

MASTER

Context-driven motion planning at non-equipped intersections

Steenhof, E.P.

Award date:
2021

[Link to publication](#)

Disclaimer

This document contains a student thesis (bachelor's or master's), as authored by a student at Eindhoven University of Technology. Student theses are made available in the TU/e repository upon obtaining the required degree. The grade received is not published on the document as presented in the repository. The required complexity or quality of research of student theses may vary by program, and the required minimum study period may vary in duration.

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain



Integrated Vehicle Safety

**Context-driven motion planning at non-equipped
intersections**
Master Thesis

E.P. Steenhof
0856593

Master: Automotive Technology
Department: Mechanical Engineering
Research Group: Control Systems Technology

TUe Supervisors: prof. dr. ir. W.P.M.H Heemels
dr. ir. E. Silvas

External committee: dr. ir. M. C. F. Donkers

TNO Supervisors: ir. N.J. Brouwer
ir. B.A. Kupers

Report ID: CST2021.013

Eindhoven, April 2021

Declaration concerning the TU/e Code of Scientific Conduct for the Master's thesis

I have read the TU/e Code of Scientific Conduct¹.

I hereby declare that my Master's thesis has been carried out in accordance with the rules of the TU/e Code of Scientific Conduct

Date

16-03-2021


Name

Eric Pascaal Steenhof

ID-number

0856593

Signature



Insert this document in your Master Thesis report (2nd page) and submit it on Sharepoint

¹ See: <http://www.tue.nl/en/university/about-the-university/integrity/scientific-integrity/>

The Netherlands Code of Conduct for Academic Practice of the VSNU can be found here also.
More information about scientific integrity is published on the websites of TU/e and VSNU

Acknowledgement

Writing this last section brings me to the end of my master thesis and my student life. I started this thesis with COVID-19 being a virus far away from the Netherlands. However, in the second month of my thesis things changed rapidly. From going to the TNO office every day and playing Ping-Pong with colleagues during lunch break to working from home alone for months. I can look back on the last one year as a tough, but very productive period and as one full of learning.

Working at TNO has shown me how a research company operates and also helped immensely in shaping me as a professional. I would also like to take this opportunity to express my gratitude to all the people I have met at TNO and to thank TNO for giving me this opportunity.

First of all, I want to thank my supervisors for all the support they have provided me throughout the thesis.

Maurice Heemels, your enthusiasm for the field of systems and control, and inspiring ideas is something that motivated me a lot. Thank you for all your valuable feedback throughout my thesis and to always challenge me to improve.

Emilia Silvas, your eye for detail and the ability to catch the smallest of the missing links, while also managing your TNO work, is something which has always fascinated me and I highly look up to. I am really grateful for all the Teams coffee meetings we had about my thesis, life and the future.

Jochem Brouwer, your technical insights were really helpful for me throughout the thesis and really improved my understanding in the field of control. Furthermore, you made me realize I should be proud of what I have achieved in my thesis and that really helped my confidence.

Berend Kupers, your vision with regards to the practical aspects of the thesis really helped me understand on what I should focus on and what is required of me. I am really thankful for your support. Also, I really appreciated, the unfortunately few coffee corner talks we all had at TNO to clear my doubts within short notices. You both did a fantastic job in supervising me even in this crazy and difficult COVID-19 period. You both found the right balance of assisting me while still challenge me. Thank you both very much for all the effort you put in my thesis.

Next, I want to thank my friends, for all their support and wonderful moments created throughout the duration of my student life, really appreciated. Special thanks to Niels Lodder for challenging me, providing helpful tips and for the office banter, really made working from home easier. I would also like to specially thank my dear friend Daniël de Snoo for being a big emotional support.

Finally, I want to sincerely thank my parents for all the unconditional love and support they have provided me over the years. It is because of your hard work and sincere support I have reached this milestone today. I am truly grateful for that and I want to thank them both for inspiring me to become an engineer.

CONTENTS

| | | |
|-------------------|--|----|
| I | Introduction | 2 |
| I-A | Contributions and proposed approach | 3 |
| II | Problem formulation | 3 |
| II-A | Automated vehicle architecture | 4 |
| III | State of the art | 4 |
| III-A | Behavioral planners reflection | 4 |
| III-A1 | Finite State-Machine | 5 |
| III-A2 | Reinforcement Learning | 5 |
| III-A3 | Partially Observable Markov Decision Process | 5 |
| III-A4 | Inverse Reinforcement Learning | 5 |
| IV | Decentralized context-based MPC motion planner | 6 |
| IV-A | Perception block | 6 |
| IV-A1 | Environmental perception | 6 |
| IV-A2 | Context information categorization | 6 |
| IV-B | Hierarchical FSM behavioral planner | 6 |
| IV-B1 | Level one: type of turn | 6 |
| IV-B2 | Level two: high or low priority | 6 |
| IV-B3 | Level three: Stop or Go | 6 |
| V | Control | 7 |
| V-1 | Trajectory Generation and Tracking | 7 |
| V-A | Reference generator | 7 |
| V-A1 | Path Smoother Spline | 7 |
| V-A2 | Velocity Profiler | 7 |
| V-B | Model Predictive Control | 7 |
| VI | Validation results at multiple intersections | 8 |
| VI-A | Simulation assumptions & choices | 8 |
| VI-B | Relevant combinations of test-cases | 9 |
| VI-C | Traffic light intersection | 9 |
| VI-D | Fourway intersection without priority pre-assigned (NPA) | 10 |
| VII | Conclusion | 12 |
| Appendix A | | 15 |
| A-A | Intersection use-cases | 15 |
| A-B | Context information categorization | 15 |
| A-C | Hierarchical FSM | 16 |
| A-C1 | Hierarchical FSM level 2 | 16 |
| A-C2 | Hierarchical FSM level 3 left turn | 16 |
| A-C3 | Hierarchical FSM level 3 straight | 16 |
| A-C4 | Hierarchical FSM level 3 right turn | 16 |
| A-D | Closest reference point calculation | 17 |
| A-E | Kinematic bicycle model | 17 |
| A-F | Successive linearization | 18 |
| Appendix B | | 18 |
| B-A | MPC parameter tuning results | 18 |
| B-A1 | Prediction horizon tuning | 18 |
| B-A2 | State cost matrix tuning | 19 |
| B-A3 | Input cost matrix tuning | 20 |
| B-B | Additional simulation results | 21 |

Context-driven motion planning at non-equipped intersections

1st E.P. Steenhof

Department of Mechanical Engineering (CST)

Eindhoven University of Technology

Eindhoven, The Netherlands

e.p.steenhof@student.tue.nl

Abstract—In recent years, several Intersection Management (IM) methods have been proposed to regulate autonomous vehicles at traffic intersections. Centralized IM methods are able to improve the throughput and regulate autonomous vehicles at intersections. However, these methods rely on two-way vehicle-to-everything (V2X, X2V) communication, which in many situations, including in urban scenarios where human-driven and autonomous vehicles coexist, are not always economically or technically feasible. Therefore, in this paper a decentralized motion planning method is presented for intersections, which does not require V2X communication and solely relies on the ego-vehicle on-board sensors. As a result this method provides a safe crossing at non-equipped intersections (i.e., no intersection-to-vehicle communication). The decentralized motion planning method integrates a high-level hierarchical finite-state machine (FSM) with Model Predictive Control (MPC) techniques. A hierarchical FSM is used as the behavioral layer of the algorithm, which is based on traffic rules and uses boolean logic to incorporate context information. Additionally, a MPC-based controller is used to track the reference trajectory of the ego-vehicle and to account for the kinematic constraints. A numerical case study involving five intersection types is provided to validate the proposed method. As we will demonstrate, the proposed solution is able to regulate the ego-vehicle at five different intersection types without the requirement of V2X communication, whilst also respecting the kinematic behavior of the ego-vehicle.

