance between the two sets is greater than a minimum distance margin, Δ, can in turn be reformulated as

$$\begin{aligned} &dist(\mathcal{S}(x), \mathcal{Z}) > \Delta \iff \\ &\exists \lambda \succeq_{\mathcal{K}^*} 0, \exists \mu \succeq_{\bar{\mathcal{K}}^*} 0 : \begin{cases} -b^T\mu + (CT(x) - d)^T\lambda > \Delta \\ A^T\mu + R(x)^T C^T \lambda = 0 \\ \left|\left|C^T\lambda\right|\right|_* \leq 1 \end{cases} \end{aligned} \tag{2.13}$$

If the dual norm $||\cdot||_*$ is the Euclidean distance and the dual cones $\mathcal{K}^*$ and $\bar{\mathcal{K}}^*$ are standard cones or second-order cones, then smoothness is ensured if (2.13) is included in the MPC formulation. Since the distance function does not handle overlapping sets, the penetration function needs to be included as shown in (2.10). This is done because if collision free trajectories are not found by the MPC then the optimization problem is infeasible. Including the penetration function, infeasibility can be solved by adding slack variables in the MPC formulation where trajectories with minimum penetration can be found if a collision cannot be avoided.

The penetration between the controlled object set and the obstacle set, $pen(\mathcal{S}(x), \mathcal{Z})$, can be formulated as following

$$\begin{aligned} pen(\mathcal{S}(x), \mathcal{Z}) = pen(\mathbf{0}, \mathcal{Z} \ominus \mathcal{S}(x)) &= \inf_{\nu:||\nu||_*=1} \{ \max_{y \in \mathcal{Z} - \mathcal{S}(x)} \{y^T\nu\}\} \\ &= \inf_{\nu:||\nu||_*=1} \{ \max_{s \in \mathcal{S}(x), z \in \mathcal{Z}} \{\nu^T(z-s)\}\} \end{aligned}$$

, see [8] for the derivation and proof. Using equations (2.7), (2.8) and (2.9) the maximization problem in the penetration can be formulated as

$$\begin{aligned} \max_{s,z} \quad & \nu^T(z - R(x)s - T(x)) \\ \text{subject to} \quad & Cz \preceq_{\mathcal{K}} d \\ & As \preceq_{\bar{\mathcal{K}}} b \end{aligned}$$

The dual of this maximization problem is given by the following

$$\begin{aligned} \min_{\lambda,\mu} \quad & d^T\lambda + b^T\mu - \nu^T T(x) \\ \text{subject to} \quad & C^T\lambda = \nu \\ & A^T\mu = -R(x)^T\nu \\ & \lambda \succeq_{\mathcal{K}^*} 0, \quad \mu \succeq_{\bar{\mathcal{K}}^*} 0 \end{aligned}$$

Using this result, the penetration function is reformulates as the following

$$pen(\mathcal{S}(x), \mathcal{Z}) =$$

$$= \inf_{\nu:||\nu||_*=1} \left\{ \min_{\lambda,\mu} \left\{ d^T\lambda + b^T\mu - \nu^T T(x) : \begin{cases} C^T\lambda = \nu \\ A^T\mu = -R(x)^T\nu \\ \lambda \succeq_{\mathcal{K}^*} 0 \\ \mu \succeq_{\bar{\mathcal{K}}^*} 0 \end{cases} \right\} \right\}$$

$$= \inf_{\nu,\lambda,\mu} \left\{ d^T\lambda + b^T\mu - \nu^T T(x) : \begin{cases} C^T\lambda = \nu \\ A^T\mu = -R(x)^T\nu \\ \lambda \succeq_{\mathcal{K}^*} 0 \\ \mu \succeq_{\bar{\mathcal{K}}^*} 0 \\ ||\nu||_* = 1 \end{cases} \right\}$$

$$= \inf_{\lambda,\mu} \left\{ b^T\mu + (d - CT(x))^T\lambda : \begin{cases} A^T\mu + R(x)^T C^T\lambda = 0 \\ ||C^T\lambda||_* = 1 \\ \lambda \succeq_{\mathcal{K}^*} 0 \\ \mu \succeq_{\bar{\mathcal{K}}^*} 0 \end{cases} \right\}$$

Since strong duality hold, the condition that the penetration should be less than a maximum penetration, p_{max}, can be written as

$$pen(\mathcal{S}(x), \mathcal{Z}) < p_{max} \iff$$
$$\exists \lambda \succeq_{\mathcal{K}^*} 0, \exists \mu \succeq_{\bar{\mathcal{K}}^*} 0 : \begin{cases} b^T\mu + (d - CT(x))^T\lambda < p_{max} \\ A^T\mu + R(x)^T C^T \lambda = 0 \\ ||C^T\lambda||_* = 1 \\ \lambda \succeq_{\mathcal{K}^*} 0 \\ \mu \succeq_{\bar{\mathcal{K}}^*} 0 \end{cases} \tag{2.14}$$

Now the signed distance condition for obstacle avoidance, $sd(\mathcal{S}(x), \mathcal{Z}) > \Delta$, will be derived using (2.10), (2.13) and (2.14). Assuming that $\mathcal{S}(x) \cap \mathcal{Z} \neq \emptyset$, then from equation (2.10) the signed distance is $sd(\mathcal{S}(x), \mathcal{Z}) = -pen(\mathcal{S}(x), \mathcal{Z})$. In that case the condition is formulated as

$$sd(\mathcal{S}(x), \mathcal{Z}) > \Delta \iff -pen(\mathcal{S}(x), \mathcal{Z}) > \Delta \iff pen(\mathcal{S}(x), \mathcal{Z}) < -\Delta$$

Using equation (2.14) and $p_{max} = -\Delta$ the signed distance condition can be rewritten as

$$sd(\mathcal{S}(x), \mathcal{Z}) > \Delta \iff$$
$$\exists \lambda \succeq_{\mathcal{K}^*} 0, \exists \mu \succeq_{\bar{\mathcal{K}}^*} 0 : \begin{cases} -b^T\mu + (CT(x) - d)^T\lambda > -\Delta \\ A^T\mu + R(x)^T C^T \lambda = 0 \\ ||C^T\lambda||_* = 1 \\ \lambda \succeq_{\mathcal{K}^*} 0 \\ \mu \succeq_{\bar{\mathcal{K}}^*} 0 \end{cases} \tag{2.15}$$

Now assuming that $\mathcal{S}(x) \cap \mathcal{Z} = \emptyset$ then from equation (2.10) the signed distance is $sd(\mathcal{S}(x), \mathcal{Z}) = dist(\mathcal{S}(x), \mathcal{Z})$. The signed distance condition is $sd(\mathcal{S}(x), \mathcal{Z}) > \Delta \iff dist(\mathcal{S}(x), \mathcal{Z}) > \Delta$, which is equivalent to the condition (2.13). However if the conditions in (2.15) are satisfied then the conditions in (2.13) also hold, because of the homogeneity in λ and μ. To include obstacle avoidance constraint as signed distance condition in the MPC formulation, the conditions in (2.15) should be added.

2.2 Reachability Analysis

Reachability analysis is often used in applications to provide safety for a dynamical system, where in the presence of disturbances or complex environment the dynamical system may be driven to unsafe states. An example of

such applications is [10] where the authors propose an approach for ensuring safety when a vehicle of different levels of automation is parking. The authors use LTL to formulate a safe parking task and then use Hamilton-Jacobi reachability analysis computing backward reachable sets to ensure safety for a vehicle with nonlinear dynamics. Another application is [11] where the authors present an approach to verify safety for high order models of automated vehicles using low order models for a planned maneuver. The authors use forward reachable sets to compute the occupancy of other traffic participants and check whether the vehicle intersects with them, i.e. a possible collision or not. The approach also considers many types of uncertainties for instance sensor noise, uncertain friction coefficient, and uncertain initial states.

2.2.1 Zonotopes

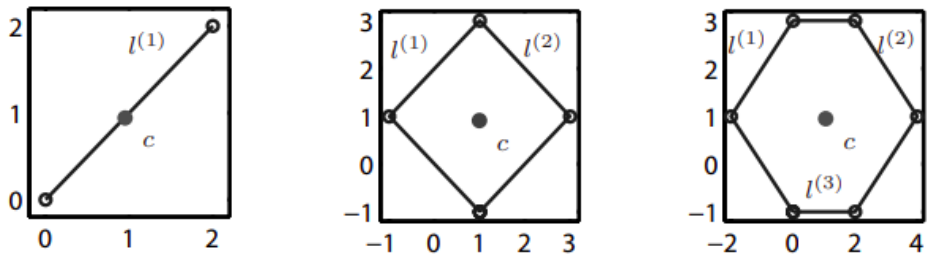
One appealing way to represent sets is using zonotopes, since they can handle high dimensional systems very efficiently. Moreover they have important properties for instance they are closed under Minkowski addition and linear maps [12]. A zonotope, $\mathcal{Z} \subset \mathbb{R}^n$ with a center $c \in \mathbb{R}^n$ and generators $g^{(i)} \in \mathbb{R}^n$, is defined as

$$\mathcal{Z} = \left\{ z \in \mathbb{R}^n \middle| \quad z = c + \sum_{i=1}^{e} \beta_i g^{(i)}, \quad \beta_i \in [-1, 1] \right\} \tag{2.16}$$

The order of the zonotope is defined as $\rho = \frac{e}{n}$. Zonotopes have three different interpretations as mentioned in [12]. Before introducing the interpretations recall the following set operations:

- The Minkowski sum of two sets $\mathcal{A} \subset \mathbb{R}^n$ and $\mathcal{B} \subset \mathbb{R}^n$ is defined as $\mathcal{A} \oplus \mathcal{B} = \{a + b | a \in \mathcal{A}, b \in \mathcal{B}\}$
- The linear transformation of the set $\mathcal{A} \subset \mathbb{R}^n$ with the transformation matrix $T \in \mathbb{R}^{p \times n}$ is define as $T \otimes \mathcal{A} = \{Ta | a \in \mathcal{A}\}$

One interpretation is that the zonotope (2.16) is a Minkowski sum of the line segments $l^{(i)} = [-1, 1] \cdot g^{(i)}$ which can be illustrated in Figure 2.1. The zonotope (2.16) can also be reformulated as $\mathcal{Z} = c \oplus G \otimes B$, given that $B = [-1, 1]^e$ and $G = [g^{(1)}, \cdots, g^{(e)}]$. The following reformulation allows the zonotope to be interpreted as the projection of the e-dimensional unit hypercube B with the transformation matrix G, translated to the center c.



(a) $c \oplus l^{(1)}$ (b) $c \oplus l^{(1)} \oplus l^{(2)}$ (c) $c \oplus l^{(1)} \oplus l^{(2)} \oplus l^{(3)}$

Figure 2.1: Constructing a zonotope using Mnkowski sum

The zonotope (2.16) which is in G-Representation is usually denoted for simplicity as $\mathcal{Z} = (c, G)$. However zonotopes can also be represented in halfspace representation, H-Representationcan as $\mathcal{Z} = \{y \in \mathbb{R}^n | Cy \leq d, \quad C \in \mathbb{R}^{\nu \times n}, d \in \mathbb{R}^p\}$. To convert a zonotope from G-Representation to H-representation the n-dimensional cross product operator is needed $nX()$.