Index Terms—Decentralize motion planning, model predictive control, finite-state machine, context information categorization.

I. INTRODUCTION

The first motorized vehicle dates back to 1885, when Karl Benz first introduced the internal combustion car to the world [1]. Since then, the car has made enormous technical advances in terms of safety and reliability, by either using passive or active safety improvements in vehicles. Passive safety improvements include airbags, seat belts and high energy crumple zones, in order to reduce injury on the passengers in case of an accident. Active safety systems focus predominantly on accident prevention and include Electronic Stability Program (EPS), Autonomous Emergency Braking System (AEBS) and Lane Keep Assist (LKA) [2].

Despite these advancements, the number of traffic deaths and injuries are increasing in the Netherlands [3]. A comprehensive study on road safety [4] concluded that, human errors were the sole cause in 94% of all accidents. In contrast, only 2% of all accidents were due solely to mechanical failures

(e.g., blowout tire) and 2% were only caused by environmental factors (e.g., weather conditions) [4]. Other studies [5–7] show that human errors play a crucial role in traffic congestion and accidents. Furthermore, recent studies indicate that driver errors contribute to up to 75% of all roadway crashes [8].

In particular traffic accidents that are related to intersections occur all too often. Statistic studies from several European countries show that 43% of all road injury accidents are related to intersections [9]. In the USA, about 96% of all intersection related accidents are caused by human drivers [10]. Human negligence is the major cause of these traffic accidents. As such there is a need for reducing human-caused traffic accidents (or even eliminating them altogether). A promising solution might be by moving the responsibility of driving from the human driver to the vehicle [11].

One technique that fits the later solution directly is Intersection Management (IM), which takes away the involvement of the human drivers by increasing the control effort from current intersection infrastructure designs (e.g., traffic light intersections). A literature study [12] shows that 93% of all proposed intersection management methods rely on fully automated and fully connected vehicles. Relatively little intersection management methods assume the coexistence of both Autonomous Vehicles (AV) and Human driven Vehicles (HV) or non-connected vehicles.

Existing approaches for IM methods can, firstly be categorized into two different approaches, *active* and *passive*. An active approach (e.g., [13]) relies on vehicle-to-everything communication (V2X, X2V) while a passive approach (e.g., [14]) does not. Secondly, IM methods can either be *centralized* or *decentralized*. Centralized methods (e.g., [15–17]), rely on a centralized control unit at each intersection. Centralized methods, however, could require a lot of time and money to realize, because they require alterations to intersections [18]. An easier way is to apply decentralized methods (e.g., [19, 20]), because they use existing intersection layouts and alter them as little as possible, in order to increase integration and to reduce cost. In this paper we will be interested in passive decentralized solutions given these benefits. Finally, note that existing approaches for IM methods can be categorized into three different categories, *optimization-based*, *rule-based* and *hybrid*.

Optimization-based methods (e.g., [16, 21, 22]) have been

developed to handle autonomous vehicles at intersections for mixed-traffic (i.e., both HV and AV) scenarios. The optimization-based methods can easily handle multiple goals and complex conditions by changing objective functions, constraints, and searching algorithms. However, optimization-based methods may lead to prohibitively high computational burden and may not always provide a global optimal solution (e.g., minimal travel time for all the vehicles) in the time interval required for intersection management [23]. Furthermore, according to [23], the computational burden of optimization-based methods significantly increases with increasing traffic volumes and the complexity of the situation (e.g., more road-users). Therefore, it might be difficult to implement optimization-based methods in real-time without the use of expensive massive cloud computing services.

Rule-based methods (e.g., [13, 15]) have successfully been developed to improve safety at intersection for mixed traffic scenario. Rule-based methods are easier to comprehend and to design, compared to optimization-based methods [12]. Furthermore, the computational burden of ruled-based methods are lower compared to optimization-based methods [12]. However, the design complexity of the rule-based method significantly increases when considering multiple goals (e.g., improve safety while also increase traffic throughput) and constraints in the model [12] due to, for example, rule explosion. The design of such systems and guarantees on their correct functionality could become complicated then.

Hybrid methods combine both rule-based and optimization-based methods and have been implemented for intersection control problems (e.g., [19]). Since hybrid methods are partially based on rules, their computational burden is less compared to optimization-based methods, making them more amendable for real-time implementation. The optimization part of hybrid methods improves their adaptivity to changing circumstances compared to rule-based methods. Nevertheless, a different combination of rule-based and optimization-based methods may lead to a significantly different performance [12] and as such effective design methods are needed.

A. Contributions and proposed approach

The research in this paper aims to advance the state of the art by introducing a novel method, which is as generic as possible and is able to handle the most common intersection types:

- T-intersection
- Roundabout
- Fourway no priority pre-assigned (NPA fourway)
- Fourway priority pre-assigned (PA fourway)
- Traffic light intersection (TL)

These intersection types also function as the use-cases for this research and are graphically shown in Appendix A-A. The goal in this paper is to design a motion planning method, which is fully decentralized and only relies on data gathered by the ego-vehicle sensors at the intersection. Note that gathering data using wireless communication (i.e., X2V) is deliberately excluded, since in a mixed (human driven and autonomous

vehicles) driving scenario not all vehicles are expected to be fully connected. This paper presents three main objectives in the context of motion planning for autonomous vehicles at intersections:

- A decentralized motion planning algorithm that only requires data gathered by the on-board ego-vehicle sensors.
- A context information categorization approach, which converts sensor data into logic boolean expressions.
- Integration of a hierarchical finite-state machine with model predictive control techniques.

In order to achieve these objectives, a ruled-based approach is used, because a rule-based approach is easier to design and to comprehend. Furthermore, the low computational burden makes it attractive for real-time implementation. The algorithm proposed in this paper is inspired by the algorithm developed in [24], where a high level ruled-based approach is integrated with a vehicle controller. The remainder of the paper is organized as follows. Section II provides the problem formulation with Section II-A explaining the terminology of the automated vehicle architecture. Section III provides a reflection on behavioral planners. In Section IV the proposed solution is explained and Section V explains the integration of MPC techniques within the proposed solution. Section VI provides the simulation results in order to validate the proposed solution. Section VII is dedicated to concluding remarks on the proposed solution.

II. PROBLEM FORMULATION

The goal of the motion planning module is to calculate appropriate actuation setpoints (e.g., steering angle and acceleration) for an autonomous vehicle, which uses a set of waypoints and is able to adapt to dynamically changing context information at intersections. The path of the ego-vehicle through the intersection is assumed to be given by a set of waypoints, which are a set of coordinate points in a global space. Mathematically this can be represented as,

$$W \triangleq \{(x^g(o), y^g(o)) \in \mathbb{R}^2 \mid o \in \mathbb{N}\}, \quad (1)$$

where $x^g : \mathbb{N} \rightarrow \mathbb{R}$ and $y^g : \mathbb{N} \rightarrow \mathbb{R}$ refers to the o^{th} longitudinal position and the lateral position respectively, in the global space. These waypoints are the input for the context-based motion planning problem as show in Fig. 1.