Consider the matrix $H = [h^{(1)}, \cdots, h^{(n-1)}] \in \mathbb{R}^{n \times n-1}$, where $h^{(i)} \in \mathbb{R}^n$. Let $H^{[i]} \in \mathbb{R}^{n \times n}$ be the H matrix where the i^{th} row is removed. The n-dimensional cross product of H is defined as

$$nX(H) = [\cdots, (-1)^{i+1} \det H^{[i]}, \cdots]^T \tag{2.17}$$

Given the generator matrix $G = [g^{(1)}, \cdots, g^{(e)}] \in \mathbb{R}^{n \times e}$, the generators that span a facet can be found by removing $e - n + 1$ generators from G. They are denoted as $G^{<\gamma,\ldots,\eta>}$, where $\gamma, ..., \eta$ are $e - n + 1$ indices of the generators that are removed from the generator matrix G. The H-Representation is given by $\mathcal{Z} = \{y \in \mathbb{R}^n | Cy \leq d\}$, where

$$C = \begin{bmatrix} C^+ \\ -C^+ \end{bmatrix}, \quad C^+ = \begin{bmatrix} C_1^+ \\ \vdots \\ C_\nu^+ \end{bmatrix}, \quad C_i^+ = \frac{nX(G^{<\gamma,\ldots,\eta>})^T}{||nx(G^{<\gamma,\ldots,\eta>})||_2}$$

$$d = \begin{bmatrix} d^+ \\ d^- \end{bmatrix} = \begin{bmatrix} C^+c + \Delta d \\ -C^+c + \Delta d \end{bmatrix}, \quad \Delta d = \sum_{j=1}^{e} |C^+ g^{(j)}|$$

There are $2\binom{e}{n-1}$ facets where the normal vector of each facet is given by C_i^+. Note that $i = 1, ..., \binom{e}{n-1}$, since the other normal vectors are given by $-C_i^+$ due to the central symmetry of zonotopes. For more details see Theorem 2 in [12].

2.2.2 Reachable sets

Consider the dynamical system (2.1) with initial state $x(t_0)$ in a set of uncertain initial states $x(t_0) \in \mathcal{X}_0 \subset \mathbb{R}^n$ and is controlled by an input trajectory $u(\cdot) \in \mathcal{U} \subset \mathbb{R}^n$. The set of states the system can reach at time $t_f > t_0$ is given by the reachable set

$$\mathcal{R}(t_f) = \left\{ x(tf) = \int_{t_0}^{t_f} f(x(t), u(t))dt \;\middle|\; x(t_0) \in \mathcal{X}_0, u(\cdot) \in \mathcal{U} \right\} \quad (2.18)$$

The reachable set in a time interval $t \in [t_0, t_f]$ is described by the union of reachable sets within that interval and is denoted by

$$\mathcal{R}([t_0, t_f]) = \bigcup_{t \in [t_0, t_f]} \mathcal{R}(t) \quad (2.19)$$

The reachable set are computed using CORA toolbox [13] based on the theory in [14]. In order to take safety into consideration the reachable set are computed in an over-approximative way, since exact reachable set can only be computed for special cases. Computing a reachable set for nonlinear systems are of more complicated than linear systems since because many important properties no longer applies for instance superposition principle that is obtaining the homogeneous and the in-homogeneous solution separately. Another important property is the use of linear transformations when computing reachable sets for linear time invariant systems since zonotopes are invariant under linear transformations. Therefore reachable set for nonlinear systems are computed in [13] by state space abstraction as linear systems for instance. The linearization errors are handled and added as uncertain input so that the reachable sets are computed in an over-approximated way. For more details on linearization, handling the linearization errors, and the computation of the reachable set the reader is referred to [14].

To compute the reachable set (2.19), the time interval $[t_0, t_f]$ is divided in N steps by a time step h such that $N = \frac{t_f - t_0}{h}$. The reachable set is then obtained by the union of the reachable sets in each smaller interval such that

$$\mathcal{R}([t_0, t_f]) = \bigcup_{k=1}^{N} \mathcal{R}(t_k), \quad t_k = [t_0 + (k-1)h, t_0 + kh]$$

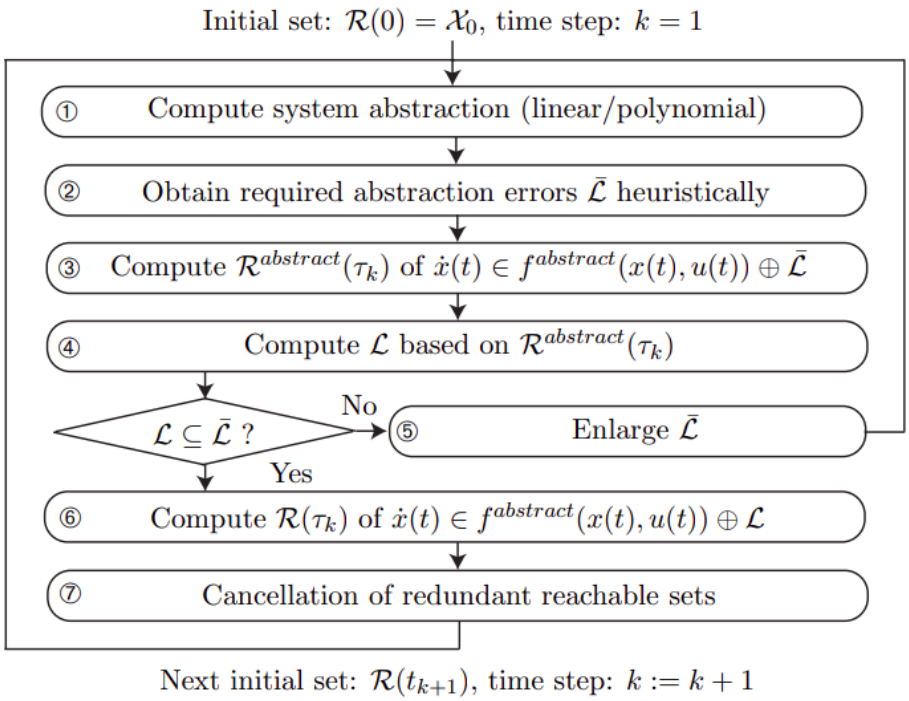


Figure 2.2: An overview of the computation of reachable set for nonlinear systems.

A brief overview of the computation of the reachable set is illustrated in Figure 2.2 and is explained in the following steps:

1. The nonlinear system (2.1) is linearized to $\dot{x} = f^{lin}(x, u)$ by using Taylor series of order κ. The reachable set are computed in an over-approximative way since the set of linearization errors $\mathcal{L}$ makes sure that $f(x, u) \in f^{lin}(x, u) \oplus \mathcal{L}$.

2. Without the set of linearization errors $\mathcal{L}$, the reachable set $\mathcal{R}^{lin}(t_k)$ is computed. The reachable set $\mathcal{R}^{err}$ which is caused by $\mathcal{L}$ is later added to $\mathcal{R}^{lin}$ because of the superposition principle.

3. Due to linearization errors obtained from the system dynamics $f^{lin}(x, u)$, the reachable set is expanded. The set $\bar{\mathcal{R}}^{err}$ is defined to restrict the expansion in which $\mathcal{R}^{err}$ has to be enclosed. The set of admissible linearization errors $\bar{\mathcal{L}}$ is obtained due to $\bar{\mathcal{R}}^{err}$.

4. The set of linearization errors $\mathcal{L}$ is computed from the admissible reachable set $\mathcal{R}(t_k) = \mathcal{R}^{lin}(t_k) \oplus \bar{\mathcal{R}}^{err}$.

5. If $\mathcal{L} \not\subset \bar{\mathcal{L}}$, then the set of linearization errors is not admissible. Consequently the reachable set $\mathcal{R}(t_k)$ is split into two sets. The reachable set computation is performed on both sets which increases the number of reachable set representation in the current time interval by one.

6. If $\mathcal{L} \subset \bar{\mathcal{L}}$ then the set of linearization errors is accepted. As mentioned before, the reachable set is now computed by the superposition of the reachable set without linearization errors and the reachable set of the accepted set of linearization errors: $\mathcal{R}(t_k) = \mathcal{R}^{lin}(t_k) \oplus \mathcal{R}^{err}$.

7. Redundant reachable sets that are covered by previous reachable sets are canceled. The time step is increased $k = k + 1$ and the initial set is for the next time step is updated to $\mathcal{R}(t_{k+1})$.

Chapter 3

Methods

In this chapter the methods and approaches to solve the problem, introduced in Section 1.1, will be presented based on the theory in Chapter 2.

3.1 Autonomous vehicle

3.1.1 Vehicle modeling

To implement obstacle avoidance NMPC in an autonomous vehicle, the vehicle is modeled where the function (2.1) needs to be found. There are many different models describing the dynamics of an autonomous car depending on the car that is modeled. One example is the four wheels model which is used in for instance [15]. The model is used in a linear time varying MPC to control the four wheels independently while the car is following a path at high speed in a slippery road covered with snow. Another often used vehicle model is the bicycle model where the four wheels are simplified into two wheels, one front wheel and one rear wheel at the center of the vehicle. There are different types of bicycle models that are more appropriate to use depending on the vehicle's speed. In [16] a comparison is done between the kinematic bicycle model and the dynamic bicycle model. The dynamic bicycle model uses tire models since it models the forces acting on the vehicle. On the other hand the kinematic bicycle model only uses the motion of the vehicle to describe the vehicle dynamics. Tire models becomes singular when the vehicle has low speeds and they are computationally demanding. Therefore as, stated in [16], the kinematic bicycle model is more appropriate to use in stop-and-go scenarios and is less computation demanding. The kinematic bicycle model preformed similarly well as the dynamic bicycle model when the kinematic

bicycle model was discretized at 200 ms and the dynamic bicycle model was discretized at 100 ms.