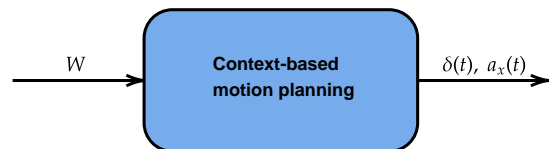


Fig. 1. Context-based motion planning input and output overview.

To enable context-based motion planning and control of the vehicle, the actuator inputs need to be calculated. This requires finding the required $\delta(t)$ steering and $a_x(t)$ acceleration setpoints that adapt to changing context information (e.g., traffic

light color change) at intersections. These actuator setpoints are the desired output of the context-based motion planning problem as shown in Fig. 1. An *automated vehicle architecture* needs to be designed in order to generate these outputs in a structured way. The terminology of such an architecture is explained in the next section.

A. Automated vehicle architecture

The core competencies of an automated vehicle architecture can be subdivided into three main categories [25], namely *perception*, *planning*, and *control*, as depicted in Fig. 2. In this paper the core competencies of an automated vehicle architecture are integrated in order to validate the proposed context-based motion planning solution at multiple intersections. Each competency is explained in more detail below. Furthermore, assumptions on the out of scope competencies are defined.

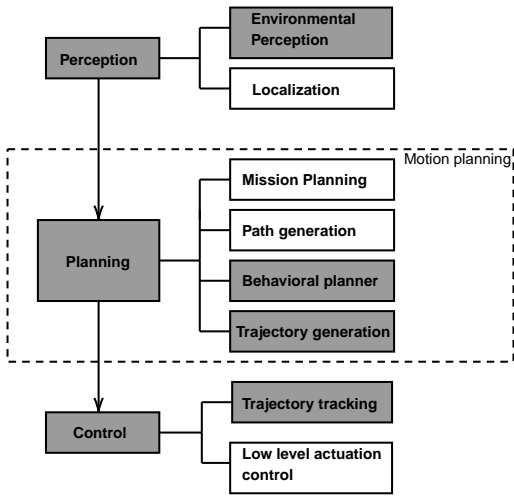


Fig. 2. Automated vehicle architecture overview, showing core and sub-core competencies and highlighting in grey the competencies related to this paper.

Perception refers to the ability of an autonomous system to collect information from the environment and extract relevant data. *Environmental perception* refers to developing a contextual understanding of the environment, such as the location of other road-users (with respect to the ego-vehicle), detection of road markings, traffic signs, and categorizing data by their relevance. *Localization* refers to the ability of the ego-vehicle to determine its position with respect to the environment (e.g., GPS coordinates). In this paper a fixed world coordinate frame is used for the localization of the ego-vehicle. The exact location of other road-users with respect to this fixed world coordinate frame is unknown.

Planning refers to the process of making purposeful decisions in order to achieve the ego-vehicle’s higher order goal, such as driving from city A to city B. The *mission planner* (i.e., the route planner) considers high level goals, such as the end destination and which roads should be taken to achieve this end destination. The *path generator* generates appropriate paths and sets of actions to achieve local objectives (e.g., turn right at a specific intersection), with the most typical objective

to reach the end goal destination. The mission planner and path generator block are outside of the scope of this paper and will be assumed available. It is assumed that the AV knows if it wants to turn left, right or go straight through at a specific intersection, given by a set of waypoints.

The *behavioral planner* (i.e., the decision maker) makes decisions to properly interact with other road-users and follow traffic rules, and thereby generates actions, such as change lanes, overtake, proceed through or stop at the intersection etc. The *trajectory generator* additionally adds the velocity information to the path generated by the path generator based on the action (e.g., stop or go at an intersection) decided by the behavioral planner [25].

The control competency, refers to the ego-vehicle’s ability to closely follow the reference trajectory (i.e., trajectory tracking). A control algorithm is used to select appropriate actuator input set points to carry out the planned motion and correct tracking errors [26]. *Low level actuation control* uses these actuator set points as reference set points for actuator control (e.g., electric motors).

The motion planning module within an automated vehicle architecture, as shown in Fig. 2, consists of four sub-competencies and is a broad research topic. The motion planner is responsible for calculating a safe, comfortable, and kinematically feasible trajectory from the current position of the ego-vehicle to the end goal position [26]. Depending on the application, the goal position may differ, such as the center of the stop line at the next intersection, or the next desired parking spot. The motion planner could use information about static and dynamic obstacles around the vehicle and generates a collision-free trajectory that satisfies dynamic and kinematic constraints on the motion of the vehicle. However, the main objective of the motion planner in this paper is to regulate the ego-vehicle at five different intersections without the requirement of V2X communication, to correctly adapt to changing context information (e.g., changing traffic light color) and to correctly interact with other road-users (e.g., give way according to traffic rules). So in order to lower the computational burden of the created solution the motion planner in this paper does not take active obstacle avoidance (i.e., driver around an obstacle) into account.

III. STATE OF THE ART

A. Behavioral planners reflection

An autonomous vehicle system requires a behavioral planner in order to function, as shown in Fig. 2. In this Section four different behavioral planners (Finite State-Machine, Reinforcement Learning, Partially Observable Markov Decision Process, Inverse Reinforcement Learning), are compared to each other and are evaluated based on the following four *Key Performance Indicators* (KPI’s): *Complexity*: The ease for a designer to understand, grasp and apply the method. *Maintainability*: The ease to adapt the method, e.g., when requirements are revised or added. *Scalability*: The ease to incorporate more road-users and various driving scenarios. A method with better scalability requires less amount of work to be revise when

the driving scenarios become more crowded or more complex due to, for example, different road layouts, traffic signs or incorporating different types of road-users (e.g., pedestrians and cyclists). *Computation/Data Burden*: The computational burden and the amount of data required in order to make the method function properly. The behavioral planners are evaluated using the KPI's and from a high-level point of view (i.e., not regarding specific urban traffic use-cases).

1) *Finite State-Machine*: A finite state-machine (FSM) is a model that describes the behavior of a system in each state. Based on the current state and a given input, the FSM performs state transitions and produces outputs. There are basic types like Mealy and Moore machines and more complex types like Harel and Unified Modeling Language (UML) state charts [27–29].

According to [30] the FSM has the following four advantages. Firstly, the rules that have to be checked are limited to the different states and the transition conditions. Rules that do not apply to the current state can be left out. This limits the conditions that have to be checked at every iteration. Secondly, the (traffic) rules can be applied directly to different state transitions and scenarios. Thirdly, the implementation of FSM as a behavior planner is simpler than other behaviour planner methods. FSM visually shows how the states change due to dynamic changes in the system, influenced by the environment and other road users. This also enables to quickly find faulty or unexpected behaviour of the system. Fourthly, the data cost and the computational burden are low due to the discrete nature.

However, FSM has the following three disadvantages [30]: Firstly, rule-explosion (i.e., low maintainability) could occur when complex use cases need to be handled, leading to a rapid increase in the number of rules/states (i.e., rule explosion). Secondly, dealing with a noisy environment could be difficult for a FSM, for example, if a low signal to noise value makes it impossible to validate a transition condition (e.g., yield sign detection). Leading to undesired behaviour of the autonomous vehicle. Thirdly, the handling of newly encountered traffic situations. Due to the discrete nature of this approach, the pre-programmed logic of the system could react in an incorrect way, i.e., guaranteeing correct behavior could be difficult.

2) *Reinforcement Learning*: Reinforcement Learning (RL) is a form of machine learning in which an agent learns how to interact with a given environment by taking actions and receiving a continuous reward. This reward works in a similar way to cost functions in optimal control algorithms. The agent could start by interacting with a simulated environment. At the start the agent will not be good at the particular task. However, over time as the agent tries to maximize its reward, the agent might eventually learn how to master the task (more details on this method can be found in [31]).