Considering the computational power and the scenario presented in section 1.1, it was concluded that the kinematic bicycle model is the best fit to model the vehicle dynamics in the NMPC. Assuming that there is no slip, i.e. the slip angle is zero, the vehicle velocity is directed as the heading of the vehicle. There are many different works that use the kinematic bicycle model with no slip angle such as [10] that uses reachability analysis for a vehicle of different levels of automation to ensure safety while parking. Another work is [17] where the authors develop a motion planning for overtaking method using MPC. For the kinematic bicycle model, from [17], which is illustrated in Figure 3.1, the nonlinear continuous-time dynamics of the vehicle is given by

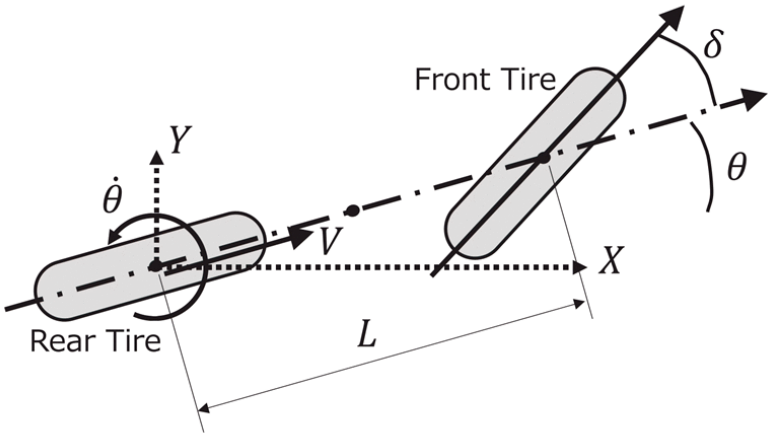


Figure 3.1: Kinematic bicycle model

$$\dot{X} = V\cos\theta \tag{3.1a}$$

$$\dot{Y} = V\sin\theta \tag{3.1b}$$

$$\dot{\theta} = \frac{V}{L}\tan\delta \tag{3.1c}$$

$$\dot{V} = a \tag{3.1d}$$

, where (X, Y) is the position of the rear wheel, V is the velocity of the vehicle, θ is the yaw of the vehicle, δ is the steering angle and a is the acceleration of the vehicle. Given the dynamics in (3.1) the function (2.1) is given by

$$f(x(t), u(t)) = \begin{bmatrix} V\cos\theta \\ V\sin\theta \\ \frac{V}{L}\tan\delta \\ a \end{bmatrix} \tag{3.2}$$

, where the states are $x(t) = [X(t), Y(t), \theta(t), V(t)]^T \in \mathbb{R}^4$ and the control inputs are $u(t) = [\delta(t), a(t)]^T \in \mathbb{R}^2$.

3.1.2 Obstacle avoidance NMPC

Since the optimization problem can only be solved numerically, the time horizon is dived in N steps by a time step k. The optimization problem is then discretized and solved for N steps. Let $F(x_k, u_k)$ denote the discretized function of the continuous-time nonlinear dynamics of the vehicle (3.2), where x_k and u_k are the states and the control inputs in the k^{th} time step respectively. As mentioned in section 2.1.1 the cost function is chosen as the desired of the performance. Since the desired performance of the vehicle is to follow a set of waypoints, the the most common form of the cost function L is the quadratic form

$$L(x_k, u_k) = (x_k - x_{ref})^T Q_1 (x_k - x_{ref}) + (u_{k+1} - u_k)^T Q_2 (u_{k+1} - u_k) \tag{3.3}$$

, where x_{ref} denotes the desired reference states. The second term in (3.3) is to ensure smoothness in the control. The weight matrices Q_1 and Q_2 are symmetric positive definite. The terminal cost will also have a quadratic form

$$L_f(x_N) = (x_N - x_{ref})^T Q_f (x_N - x_{ref}) \tag{3.4}$$

, where the matrix Q_f is positive definite and chosen such that semiglobal asymptotic stability for the closed loop system is ensured as explained in [5].

Recall that including obstacle avoidance in the NMPC optimization problem could be done by adding (2.15). In order for the NMPC to not be infeasible, i.e. when collision cannot be avoided, the penetration depth, Δ, in (2.15) is chosen as slack variable s. Including the slack variable in the objective function of the minimization problem of the NMPC will make it possible to achieve collision avoidance while the optimization problem is easier to solve. Consider M

static obstacles, $\{\mathcal{Z}_1, \cdots, \mathcal{Z}_M\}$, where the obstacles are zonotopes that can be represented in halfspace representation, as shown in subsection 2.2.1. Consequently the cone in the obstacle convex set (2.7) is the standard cone, $\mathcal{K} = \mathbb{R}^l_+$, which in turn makes the ordering $\preceq_\mathcal{K}$ as the standard element-wise inequality $\leq$. The Euclidean distance, 2-norm, is used and the dual norm of it in (2.15) is also the for the 2-norm, see [9] for details. The optimization problem for the obstacle avoidance NMPC of M static obstacles is reformulated as

$$\min_{\mathbf{x},\mathbf{u},\lambda,\mu,\mathbf{s}} \quad \sum_{k=0}^{N-1} L(x_k, u_k) + L_f(x_N) + \sum_{k=0}^{N} \left[\kappa \cdot \sum_{m=1}^{M} s_k^{(m)}\right] \tag{3.5a}$$

$$\text{subject to} \quad x_{k+1} = F(x_k, u_k) \tag{3.5b}$$

$$x_0 = x(t_0) \tag{3.5c}$$

$$u_k \in \mathcal{U} \tag{3.5d}$$

$$-b^T \mu_k^{(m)} + (C^{(m)} T(x_k) - d^{(m)})^T \lambda_k^{(m)} > -s_k^{(m)} \tag{3.5e}$$

$$A^T \mu_k^{(m)} + R(x_k)^T C^T \lambda_k^{(m)} = 0 \tag{3.5f}$$

$$\left|\left|{C^{(m)}}^T \lambda_k^{(m)}\right|\right|_2 = 1 \tag{3.5g}$$

$$\lambda_k^{(m)} \geq 0 \tag{3.5h}$$

$$\mu_k^{(m)} \geq 0 \tag{3.5i}$$

$$s_k^{(m)} \geq 0 \tag{3.5j}$$

$, \forall k \in [0, N], \quad \forall m \in [1, M], \quad k, m \in \mathbb{N}.$

The optimization problem is solved using CasADi which is an open-source software for nonlinear optimization and algorithmic differentiation, see [18] for more details. Note that the optimization variables are a collection of states, control inputs, slack variables and dual variables associated with the obstacles' and vehicle's set. The collections for N steps are denoted by

$$\mathbf{x} = [x_0, ..., x_N], \quad \mathbf{u} = [u_0, ..., u_N], \quad \lambda^{(m)} = [\lambda_0^{(m)}, ..., \lambda_N^{(m)}],$$
$$\mu^{(m)} = [\mu_0^{(m)}, ..., \mu_N^{(m)}], \quad \mathbf{s}^{(m)} = [s_0^{(m)}, ..., s_N^{(m)}]$$

Given a collection of reference states $\mathbf{x}_{ref} = [x_{ref}^1, \cdots, x_{ref}^\eta]$ for the vehicle to follow, the obstacle avoidance NMPC is formulated such that the vehicle

follows one reference state at a time. Once the vehicle is nearby the reference state, the reference state is updated and the vehicle follows the next reference state in $\mathbf{x}_{ref}$. Therefore the reference state in the cost function (3.3) is constant for all N time steps in the NMPC optimization problem (3.5). The terminal cost function $L_f(x_N)$ in the NMPC optimization problem (3.5) is given by (3.4). Each obstacle has its slack variable $s_k^{(m)}$ at time step k, that describes the penetration depth between the obstacle and the vehicle sets. As seen in (3.5a), a weighted sum of all penetration depths of the obstacles for all N time steps is included in the optimization problem. Choosing a big value for κ will force the slack variables to be as small as possible where zero penetration depths can be achieved. Hence the slack variables will be activated only when the problem is infeasible, i.e. they will remain zero until the optimization problem cannot find a trajectory $\mathbf{x}$ that is collision free. However if the optimization problem cannot find collision free trajectory, a stop condition is introduced, $\frac{1}{N+1}\sum_{k=0}^{N}\left\{\frac{1}{M}\cdot\sum_{m=1}^{M}s_k^{(m)}\right\} > \Delta$. The stop conditions is based on the mean of all slack variables that represents the penetration between the vehicle's set (2.8) and an obstacle set (2.7). To guarantee safety and not be too conservative the Δ is added to the vehicles set in order to allow some penetration while collision is still avoided. However if the mean of all slack variables are greater than Δ, then the vehicle will collide and in turn a command is sent to stop the vehicle. The vehicle will not stop forever but it will still solve the NMPC optimization problem to find another trajectory that does not violate the stop condition and is collision free. An overview of the obstacle avoidance NMPC algorithm is illustrated in Algorithm 1.

Algorithm 1: Obstacle avoidance NMPC algorithm

```
while x_ref^η is not reached do
    Solve (3.5)
    if 1/(N+1) Σ_{k=0}^{N} { 1/M · Σ_{m=1}^{M} s_k^(m) } > Δ then
        Stop the vehicle
    else
        Apply optimal control u_0^*
    end
    if x_ref is near then
        Update x_ref
    end
end
```

3.2 Edge server

To compute the reachable set for a vehicle in CORA toolbox [13], the start time t_0, final time t_f, the vehicle model, initial set $\mathcal{X}_0$ and input trajectory $\mathbf{u}$ in $[t_0, t_f]$ should be known as shown in (2.18). The initial set of the vehicle is a zonotope and can be denoted as $\mathcal{X}_0 = (c, G)$. Since it is assumed that the vehicle knows its current state $x(t_0)$, the initial state is constructed such that the center of the zonotope is $c = x(t_0)$ and the generator matrix G consists of the uncertainty of each state. For example the vehicle's uncertainty in the x-position is the vehicles length plus some margin. The model of the vehicle is given by (3.2) and the input trajectory is the optimal control input trajectory $\mathbf{u}$ from the NMPC controller. As explained in subsection 2.2.2, the reachable set is that is constructed consists of $n = \frac{t_f - t_0}{h}$ reachable sets, where h is the time step. This can be expressed as

$$\mathcal{R}(x(t_0), \mathbf{u}, t_0, t_f, h) = \left\{ \mathcal{Z}_1, \cdots, \mathcal{Z}_n \right\} \tag{3.6}$$

The edge server computes the reachable set for a vehicle, converts the zonotopes to H-representation and then broadcasts them so that other vehicles use the zonotopes as obstacles in their obstacle avoidance NMPC algorithm.

Suppose there are two vehicles that have an intersecting trajectory and there is a high possibility for collision. If the vehicles know each other's future positions, then both can predict the collision and plan a safe trajectory accordingly. For the reachable sets to be useful for the vehicles to plan a safe trajectory, they should be accurate and not delayed. If the reachable sets are delayed and old then the collision free trajectory is planned based on old information, hence the safety is not guaranteed at the current time. To illustrate this, assume that vehicle A sends its current state and input trajectory at time $t = t_0$ to the edge server. The edge server receives the state of vehicle A at time $t = t_0 + \frac{\tau(X,Y)}{2}$, where $\tau(X, Y)$ is the round trip time of the network, i.e the network latency. The time it takes for the zonotopes (3.6) to be computed is t_c. Assume vehicle B is listening for the zonotopes of vehicle A, then the time it takes for the reachable set to transfer from the edge server to vehicle B is $\frac{\tau(X,Y)}{2}$. If the edge server computes the zonotopes for vehicle A such that $t_f = t_0 + \frac{\tau(X,Y)}{2} + t_c + \frac{\tau(X,Y)}{2}$, then vehicle B will receive the zonotopes describing the possible set of states vehicle A is in at the current time. However to guarantee safety, a time margin is added so that vehicle B knows the future positions of vehicle A instead of the current positions. Hence, the final time of the computed reachable sets is given by

$$t_f = t_0 + t_c + \tau(X, Y) + t_{margin} \tag{3.7}$$

3.3 Implementation overview

The implementation was done using Robot Operatin System, ROS, [19] where the edge server and the obstacle avoidance NMPC were developed and examined in a simulation environment.