There are three advantages [30, 32], for reinforcement learning. Firstly, there is less rule explosion compared to the FSM method. Secondly, the scalability of the algorithm is better. The algorithm could learn new scenarios directly from newly created simulation environments. Thirdly, the algorithm

can safely filter out failure cases (e.g., AV not stopping for a red light) in a simulation environment.

While reinforcement learning is a very interesting and highly promising area of research, there are two major disadvantages according to [30]. Firstly, many simulation environments used to learn the policies required for autonomous driving are often too simplified. Due to this simplicity, the policies learned may not be transferable to real world traffic environments. Severe computational requirements occur when more realistic simulators are used. Secondly, there is an issue concerning safety. It is, according to [30], “difficult to perform rigorous safety assessment of a learned system, as they are mostly black boxes in terms of the way in which decisions are made.”

3) *Partially Observable Markov Decision Process*: (POMDP) uses a special reinforced learning model with rewards, which has the main advantage, according to [32], that it is able to make decisions in case there is uncertainty in the system and there are unobservable states (more details about how the POMDP model works can be found in [33]). However, according to [32], POMDP models lead to a computationally high burden behaviour planner, due to the added mathematical complexity of POMDP models compared to the traditional RL models.

4) *Inverse Reinforcement Learning*: In Inverse Reinforcement Learning (IRL) rather than trying to obtain a policy given a reward function, the approach is to use data gathered from human actions to create the maximum reward policy instead of learning such a policy on its own. Once the reward policy is learned by the IRL model, the behavioral planner could execute driving decisions similarly to a human driver.

IRL methods, according to [30], have two main advantages. Firstly, the scalability of the algorithm is better than the FSM method and the RL model. This is because IRL can use newly gathered data, from other road users (e.g., human drivers), to learn how to handle newly encountered scenarios. Meaning that IRL could keep on improving when more data is available for the maximum reward policy. Secondly, the maintainability is better compared to the FSM method, since it does not necessarily require a predefined set of rules.

IRL is promising as a behaviour planner, however it has the following three disadvantages [34]. Firstly, data acquisition: IRL requires massive data sets to train the algorithm, these data sets should be unbiased and of good quality. Data acquisition might also be difficult due to privacy laws and regulations.

Secondly, the computational burden: IRL requires sufficient time to learn and develop a good maximum reward policy with a considerable amount of accuracy and relevancy. Depending on the application, IRL also needs high computational power to process the data quickly enough for real-time traffic applications.

Thirdly, high error-susceptibility: IRL is an autonomous system but highly susceptible to errors. For example, in case an IRL algorithm is trained with data sets small enough to be noninclusive. This could lead to biased predictions coming from a biased data set. Leading to a chain of errors that could

remain undetected for long periods of time, because of the automated nature and complexity of the algorithm.

In Table I an overview of the discussed behaviour planners is shown, against the four KPI criteria. In this thesis the FSM method is used as the behaviour planner for the proposed solution, because the low computation burden and complexity makes it the most attractive method to handle complex urban scenarios within reasonable computation times. Furthermore, the low data requirement of the FSM method makes it an attractive choice for a behavior planner where no V2X communication is available. The design of the FSM behavioral planner is explained in detail in Section IV-B.

TABLE I
BEHAVIOUR PLANNERS HIGH-LEVEL KPI OVERVIEW

| | FSM | RL | POMDP | IRL |
|-------------------------|-----|-----|-------|-----|
| Computation/Data Burden | ++ | - | - | -- |
| Scalability | - | +/- | + | ++ |
| Complexity | ++ | +/- | - | - |
| Maintainability | - | + | + | ++ |

IV. DECENTRALIZED CONTEXT-BASED MPC MOTION PLANNER

In this thesis a decentralized (i.e., implemented inside the ego-vehicle) context-based MPC motion planning algorithm is designed, which combines the three core competencies (as described in Section II-A) of an automated vehicle architecture (shown in Fig. 2). The block diagram of the proposed decentralized context-based MPC motion planner algorithm is shown in Fig. 3.

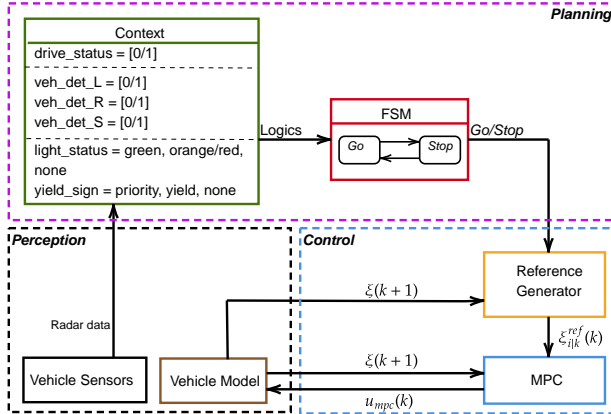


Fig. 3. The block-diagram of the decentralized Context-based model-predictive control motion planning algorithm.

The function of each block is explained in more detail in the next sections.

A. Perception block

The perception block consists of two sub-blocks, namely the vehicle model and vehicle sensors. The vehicle model block is a kinematic bicycle model and is used for the localization

part of the algorithm (Fig. 2), using world coordinates. The vehicle sensors handle the environmental perception part of the algorithm (Fig. 2). In order to connect the perception and the planning competencies (black dotted and purple dotted blocks in Fig. 3), *context information categorization* is required, as explained below in more detail.

1) *Environmental perception*: The vehicle sensors provide crucial information on the driving environment and provide the environment perception. Environmental perception consists of two critical elements: Firstly, detect and recognize traffic light color and traffic signs, which is assumed to be available, i.e., a processing algorithm using the camera on the ego-vehicle is able to give this information. Secondly, road-users detection, which is handled by short-range radars on the ego-vehicle.

2) *Context information categorization*: Logic boolean expressions (green box in Fig. 3) are used as the context categorization, since the FSM requires logic transition statements. The radars in the vehicle split the front view into three detection zones: *Left*, *Straight* and *Right* (as shown in Fig. 17 in Appendix A-B), each with their own logic boolean conditions. 0 means no road-user is detected, 1 means a road-user is detected in this radar zone. Furthermore, traffic light status and yield sign detection are also implemented using a logic boolean expression. All these logic booleans are used in the *Hierarchical FSM*, which is described in Section IV-B.

B. Hierarchical FSM behavioral planner

Using hierarchy in the FSM (thus obtaining a hierarchical FSM) improves, according to [30], the maintainability and the scalability of the FSM behavior planner we will create. A three level hierarchical FSM is created as the behavior layer of the proposed algorithm (FSM box in Fig. 3).

In this Section the three level hierarchical FSM is explained, from top to bottom. In Fig. 4 the blue blocks are level one, the red blocks are level two and the black blocks are level three of the hierarchical FSM.

1) *Level one: type of turn*: The first level of the hierarchical FSM consists of three different states (*left turn*, *straight*, *right turn*), which correspond to the three types of manoeuvres (i.e., left turn, straight and right turn) the ego-vehicle can make at an intersection. Each of these manoeuvres has its own logic boolean expression and functions as the state transition conditions as shown in Fig. 4.

2) *Level two: high or low priority*: The second level of the hierarchical FSM consists of two different states being high priority and low priority road. The transition between these two states depends on context (environment perception) logic boolean conditions, e.g., if a yield sign or a traffic light is detected, as shown in Fig. 18. Since these are the boolean conditions of the transition of level two, both these states are inside the parent states *Straight*, *Left Turn* and *Right Turn* of level one (as shown in Fig. 4).

3) *Level three: Stop or Go*: The third level consists of two states *Go* and *Stop*, which are both inside the parent states *HighPriorityRoad* and *LowPriorityRoad* of level two. Now the context logic boolean conditions (green box in Fig. 3) of the

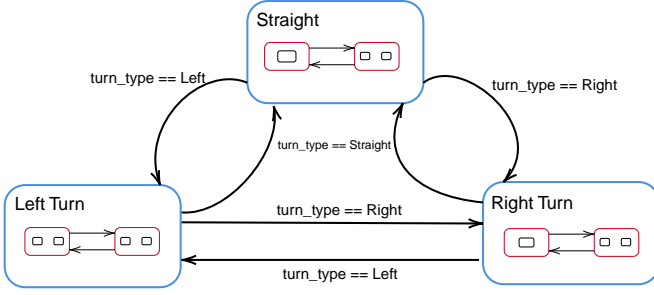


Fig. 4. The blue blocks are the first level, the red blocks are the second level and the black blocks are the third level of the hierarchical FSM.

radar detection zone are used. The transition statements are designed such that the ego-vehicle is obeying the traffic rules. For example, when the ego-vehicle is on a high priority road and wants to turn left, in case it detects a vehicle in the front radar it should stop to give priority to the upcoming traffic (Fig. 19), as mandated by the Dutch traffic rules (see [35] to find all the Dutch traffic rules). This leads to different boolean expressions on the transition between the Go and Stop state depending on the previous higher level states (turn type and high or low priority), as shown in Fig. 19 until 24 in Appendix A-C.

V. CONTROL

Based on the two state decision Go or Stop made by the hierarchical FSM a reference trajectory will be generated by the *reference generator* (yellow box in Fig. 3) and is explained in Section V-A. In order to closely follow the reference trajectory from the reference generator, a control algorithm (Control layer in Fig. 3) is required to calculate appropriate actuator inputs (i.e., steering angle and acceleration of the ego-vehicle). Model Predictive Control (MPC) has seen an uprise in automotive applications [36] and has, according to [25], some attractive advantages compared to a PID controller. Firstly, MPC allows for an easier design of a multi variable feedback controller. Secondly, MPC respects actuator limitations by setting constrains. Thirdly, MPC uses an objective function to optimize the control input effort. The MPC algorithm is explained in Section V-B.

1) *Trajectory Generation and Tracking*: There are two general approaches to trajectory generation with known path information. The first approach uses the optimization method to both generate a trajectory and to track it simultaneously. This method integrates both the generation and tracking tasks into one optimization problem. This approach is often applied for optimal time application (e.g., [37]). Running the optimization problem in real time is a challenge due to limited processing power, and may lead to high computation times for planning in a complex environment [25] (e.g., urban scenarios). The second approach is to decouple trajectory generation and tracking. This is the approach used in this

thesis, since this approach reduces the computational burden of the Model Predictive Control algorithm.

A. Reference generator

The Reference generator (yellow block in Fig. 3) consists of two main parts. Firstly, the *Path Smoother Spline* part generates a smooth continuous vehicle path. Secondly, the *Velocity Profiler* generates a velocity profile of the driving path that satisfies a set of specified kinematic constraints.

1) *Path Smoother Spline*: The Path Smoother Spline block generates a smooth continuous vehicle path, consisting of a sequence of discretized (sampled) waypoints, by fitting the input reference path waypoints to a second order cubic spline. The path-smoothing algorithm interpolates a parametric cubic spline that passes through all input discretized reference waypoints points, converting a C^1 -continuous path to a C^2 -continuous path ([38, 39] for more details).

2) *Velocity Profiler*: Choosing the correct velocity profile can, according to [40], improve smoothness, reduce wear, and lower transfer times for a broad range of motion control applications. Trapezoidal profiles tend to increase wear [41], because of several discontinuities between the acceleration regions, which could lead to unsought vibrational effects. S-curve profiles solve this problem, but are more complex mathematically and thus increase the computation time of the algorithm. The Velocity Profiler generates a s-curve velocity profile of a driving path that satisfies the following set of specified kinematic constraints:

- The maximum allowable speed of the vehicle.
- The maximum longitudinal acceleration and deceleration of the vehicle.
- The maximum longitudinal jerk of the vehicle.

The generated velocity profile is a seven-interval curve. At each time interval within the curve, the jerk, acceleration, and velocity of the vehicle change to satisfy the specified constraints (see [42] for more details).

B. Model Predictive Control

The general form of the MPC problem for an autonomous vehicle, solved at discrete steps, used in this paper is formulated as,

$$\min_{V_k} J(s(k), V_k), \quad (2a)$$

$$s.t. \quad s_{i+1|k} = f(s_{i|k}, v_{i|k}), \quad \forall \quad i = 1, \dots, N_p, \quad (2b)$$

$$s_{0|k} = s(k), \quad (2c)$$

$$V_k = (v_{0|k}, \dots, v_{N_p-1|k}), \quad (2d)$$

$$s_{i|k} \in \mathcal{C}_{i|k}, \quad \forall \quad i = 1, \dots, N_p, \quad (2e)$$

$$v_{i|k} \in \mathcal{V}_{i|k}, \quad \forall \quad i = 0, \dots, N_p - 1, \quad (2f)$$

where s and v represent the states and inputs of the system respectively, J is the cost term representing the trajectory following error, f represents the vehicle kinematics, $\mathcal{C}_{i|k}$ represents the state constrains, $\mathcal{V}_{i|k}$ represents the input constraints

and (2c) represents the initial state condition at any time step k . V_k is the vector of stacked inputs obtained over a prediction horizon N_p , which refers to the length of the look-ahead window for which future states and inputs are predicted. The general form MPC problem (2a) can be rewritten to a the reference trajectory tracking problem, which uses a quadratic cost function and is given by,

$$\min_{U_k} \sum_{i=0}^{N_p-1} \left((\xi_{i|k} - \xi_{i|k}^{ref})^T Q (\xi_{i|k} - \xi_{i|k}^{ref}) + (u_{i|k} - u_{i|k}^{ref})^T R (u_{i|k} - u_{i|k}^{ref}) + \Delta u_{i|k}^T E \Delta u_{i|k} \right) \quad (3a)$$

$$s.t. \quad \xi_{i+1|k} = m^{KB}(\xi_{i|k}, u_{i|k}), \quad \forall \quad i = 0, \dots, N_p - 1, \quad (3b)$$

$$\xi_{0|k} = \xi(k), \quad (3c)$$

$$U_k = (u_{0|k}, \dots, u_{N_p-1|k}), \quad (3d)$$

$$\xi_{min} \leq \xi_{i|k} \leq \xi_{max} \quad \forall \quad i = 1, \dots, N_p, \quad (3e)$$

$$u_{min} \leq u_{i|k} \leq u_{max} \quad \forall \quad i = 0, \dots, N_p - 1, \quad (3f)$$

$$\Delta u_{min} \leq \Delta u_{i|k} \leq \Delta u_{max} \quad \forall \quad i = 0, \dots, N_p - 1, \quad (3g)$$

where (3a) represents the quadratic cost function with a finite horizon, where state weight matrix $Q \succ 0$, input weight matrix $R \succeq 0$ and input rate weight matrix $E \succeq 0$ are (semi-)positive definite matrices. $\xi_{i|k}^{ref}$ represents the prediction reference states. (3b) represents the non-linear kinematic vehicle model (as explained in Appendix A-E). (3c) represents the initial state conditions at any time step k , (3d) represents the vector of stacked inputs obtained over the prediction horizon, (3e) represents the vehicle state constrains, (3f) represents the vehicle input constrains, (3g) represents the vehicle input rate constrains.