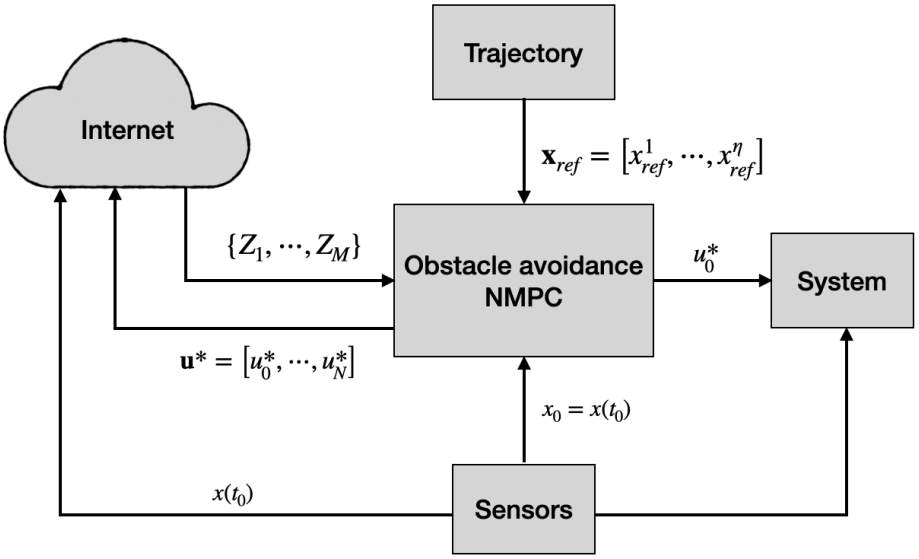


Figure 3.2: Overview of an autonomous vehicle

An overview of the implementation of an autonomous vehicle is illustrated in Figure 3.2. The internet in Figure 3.2 represents the communication link between the autonomous vehicle and the edge server. As mentioned in the section 1.2, the current state $x(t_0)$ and the original trajectory which the vehicle is following $\mathbf{x}_{ref}$ are known. The static obstacles, $\{\mathcal{Z}_1, \cdots, \mathcal{Z}_M\}$, given in H-Representation, are received from the edge server. They consist of reachable sets of other vehicles and obstacles in the environment such as sidewalks and construction sites. Given the known current state $x(t_0)$ from the sensors, the known trajectory $\mathbf{x}_{ref}$ and the obstacles $\{\mathcal{Z}_1, \cdots, \mathcal{Z}_M\}$ from the edge server, the obstacle avoidance NMPC finds an optimal control trajectory $\mathbf{u}^*$ according to Algorithm 1. Since the optimization problem (3.5) assumes static obstacles, those obstacles, particularly the reachable sets, are updated frequently in the optimization problem in order to account for moving obstacles. As shown in

Figure 3.2 the autonomous vehicle sends its current state $x(t_0)$ and the current optimal control trajectory $\mathbf{u}^*$ to the edge server so that the edge server computes its reachable sets and broadcasts them to other autonomous vehicles.

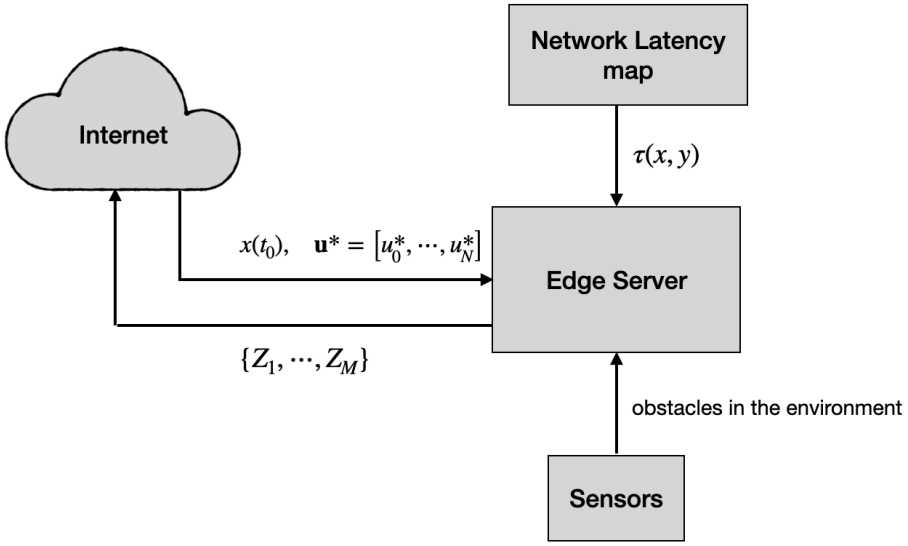


Figure 3.3: Overview of the edge server

As illustrated in Figure 3.3, the edge server receives the current state of an autonomous vehicle, $x(t_0)$ and its optimal control trajectory $\mathbf{u}^*$. Considering that the latency of the network, $\tau(X, Y)$, is known from a network latency map, the edge server computes the reachable sets of the autonomous vehicle as (3.6). Assuming the sensor provides information of the obstacles in the environment to the edge server, those obstacles together with the reachable sets of an autonomous vehicle are broadcasted to other vehicles as $\{\mathcal{Z}_1, \cdots, \mathcal{Z}_M\}$. The edge server frequently computes the reachable sets of the vehicles and broadcast them in order to account for moving vehicles.

The accuracy of the computation of reachable sets is an important factor for guaranteeing safety. To accurately compute the reachable set, a small time step h should be used in (3.6). However too small time step will increase the computational time, t_c, of the reachable set. Therefore the time step, h is chosen in a way that minimizes the computational time while preserving the accuracy to some degree. The choice of h, will be shown in the Results. Another param-

eter that affects the accuracy of the reachable sets is the zonotope order ρ, see (2.16). If the zonotope order is 1 then the number of generators is equal to the dimension of the zopnotope, as seen in Figure 2.1 for a two dimensional zonotope. The shape of the reachable set can be expressed more accurately if there are more generators. However increasing the zonotope order is meaningful to some extent, because suppose a reachable set have a square form with rounded edges. Then increasing the zonotope order will express the reachable more accurately and result in many vertices close to each other to resemble the rounded edge. The complexity of the zonotope will increase as the zonotope order increases and the computation of the reachable set is less efficient. Therefore choosing the zonotope order will be based on minimizing the complexity but still representing the reachable set accurately to some extent. In [20] several zonotope order reduction techniques are presented and compared with each other. The idea of the reduction technique is to reduce the order of the zonotope to a smaller order with a zonotope that is over-approximated as tight as possible. One reason to use zonotope order reduction in the implementation is to reduce the computational time of the obstacle avoidance NMPC algorithm. Recall that the zonotopes can be represented in the H-representation as explained in subsection 2.2.1. A large zonotope order results in a large number of generators which in turn lead to large number of rows in the matrix C. The amount of optimization variables in the vector λ_k is the same as the number of rows in the matrix C, see (3.5g). In conclusion reducing the order of the zonotope, reduces the number of optimization variables in the obstacle avoidance NMPC which reduces the complexity of the optimization problem (3.5) and in turn decreases the computational time. The choice of reduction technique will be presented and justified in the Results.

Chapter 4

Results

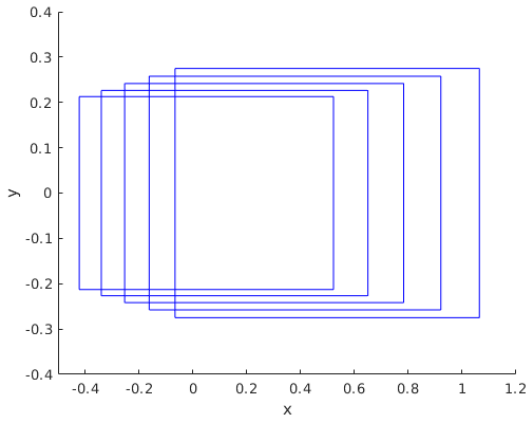
In this chapter the results for the justification of the choice of some parameters are shown. In addition simulation results of the implementation are presented for different cases.

4.1 Parameters evaluation

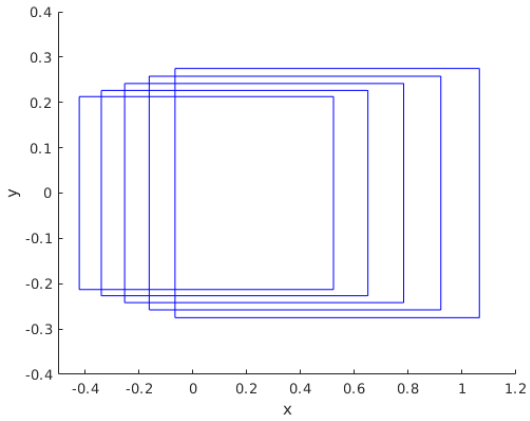
4.1.1 Time step for the reachable set computation

The comparison between different time steps, h, for the reachable set computation is presented here. The reachable sets are computed for the time horizon $t_f = t_0 + 0.5$ s and two different input trajectories $u_1 = [0, 0.5]^T$ and $u_2 = [\frac{\pi}{4}, 0.5]^T$. The initial state of the vehicle is $x(t_0) = [0, 0, 0, 1]^T$, where the vehicle has an initial velocity. The two different inputs were chosen to examine two extreme cases. The first one, u_1, is the vehicle driving straight with no steering but only accelerating with $0.5m/s^2$. The second one, u_2, is the vehicle accelerating with $0.5m/s^2$ with maximum steering, $\delta = \frac{\pi}{4}$. The initial set of the vehicle is given by

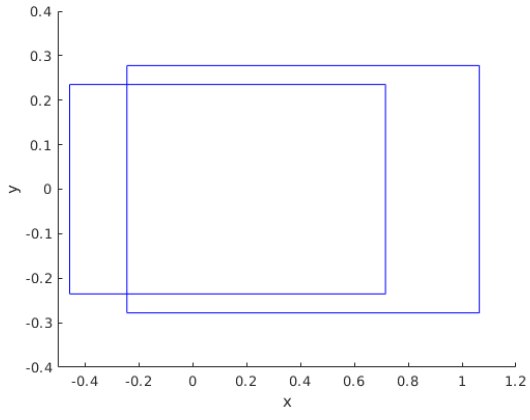
$$\mathcal{X}_0 = (c, G), \quad c = x(t_0), \quad G = \begin{bmatrix} 0.4 & 0 & 0 & 0 \\ 0 & 0.2 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0.2 \end{bmatrix}$$