The kinematic bicycle (plant) model (m^{KB}) is a non-linear system (for more details see Appendix A-E). Although some non-linear MPC (NMPC) techniques are proposed in literature [43, 44], it should be noticed that the computational effort necessary in non-linear techniques is much higher than in linear MPC (LMPC). In NMPC a nonlinear programming problem is solved online, which could be non-convex, and also could have a larger number of decision variables. Furthermore the global minimum, according to [45], is in general difficult to find. Thus a linear technique is proposed to overcome the problem related to the computational burden of NMPC. The fundamental idea of this technique consists in using a successive linearization approach, as outlined in [46], yielding a linear, time-varying description of the system. According to [46] it is possible to transform the non-linear optimization problem to be solved at each sampling time (approximately) in a Quadratic Programming (QP) problem. These type of problems are convex and can be solved rapidly by numerically robust optimization algorithms.

The reference generator (Section V-A) provides reference conditions for the ego-vehicle, over the prediction horizon. The prediction reference states ($\xi_{i|k}^{ref}$) are calculated by filtering the reference spline points, which have been generated by the path smooth spline and the velocity profiler (inside the Reference Generator). This filtering requires the closest reference spline point to the current ego-vehicle states. The closest reference spline points calculation is explained in Appendix A-D. To adapt to these changing operating conditions, MPC supports updating the prediction model and its associated nominal conditions at each optimization step. This can be useful when, for example, in the corners the heading angle ϕ of the ego-vehicle can change from 0 to 90 degrees within the prediction horizon. This leads to the plant model and the nominal conditions to vary over the prediction horizon, so a time-varying MPC algorithm will be used. Such a linear time-varying (LTV) model is useful when controlling periodic systems or nonlinear systems that are linearized around a time-varying nominal reference trajectory provided by the reference generator. The MPC algorithm used in this paper thus requires a LTV prediction model and is explained in detail in Appendix A-F.

VI. VALIDATION RESULTS AT MULTIPLE INTERSECTIONS

In this Section, the proposed decentralized context-based MPC motion planning algorithm is validated using the five use-cases (i.e., intersection types) as described in Section I-A. Firstly, the assumptions and choices regarding the simulations are explained. Secondly, the results of relevant combinations of test-cases is shown using tables. Thirdly, the results of a left turn at a traffic light intersection are explained to show the proper behavior of the algorithm. Fourthly, the results of a straight drive at a NPA fourway are explained, to show the conservative behavior (i.e., ego-vehicle waits unnecessarily long) of the algorithm. The context-based MPC motion planning algorithm is implemented as a constrained MPC problem in MATLAB using the adaptive MPC toolbox with design parameters provided in Table II. The MPC parameters (Q, R, N_p) in Table II have been determined by sensitivity analysis, the results of this analysis is shown in Appendix B-A. The hierarchical FSM is designed using the state-flow Simulink toolbox.

A. Simulation assumptions & choices

In this Section the simulation assumptions and choices are given for perception, planning and control competencies (dotted blocks in Fig. 3). For the perception module the following assumptions and choices are defined: Firstly, the intention of other road users are unknown. Secondly, no target classification is taken into account (i.e., critical and non-critical targets). Other road-users are only detected or not detected. Thirdly, the context logic only can change every simulation time step ($T_s = 0.1$ s). Fourthly, the sensors have a fixed detection range. Finally, sensor failure and fault detections are not taken into account.

TABLE II
DESIGN PARAMETERS - SIMULATION SETTINGS.

| Ego-vehicle | | |
|--------------------------------|-----------------------------|---------------------|
| Mass | 1280 | [kg] |
| Distance COG to rear axle | 1.08 | [m] |
| Distance front and rear axle | 2.60 | [m] |
| Actuator & comfort constraints | | |
| Steering angle * | $37/180\pi$ | [rad] |
| Longitudinal acceleration * | $0.2g$ | [m/s ²] |
| Longitudinal deceleration * | $-0.3g$ | [m/s ²] |
| Longitudinal Jerk * | $0.25g$ | [m/s ³] |
| Steering angle rate * | $500/180\pi$ | [rad/s] |
| Max longitudinal velocity # | 13.88 | [m/s] |
| MPC settings | | |
| State cost matrix Q | $diag(0.2 \ 0.2 \ 0 \ 0.8)$ | |
| Input cost matrix R | $diag(0.05 \ 0)$ | |
| Input rate cost matrix E | $diag(0.001 \ 0.001)$ | |
| Prediction horizon N_p | 10 | |
| Sample time T_s | 0.1 [s] | |

* Input constraint, # State constraint

For the planning module the following assumptions and choices are defined: Firstly, the other road-users obey the current set of dutch traffic laws [35]. Secondly, there is no V2X communication. Thirdly, the red and orange traffic light is treated equally, both lead to a Stop request.

For the control module the following assumptions and choices are defined: Firstly, whenever a Stop request is received, the ego-vehicle will brake to standstill in 20 [m] from its current position. Secondly, whenever a Go request is received, the ego-vehicle will accelerate to the desired reference velocity within 30 [m]. Thirdly, the reference waypoints (W in (1)) are in the middle of the road. Finally, the MPC state and input cost matrices do no change during the simulation.

B. Relevant combinations of test-cases

Different test-cases have been simulated to find the coverage of the algorithm regarding the five different use-cases (as described in Section I-A). There are already 90 possible test-cases that can be simulated, when assuming there is only one vehicle coming from each direction at a current time and not taking into account the possible trajectories (e.g., turn right or left at the intersection) the other road-users can follow. However, test-cases can be combined to only relevant test-cases. The test-cases of PA fourway and T-intersection use-cases are combined into one results table, since a T-intersection is a PA fourway with one road less. The T-intersection has less variety in possible test-cases compared to the PA fourway use-case. The roundabout test-cases are simulated as a right turn, since joining and exiting a roundabout requires a right turn locally by the ego-vehicle in right-hand traffic countries. A traffic light intersection can be seen as a fourway intersection where the priority transition changes with time instead of the ego-vehicle position at the intersection. A green light indicates a high priority road and a red or an orange light indicates a low priority road, as indicated by state transition conditions shown in Fig. 18. The results of all the relevant test-cases are shown in the Tables III, IV, V, VI for the NPA fourway, Traffic light, PA fourway/ T-intersection and roundabout use-cases respectively.

The results that are indicated with a '+' in the tables, mean that the algorithm works as intended (i.e., the Stop and Go request are correct). An example of correct behavior is explained in Section VI-C. The 'o' indication means the algorithm works, however it shows a conservative behavior (i.e., receives a Stop request when it should be allowed to receive a Go request). The 'o' indicator is further explained in Section VI-D.