(a) $h = 0.05, \quad t_c = 303$ ms

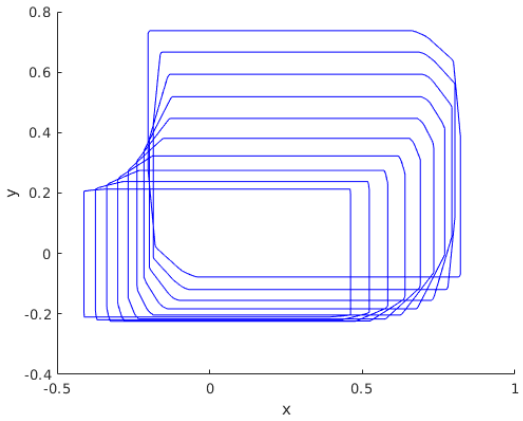


(b) $h = 0.1, \quad t_c = 246$ ms

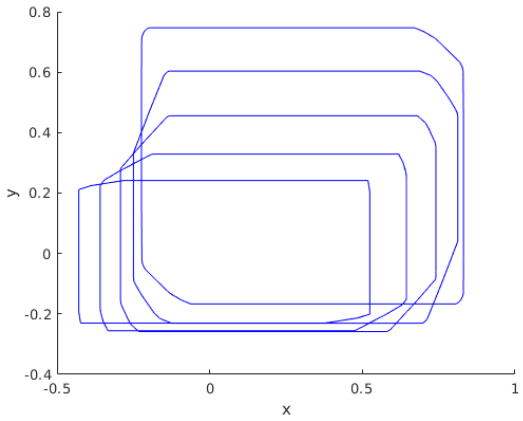


(c) $h = 0.25, \quad t_c = 182$ ms

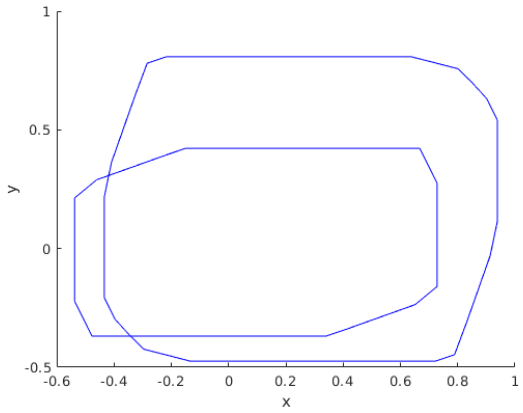
Figure 4.1: Reachable set computation for 0.5 s time horizon and 0 steering angle



(a) $h = 0.05, \quad t_c = 340$ ms



(b) $h = 0.1, \quad t_c = 258$ ms



(c) $h = 0.25, \quad t_c = 203$ ms

Figure 4.2: Reachable set computation for 0.5 s time horizon and 45 steering angle

For the case where the steering angle is 0, as seen in Figure 4.1, the reachable set describe the same area where the vehicle will be in 0.5 seconds. However as expected the number of zonotopes are less for bigger time steps, which in turn result in less computational time. Considering the second case, see Figure 4.2, the reachable sets are different for different time steps. The reachable sets are less accurate for bigger time steps when the vehicle has more complex motion. However the accuracy of the reachable set in Figure 4.2b is more or less similar compared to the reachable set in Figure 4.2a. Since the time step is less, the computational time is less because less zonotopes are computed. Moreover decreasing the number of zonotopes will decrease the number of obstacles in the obstacle avoidance NMPC which in turn decreases the computational time of the obstacle avoidance NMPC algorithm. Therefore the time step $h = 0.1$ was chosen for the computation of the reachable set.

4.1.2 Zonotope reduction technique

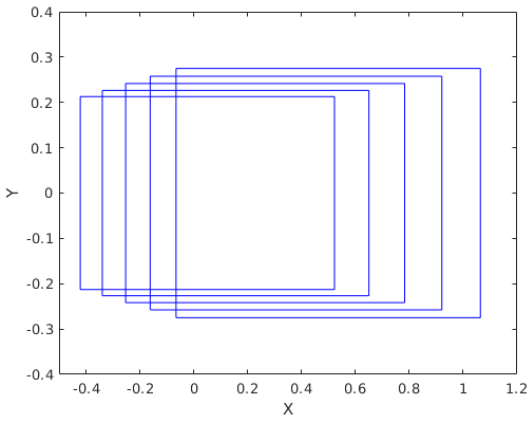
Although there are many different techniques to reduce the order of zonotopes, only the box method and PCA method are compared here which are fast algorithms. The comparison is done for two cases with different input trajectories:

$$\begin{aligned} &\text{case 1:} \quad \mathcal{R}_1(x(t_0), u_1, t_0, t_f, h) \\ &\text{case 2:} \quad \mathcal{R}_2(x(t_0), u_2, t_0, t_f, h) \end{aligned}$$

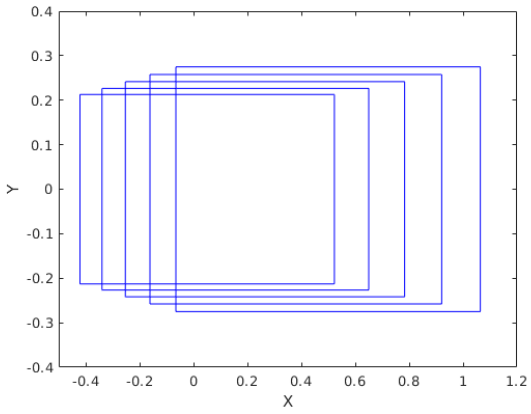
, where

$$h = 0.1, \quad t_0 = 0, \quad t_f = 0.5, \quad u_1 = \begin{bmatrix} 0 \\ 0.5 \end{bmatrix}, \quad u_2 = \begin{bmatrix} \frac{\pi}{4} \\ 0.5 \end{bmatrix},$$

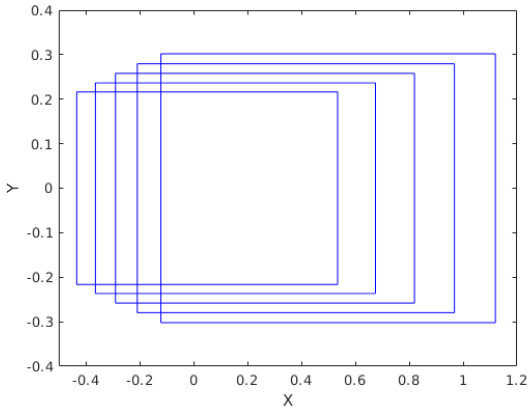
$$\mathcal{X}_0 = (c, G), \quad c = x(t_0) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \quad G = \begin{bmatrix} 0.4 & 0 & 0 & 0 \\ 0 & 0.2 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0.2 \end{bmatrix}$$



(a) Without reduction

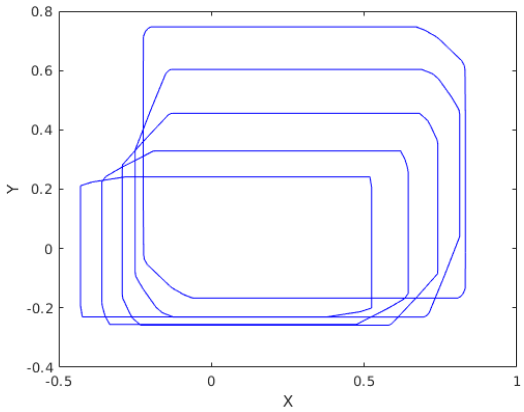


(b) Box reduction method

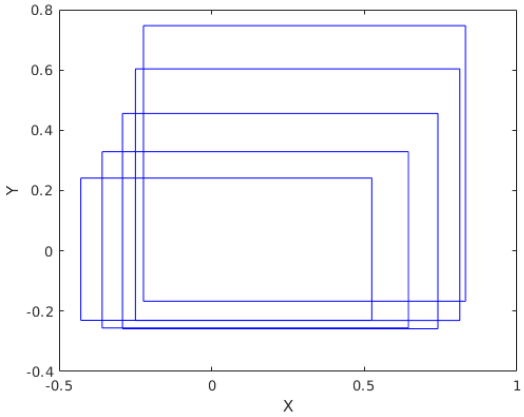


(c) PCA reduction method

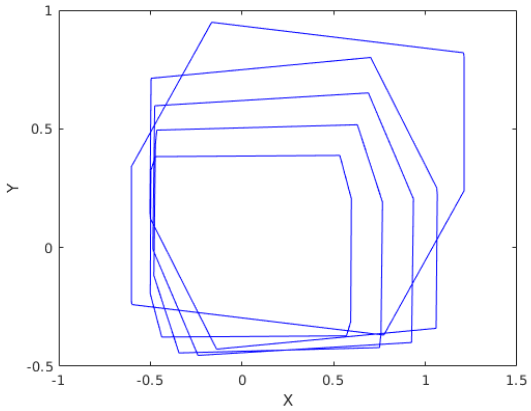
Figure 4.3: Reduction techniques comparison for reachable set, projected on the X and Y states, of case 1.



(a) Without reduction



(b) Box reduction method



(c) PCA reduction method

Figure 4.4: Reduction techniques comparison for reachable set, projected on the X and Y states, of case 2.

Comparing the reachable sets in Figure 4.3 for case 1, both the box method and the PCA method have similar accuracy, in other words describe a similar area when the zonotope is projected on the X and Y states. Although the look similar they have different zonotope order, ρ, where the zonotopes without reduction have 20 generators, i.e $\rho = 5$, while both the box method and the PCA method reduce the zonotopes to order $\rho = 1$, i.e have 4 generators. Not only the zonotope order are different but the number of vertices, where a zonotope in Figure 4.3a 36 vertices, while a zonotope in Figure 4.3b has 4 vertices and a zonotope in Figure 4.3 has 9 vertices. Since the zonotopes have similar accuracy in case 1, case 2 is examined where a reachable set is computed for more complex vehicle motion. In Figure 4.4 it is seen that the box method is more accurate than the PCA method. Furthermore, as seen in Figure 4.5, the box method reduces the zonotope order and over-approximate the zonotope accurately. The over-approximation is not as conservative as the PCA method, and also the number of vertices is reduced from 36 to 4 when using the box method. Reducing the zonotope order and number of vertices in turn decreases considerably the computational time for the obstacle avoidance NMPC while safety is guaranteed since the reduced zonotope is an over-approximation. Therefore the box reduction technique is used in the edge server and the reduced zonotopes are broadcasted to the autonomous vehicles.

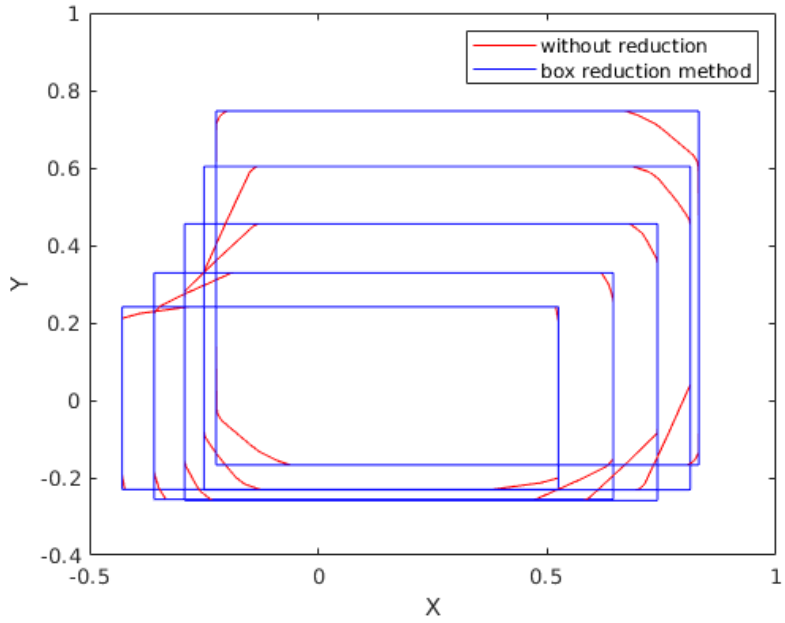


Figure 4.5: Illustration of how the box method reduces the zonotope order.