TABLE III
TEST-CASE RESULTS FOR THE NPA FOURWAY INTERSECTION, '+' PASSED, 'O' CONSERVATIVE.

| NPA fourway | Vehicle detected in radar zone | | | | | | | |
|-------------|--------------------------------|---|---|---|-----|-----|-----|-------|
| | Turn EV | L | S | R | L&S | L&R | S&R | L&S&R |
| Right | + | + | + | + | + | + | + | + |
| Straight | + | + | o | + | + | o | o | o |
| Left | + | o | o | o | o | o | o | o |

TABLE IV
TEST-CASE RESULTS FOR THE TL INTERSECTION, '+' PASSED, 'O' CONSERVATIVE.

| Traffic Light | Priority | Vehicle detected in radar zone | | | | | | | |
|---------------|----------|--------------------------------|---|---|-----|-----|-----|-------|---|
| | | L | S | R | L&S | L&R | S&R | L&S&R | |
| Right | High | + | o | + | + | o | + | o | o |
| | Low | + | + | + | + | + | + | + | + |
| Straight | High | + | + | + | + | + | + | + | + |
| | Low | + | + | + | + | + | + | + | + |
| Left | High | + | + | + | + | + | + | + | + |
| | Low | + | + | + | + | + | + | + | + |

TABLE V
TEST-CASE RESULTS FOR THE PA FOURWAY INTERSECTION & T-INTERSECTION, '+' PASSED, 'O' CONSERVATIVE.

| PA fourway & T-int | Priority | Vehicle detected in radar zone | | | | | | | |
|--------------------|----------|--------------------------------|---|---|-----|-----|-----|-------|---|
| | | L | S | R | L&S | L&R | S&R | L&S&R | |
| Right | High | + | + | + | + | + | + | + | + |
| | Low | o | o | o | o | o | o | o | o |
| Straight | High | + | + | + | + | + | + | + | + |
| | Low | o | o | o | o | o | o | o | o |
| Left | High | + | o | + | + | + | + | + | o |
| | Low | o | o | o | o | o | o | o | o |

C. Traffic light intersection

In this test-case scenario, the ego-vehicle wants to make a left turn at a traffic light intersection. In Fig. 5 the context at the intersection is shown, when the ego-vehicle received a Stop request (at $t = 0$ s), since the traffic light for the ego-vehicle is red as shown in Fig. 7. The Stop request leads to a deceleration as shown in Fig. 8 and thus to a decrease in

TABLE VI

TEST-CASE RESULTS FOR THE ROUNDABOUT INTERSECTION, '+' PASSED, 'o' CONSERVATIVE.

| Roundabout | | Vehicle detected in radar | | | | | | | |
|------------|----------|---------------------------|---|---|-----|-----|-----|-------|--|
| Turn EV | Priority | L | S | R | L&S | L&R | S&R | L&S&R | |
| Right | High | + | + | + | + | + | + | + | |
| | Low | + | o | + | o | + | + | o | |

the longitudinal velocity as shown in Fig. 7. At $t = 9$ s the traffic light turns green (Fig. 7), however the ego-vehicle does not directly receive a Go request, because the front radar (as shown in Fig. 6) detects the purple vehicle. Since, according to the dutch traffic rules [35] turning traffic have to give way to straight through traffic. At $t = 10.5$ s the purple vehicle is outside of the front radar detection zone and the ego-vehicle receives a Go request and starts to accelerate (Fig. 8). This is the desired behavior of the motion planning and thus receives the '+' indicator in the result Table IV.

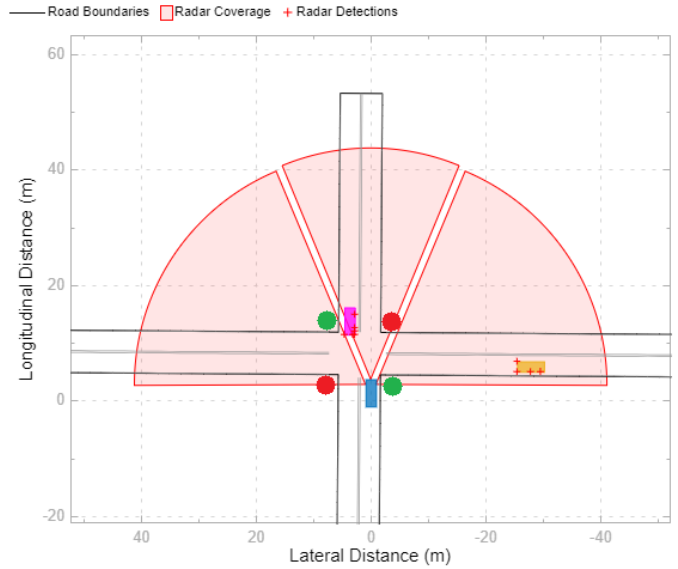


Fig. 6. The context information at the traffic light intersection when the light turns green for the ego-vehicle (blue).

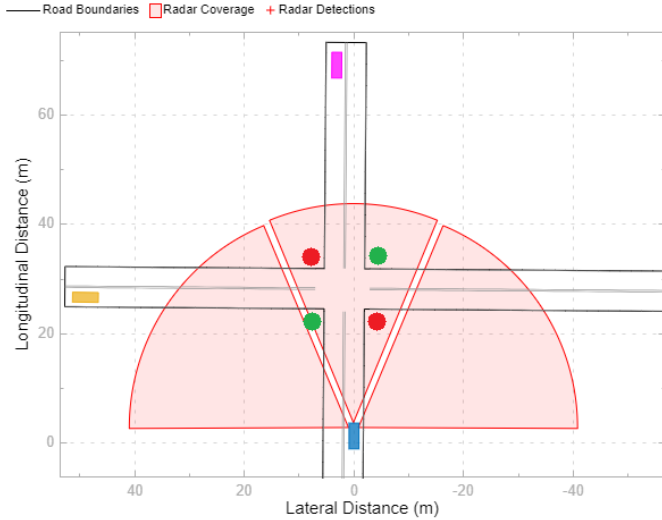


Fig. 5. The context information at the traffic light intersection when the ego-vehicle received a Stop request.

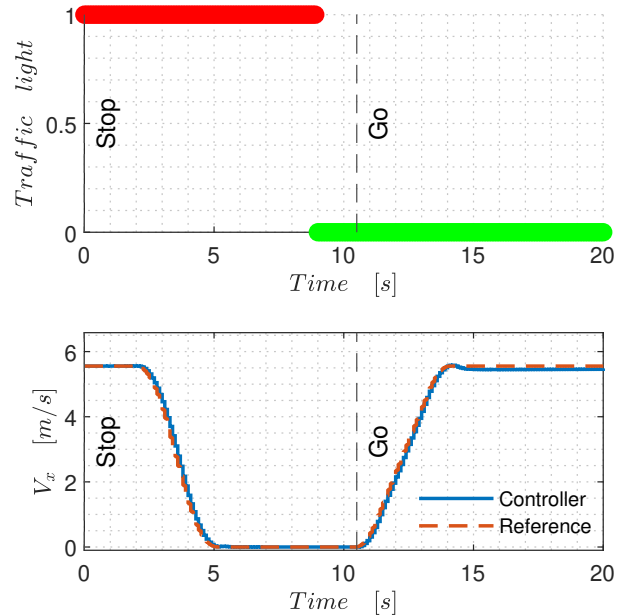


Fig. 7. The traffic light color status and longitudinal velocity of the ego-vehicle as a function of time.