4.2 Simulation

To evaluate the edge server and obstacle avoidance NMPC algorithm the scenario described in subsection 1.1 will be examined. Figures that capture the evolution of the simulation in different time instances are presented to illustrate the scenario. The scenario is divided in different cases to demonstrate how the autonomous vehicles behave with and without the edge server and obstacle avoidance NMPC implementations. The following parameters are common for all cases:

$$N = 25, \quad \kappa = 500, \quad \rho = 5, \quad h = 0.1,$$

$$Q_1 = \begin{bmatrix} 5 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 4 \end{bmatrix}, \quad Q_2 = \begin{bmatrix} 4 & 0 \\ 0 & 2 \end{bmatrix}, \quad Q_f = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 6 \end{bmatrix}$$

Note that the computational time for reachable sets varied between 180 ms and 250 ms, while the network latency varied between 10 ms and 50 ms. Since the difference is big, a fake delay was added in the implementation to mimic the network latency $\tau(X, Y)$. Throughout the simulation, two autonomous vehicles are considered: vehicle A which is the blue vehicle in the simulation and vehicle B which is the green vehicle in the simulation. Figure 4.6 illustrates the scenario in the simulation environment. The red polygons are obstacles in the environment and represent the sidewalks and the construction site as shown in Figure 1.1.

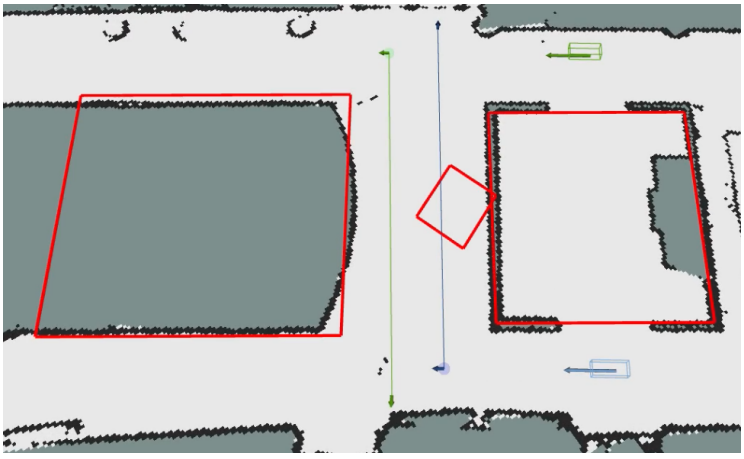


Figure 4.6: The scenario in the simulation environment.

Case 1:

In this case the edge server and the obstacle avoidance NMPC are implemented. However the edge server does not consider the latency properly in the reachable set computation. The network latency for both vehicles are assumed to be $\tau(X, Y) = 0.1$. The computational time for the reachable set vehicle A is $t_c^A = 0.17$ and the computation time for the reachable set of vehicle B is $t_c^B = 0.15$. The edge server computes the reachable set as (3.6) with t_f as (3.7), but in this case the final time for both vehicles is chosen as $t_f = t_0 + 0.1$.

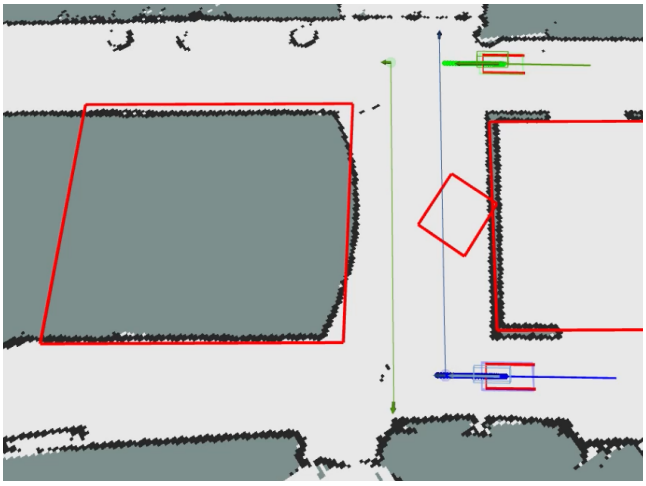


Figure 4.7: Case 1 at time instance t_1

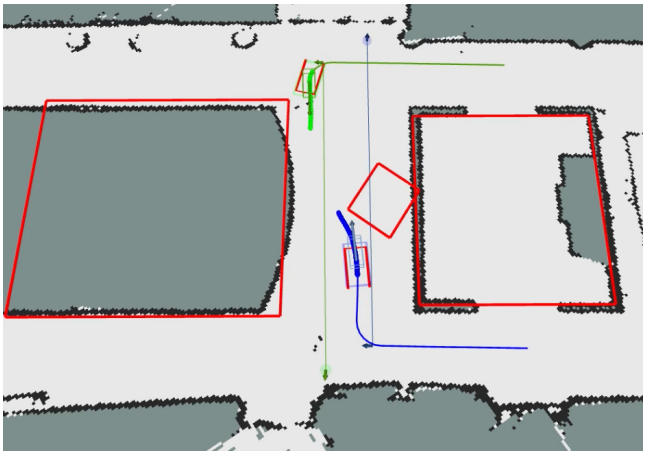


Figure 4.8: Case 1 at time instance t_2

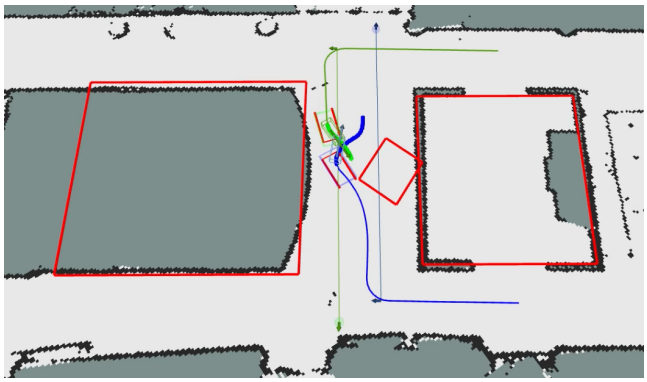


Figure 4.9: Case 1 at time instance t_2

Figure 4.8 illustrates how the obstacle avoidance NMPC for vehicle A is planning to avoid the construction site obstacle, where the colored circles represents the planned states by the obstacle avoidance NMPC. The latency caused by the computational time t_c and the network latency $\tau(X, Y)$ can be seen in Figures 4.7 - 4.9, where the red open rectangle represents the vehicle's set received in the edge server. It is seen that the vehicles' current states are not the same as the ones received by the edge server. Note that the edge server computes the reachable sets for the delayed states and therefore for the time horizon 0.1 s, the reachable sets do not cover the actual vehicles' current states. Since the obstacle avoidance NMPC of vehicle B tries to avoid the reachable sets of vehicle A, the planned states of the NMPC will not be accurate because it is based on the delayed position of vehicle A. This can be seen in Figure 4.9, where the NMPC of vehicle B plans to avoid the reachable set of vehicle A but a collision has already occurred. They are in contact where the distance between the vehicles is zero, see Figure 4.10. The distance between the vehicles is given by the minimum distance between the vehicles' sets as explained in subsection 3.1.2. Therefore the unsafe distance is chosen as 0.1 since the dimension of the vehicles in the simulation is 0.25 x 0.5. If the distance between the vehicles is less than 0.1 then a collision or an unsafe situation has occurred.

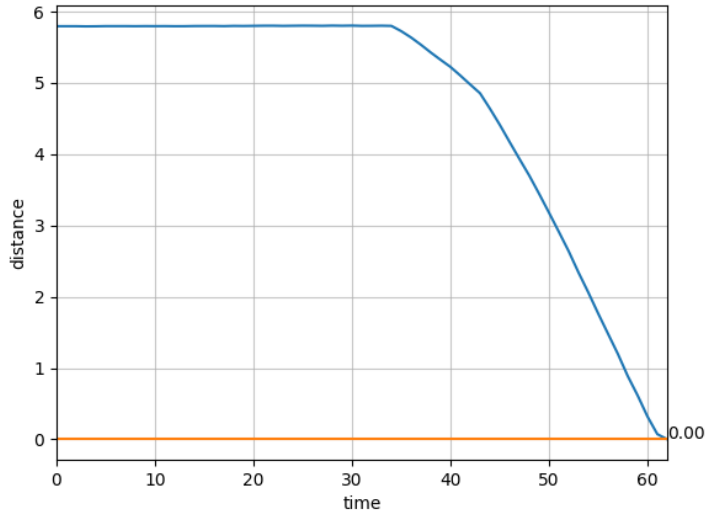


Figure 4.10: The distance between the vehicles over time for case 1.

Case 2:

For this case the latency is properly considered in the computation of reachable sets. The network latency is assumed, the same as in case 1, to be $\tau(X, Y) = 0.1$ for both vehicles. The computational time for both vehicles' reachable sets is $t_c^A = t_c^B = 0.22$. The reachable sets for both vehicles are both computed for $t_f = t_0 + 0.4$, where the margin time is chosen as $t_{margin} = 0.08$.

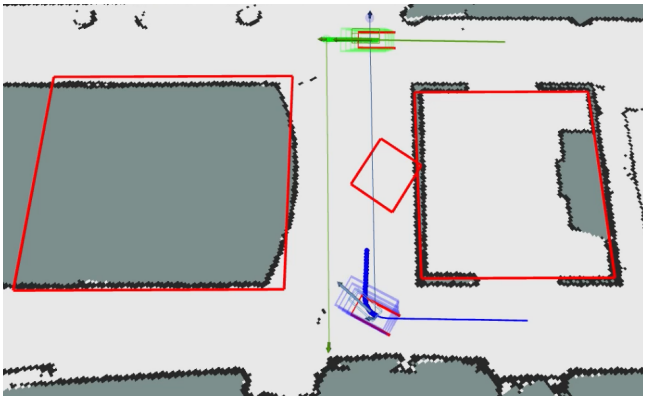


Figure 4.11: Case 2 at time instance t_1

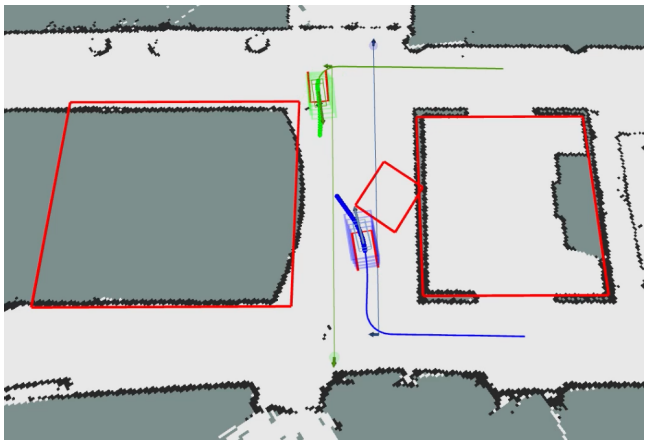


Figure 4.12: Case 2 at time instance t_2

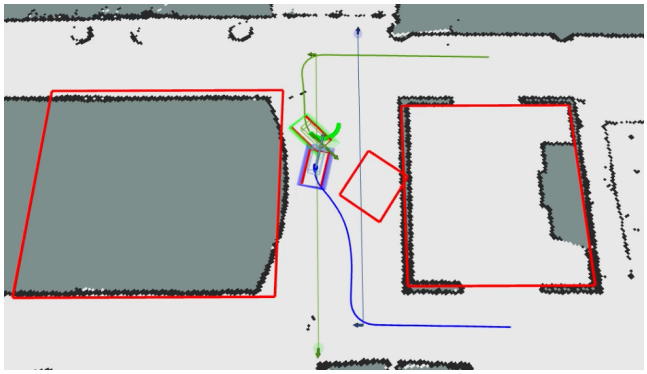


Figure 4.13: Case 2 at time instance t_2

In Figure 4.11 one can see that although there is delay between the vehicles and the edge server, the reachable sets are computed such that the delay is taken into account where they cover the current states of the vehicles. However in some cases, which is illustrated in Figure 4.13, the vehicles get stuck and it takes awhile to avoid each other. This happens when the margin time is small which was in this case $t_{margin} = 0.08$. If the time margin is too small then the reachable set will only describe where the vehicle's state at the current time and not in the future. Assuming that the vehicles are close to each other then the obstacle avoidance NMPC will not have much time to plan and cannot find a collision free trajectory. This can be shown in Figures 4.14 - 4.16. When the vehicles are close to each other, see Figure 4.14, then the obstacle avoidance NMPC plans a trajectory that intersects with the reachable sets and forces the slack variables to be greater than zero, see Figure 4.16. When the stop value

increases then, as shown in Algorithm 1, a command is sent to the vehicle to stop which can be seen in Figure 4.15. It can be seen that vehicle B stops first and then vehicle A which corresponds to order of their stop value shown in Figure 4.16. Note that both vehicles are not in within unsafe distance where the minimum distance, seen in Figure 4.14, is 0.2.

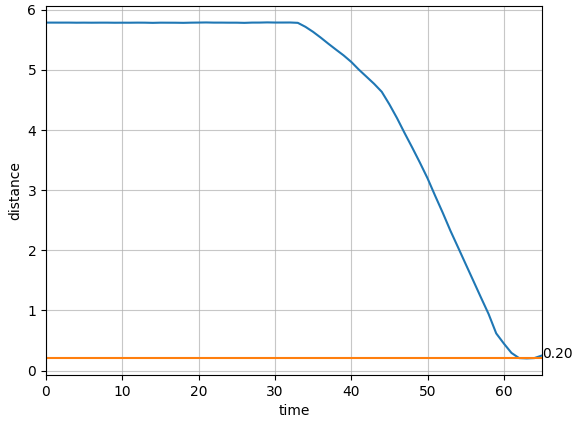


Figure 4.14: The distance between the vehicles over time for case 2.

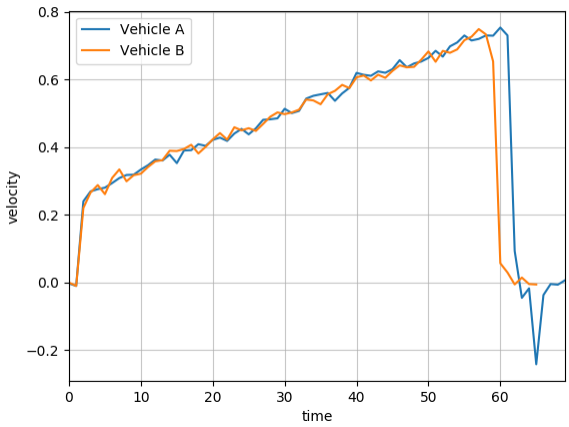


Figure 4.15: The velocity of the vehicles over time for case 2.

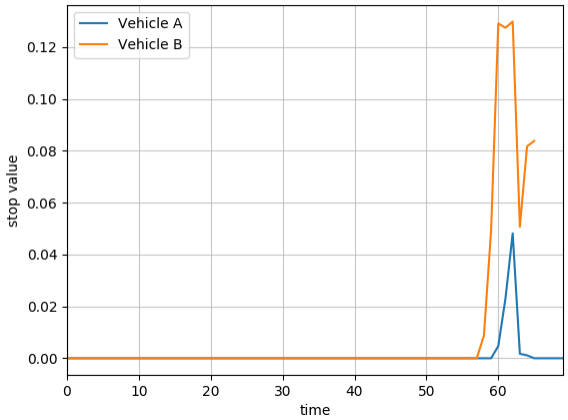


Figure 4.16: The stop value, $\frac{1}{N+1}\sum_{k=0}^{N}\left\{\frac{1}{M}\cdot\sum_{m=1}^{M}s_k^{(m)}\right\}$, for the vehicles over time for case 2.

Case 3:

Here the network latency is assumed to be different for each vehicle, where $\tau^A = 0.2$, $\tau^B = 0.1$. The computational time for the vehicles' reachable sets is $t_c^A = 0.28$, $t_c^B = 0.21$. The final time for the reachable set computation is different where $t_f = t_0 + 0.7$ for vehicle A and $t_f = t_0 + 0.5$ for vehicle B. The time margin in this case is ca $t_{margin} = 0.2$ for both vehicles.

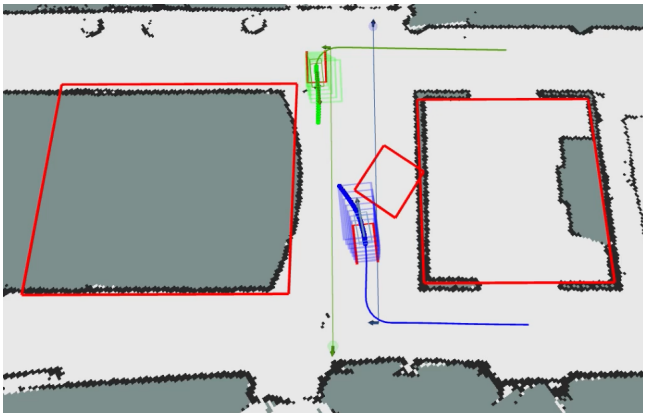


Figure 4.17: Case 3 at time instance t_1

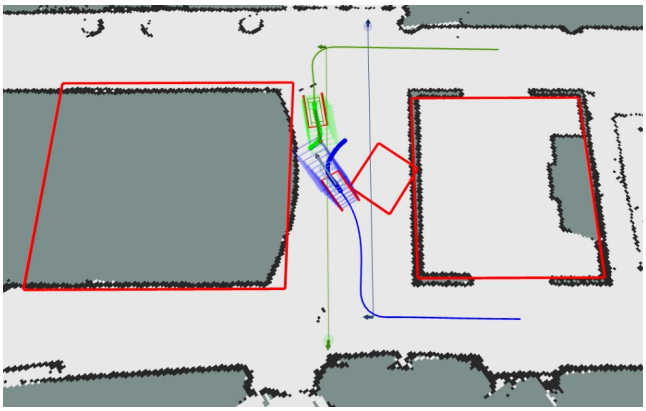


Figure 4.18: Case 3 at time instance t_2

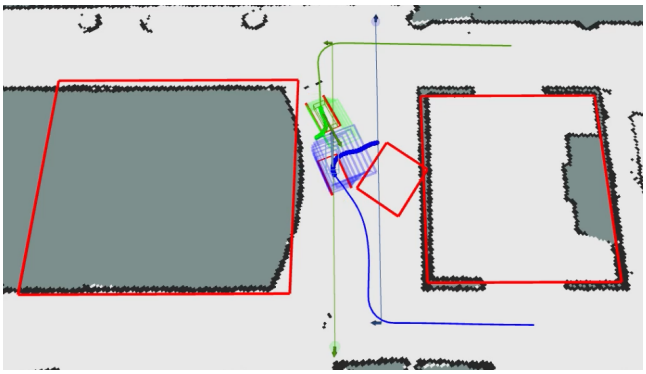


Figure 4.19: Case 3 at time instance t_3

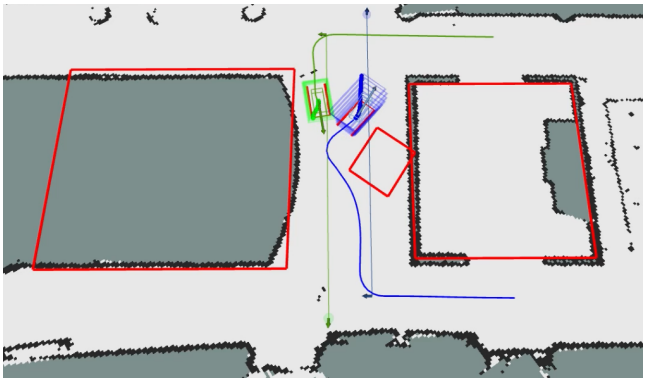


Figure 4.20: Case 3 at time instance t_5

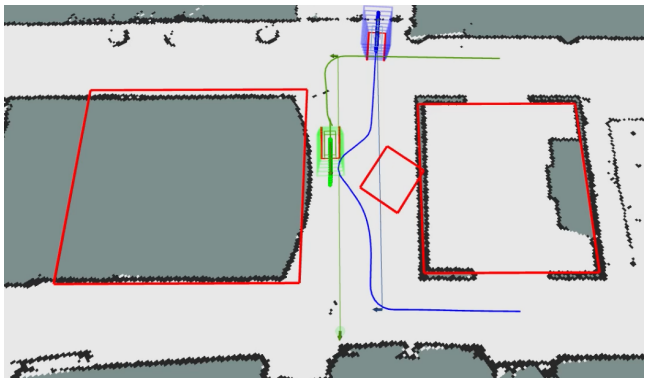


Figure 4.21: Case 3 at time instance t_6

Seeing Figures 4.17 - 4.21 both vehicles succeed in passing each other safely. In Figure 4.18 and Figure 4.19, it can be seen that the obstacle avoidance NMPC of vehicle B tries to find a collision free trajectory and avoid the zonotopes of vehicle A. However, as shown in Figures 4.22 - 4.24, vehicle B stops since it is close to vehicle A and the mean value of all slack variables is greater than zero. Vehicle A also stops because the predicted trajectory was not collision free, see Figure 4.24. Thereafter in Figure 4.20 it is seen that vehicle B has revered which is also illustrated in Figure 4.23. This makes way for vehicle A to pass vehicle B and as seen in Figure 4.21, where both vehicles pass each other safely. The vehicles were not within an unsafe distance with each other, as seen in Figure 4.22, where the minimum distance is 0.24.

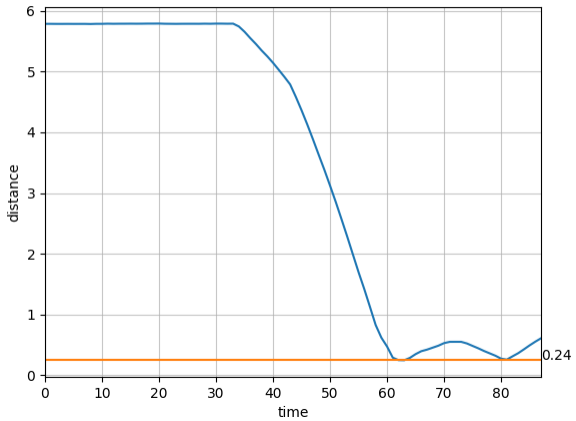


Figure 4.22: The distance between the vehicles over time for case 3.

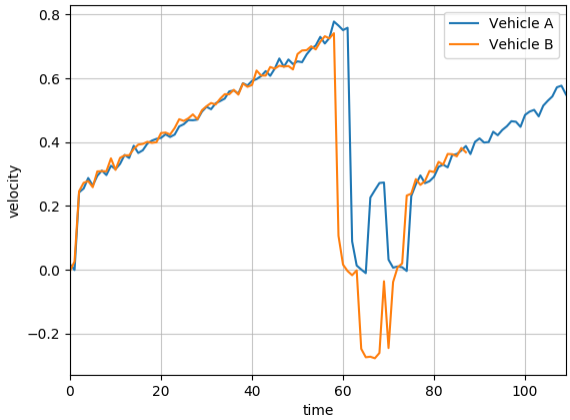


Figure 4.23: The velocity of the vehicles over time for case 3.

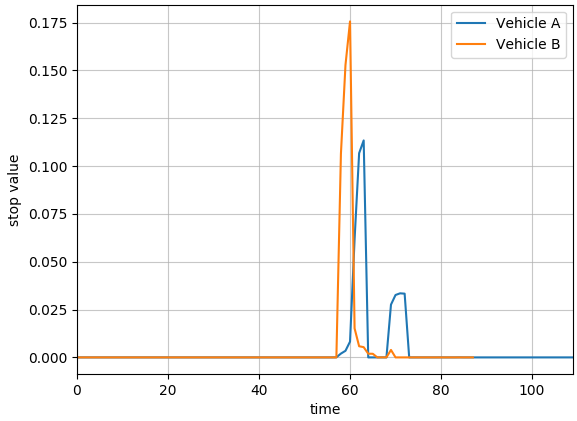


Figure 4.24: The stop value, $\frac{1}{N+1}\sum_{k=0}^{N}\left\{\frac{1}{M}\cdot\sum_{m=1}^{M}s_k^{(m)}\right\}$, for the vehicles over time for case 3.

Chapter 5

Conclusions

The objective of this thesis was to implement a latency-aware edge server and obstacle avoidance NMPC for autonomous vehicles in an intersection scenario to achieve safe autonomous driving. As demonstrated in the results, implementing obstacle avoidance NMPC, the autonomous vehicle was able to avoid static obstacles and even if the NMPC could not find a collision free trajectory a stop condition was implemented that still guarantees safety. From the simulation results it was shown that the latency-aware edge server provides useful information in the form of reachable sets of autonomous vehicles in the intersection. Using these information in the obstacle avoidance NMPC algorithm, the autonomous vehicles were able to avoid collision and achieve safe autonomous driving. Although there were some cases where the vehicles could not move forward and continue on their original trajectory or got stuck and it took awhile for them to pass each other, the implementation was overall successful in the scenario presented in the problem formulation.

Chapter 6

Future Work

The implementation was examined in simulation and an extension could be to implement it on a real system for example on a F1/10 RC car. The vehicle's sensors could be used along with the obstacle avoidance NMPC. However this will introduce some uncertainty on the stetes and there could be some disturbances caused by modelling the system in the NMPC. One way to handle disturbed nonlinear systems is using reach-set MPC as explain in [21]. The authors uses reachability analysis with MPC to implement controller that guarantees safety despite disturbances and measurement noise. The obstacle avoidance NMPC algorithm was computational demanding, in spite of different techniques that were used to decrease the complexity of the obstacles and in turn decrease the computational time of the NMPC. To be able to implement it on a real system and achieve a fast solution one should either find a better and more efficient solver than CasADi or change the formulation of the MPC. Instead of NMPC one could use a linear time-varying MPC as presented in [22], where the system is linearized around a reference trajectory. However the obstacle avoidance conditions introduces non convexity of the optimization problem and should be reformulated. Another extension to this thesis could be to implement the sensors and sensor processing for the edge server for example implementing a camera that detects obstacles via deep learning or other techniques. Another example could be if the computational time of the reachable sets is less, then one could model the network latency as a coverage map and implement a function of the network latency.

Bibliography

[1] Frank J. Jiang et al. “Human-Centered Design for Safe Teleoperation of Connected Vehicles**This work is supported by the Swedish Strategic Research Foundation, the Swedish Research Council, and the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.” In: *IFAC-PapersOnLine* 53.5 (2020). 3rd IFAC Workshop on Cyber-Physical & Human Systems CPHS 2020, pp. 224–231. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2021.04.101. URL: https://www.sciencedirect.com/science/article/pii/S2405896321001944.

[2] Bogdan Antonescu, Miead Tehrani Moayyed, and Stefano Basagni. “mmWave channel propagation modeling for V2X communication systems”. In: *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*. 2017, pp. 1–6. DOI: 10.1109/PIMRC.2017.8292718.

[3] Rolf Findeisen and Frank Allgöwer. “An Introduction to Nonlinear Model Predictive Control”. In: Jan. 2002.

[4] H. Chen and F. Allgöwer. “Nonlinear Model Predictive Control Schemes with Guaranteed Stability”. In: *Nonlinear Model Based Process Control*. Ed. by Ridvan Berber and Costas Kravaris. Dordrecht: Springer Netherlands, 1998, pp. 465–494. ISBN: 978-94-011-5094-1. DOI: 10.1007/978-94-011-5094-1_16. URL: https://doi.org/10.1007/978-94-011-5094-1_16.

[5] Lars Grüne and Jürgen Pannek. “Nonlinear Model Predictive Control”. In: *Nonlinear Model Predictive Control: Theory and Algorithms*. London: Springer London, 2011, pp. 43–66. ISBN: 978-0-85729-501-9. DOI: 10.1007/978-0-85729-501-9_3. URL: https://doi.org/10.1007/978-0-85729-501-9_3.

[6] Björn Lindqvist et al. "Nonlinear MPC for Collision Avoidance and Control of UAVs With Dynamic Obstacles". In: *IEEE Robotics and Automation Letters* 5.4 (2020), pp. 6001–6008. DOI: 10.1109/LRA.2020.3010730.

[7] *A Multi-Stage Optimization Formulation for MPC-Based Obstacle Avoidance in Autonomous Vehicles Using a LIDAR Sensor*. Vol. Volume 2: Dynamic Modeling and Diagnostics in Biomedical Systems; Dynamics and Control of Wind Energy Systems; Vehicle Energy Management Optimization; Energy Storage, Optimization; Transportation and Grid Applications; Estimation and Identification Methods, Tracking, Detection, Alternative Propulsion Systems; Ground and Space Vehicle Dynamics; Intelligent Transportation Systems and Control; Energy Harvesting; Modeling and Control for Thermo-Fluid Applications, IC Engines, Manufacturing. Dynamic Systems and Control Conference. V002T30A006. Oct. 2014. DOI: 10.1115/DSCC2014-6269. eprint: https://asmedigitalcollection.asme.org/DSCC/proceedings-pdf/DSCC2014/46193/V002T30A006/4445281/v002t30a006-dscc2014-6269.pdf. URL: https://doi.org/10.1115/DSCC2014-6269.

[8] Xiaojing Zhang, Alexander Liniger, and Francesco Borrelli. *Optimization-Based Collision Avoidance*. 2018. arXiv: 1711.03449 [math.OC].

[9] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. DOI: 10.1017/CBO9780511804441.

[10] Frank J. Jiang et al. "Ensuring safety for vehicle parking tasks using Hamilton-Jacobi reachability analysis". In: *2020 59th IEEE Conference on Decision and Control (CDC)*. 2020, pp. 1416–1421. DOI: 10.1109/CDC42340.2020.9304186.

[11] Matthias Althoff and John M. Dolan. "Reachability computation of low-order models for the safety verification of high-order road vehicle models". In: *2012 American Control Conference (ACC)*. 2012, pp. 3559–3566. DOI: 10.1109/ACC.2012.6314777.

[12] Matthias Althoff. *On Computing the Minkowski Difference of Zonotopes*. 2016. arXiv: 1512.02794 [cs.CG].

[13] Matthias Althoff. "An Introduction to CORA 2015". In: Jan. 2015. DOI: 10.29007/zbkv.

[14] Matthias Althoff. "Reachability Analysis and its Application to the Safety Assessment of Autonomous Cars". PhD thesis. July 2010.

[15] P. Falcone et al. “A linear time varying model predictive control approach to the integrated vehicle dynamics control problem in autonomous systems”. In: *2007 46th IEEE Conference on Decision and Control*. 2007, pp. 2980–2985. DOI: 10.1109/CDC.2007.4434137.

[16] Jason Kong et al. “Kinematic and dynamic vehicle models for autonomous driving control design”. In: *2015 IEEE Intelligent Vehicles Symposium (IV)*. 2015, pp. 1094–1099. DOI: 10.1109/IVS.2015.7225830.

[17] Makoto Obayashi, Keisuke Uto, and Gaku Takano. “Appropriate overtaking motion generating method using predictive control with suitable car dynamics”. In: *2016 IEEE 55th Conference on Decision and Control (CDC)*. 2016, pp. 4992–4997. DOI: 10.1109/CDC.2016.7799032.

[18] Joel A E Andersson et al. “CasADi – A software framework for nonlinear optimization and optimal control”. In: *Mathematical Programming Computation* 11.1 (2019), pp. 1–36. DOI: 10.1007/s12532-018-0139-4.

[19] Stanford Artificial Intelligence Laboratory et al. *Robotic Operating System*. Version ROS Melodic Morenia. May 23, 2018. URL: https://www.ros.org.

[20] Anna-Kathrin Kopetzki, Bastian Schürmann, and Matthias Althoff. “Methods for order reduction of zonotopes”. In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. 2017, pp. 5626–5633. DOI: 10.1109/CDC.2017.8264508.

[21] Bastian Schürmann, Niklas Kochdumper, and Matthias Althoff. “Reach-set Model Predictive Control for Disturbed Nonlinear Systems”. In: *2018 IEEE Conference on Decision and Control (CDC)*. 2018, pp. 3463–3470. DOI: 10.1109/CDC.2018.8619781.

[22] Pedro F. Lima, Jonas Mårtensson, and Bo Wahlberg. “Stability conditions for linear time-varying model predictive control in autonomous driving”. In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. 2017, pp. 2775–2782. DOI: 10.1109/CDC.2017.8264062.

Appendix A

Something Extra