D. Fourway intersection without priority pre-assigned (NPA)

In this scenario the ego-vehicle wants to cross a NPA fourway. Since no yield sign, priority sign or traffic light is detected, the initial priority of the ego-vehicle is set to low priority. In Fig. 9 the intersection context is shown when the ego-vehicle received a Stop request, because the purple vehicle was detected by the right radar. The ego-vehicle should indeed stop since, according to the traffic rules [35], vehicles coming from the right have priority at NPA intersections. The ego-vehicle starts to decelerate and comes to a standstill as shown in Appendix B-B Fig. 35 and Fig. 36. Fig. 10 shows a context situation where the ego-vehicle receives the Go request and is able to pass through the intersection. Since it has priority over the yellow vehicle detected in the left radar and no vehicle

is detected in the right radar (Fig. 10). The proposed motion planning algorithm shows conservative behavior, an example of this behavior is shown in Fig. 11. The ego-vehicle should be allowed to start accelerating, since both the purple and yellow vehicle move away from the ego-vehicle. However, with the current FSM design the ego-vehicle still receives a Stop request since it detects a vehicle in both the right and left radar. This is considered conservative behavior and thus is marked with a 'o'. In the current algorithm design no

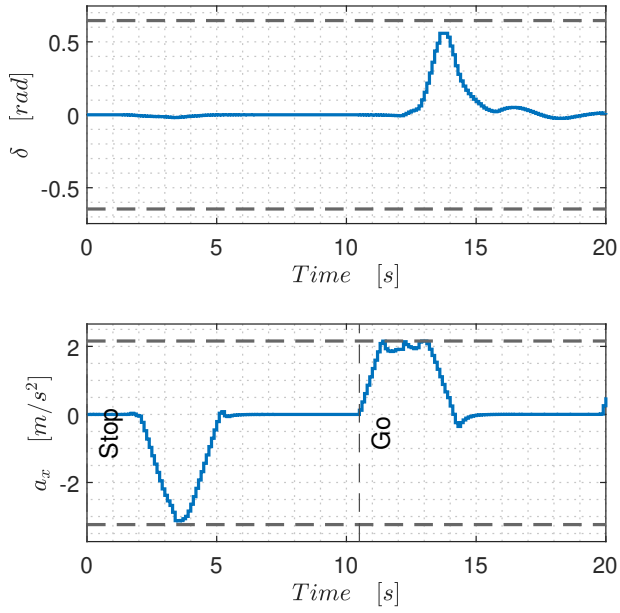


Fig. 8. The actuator input generated by the MPC controller, the horizontal dotted lines show the actuator limits.

information on the driving direction of the other vehicles at the intersection is taking into consideration, leading to this conservative behavior. A solution would be to improve the context information to incorporate lane direction information of all the road-users (e.g., drive-towards critical target or drive-away non-critical target).

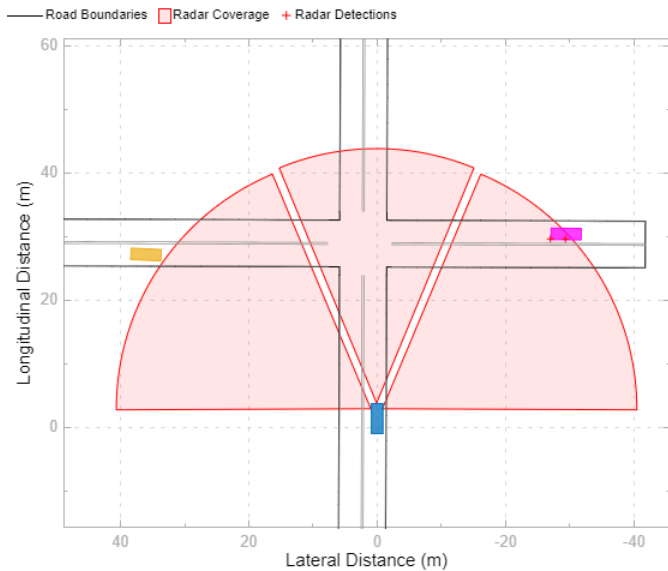


Fig. 9. The context information at a no priority pre-assigned fourway intersection when the ego-vehicle received a Stop request.

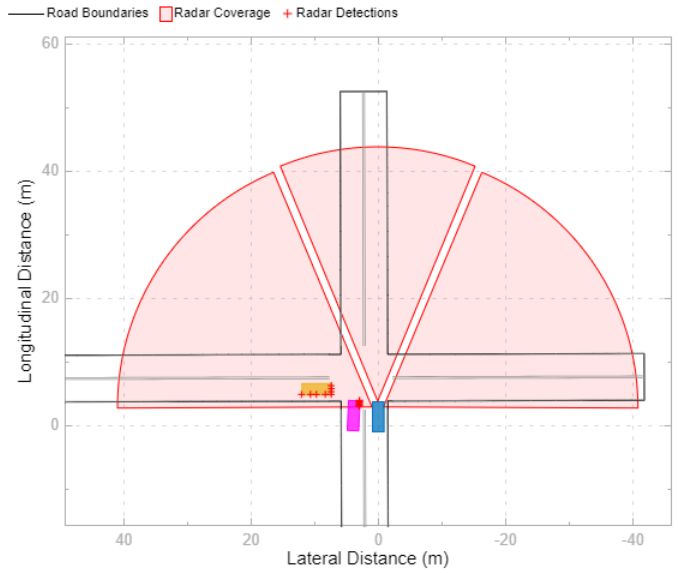


Fig. 10. The context information at a no priority pre-assigned fourway intersection when the ego-vehicle receives a Go request, since no vehicle is detected in the right radar.

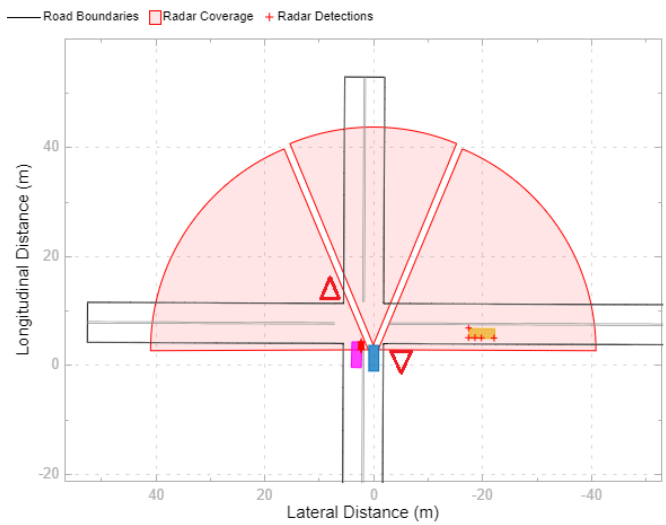


Fig. 11. The context information at a PA fourway intersection, yield signs indicate the low priority roads. The ego-vehicle (blue vehicle) wants to make a left turn and keeps receiving a Stop request (conservative behavior) since it detects the yellow vehicle.

VII. CONCLUSION

In this paper a context-based motion planning method for autonomous vehicles to safely cross intersections is presented, which is decentralized (i.e., inside the ego-vehicle) and does not require equipped (i.e., I2V) intersections or connected vehicles. Since it is not economically feasible to equip every traffic intersection, and furthermore, in mixed traffic scenarios not all vehicles are expected to be connected with each other. To solve this, the proposed method only relies on context information generated based on the on-board ego-vehicle sensors. The proposed method converts this context information into logic booleans, based on traffic rules. These logic booleans lead to state transitions in a hierarchical finite-state machine behavior planner, leading to either a Go or a Stop request for the ego-vehicle. A reference generator converts the desired path into a kinematically feasible trajectory for the ego-vehicle, based on the Go or Stop request from the hierarchical FSM. The MPC controller calculates desired inputs for the ego-vehicle in order to track the generated trajectory by the reference generator, while respecting the actuator limits of the ego-vehicle. Simulation results using five different intersection layouts demonstrate the effectiveness of the proposed method, without the need of V2X communication and without the requirement of a specific FSM design for each intersection layout, while also obeying traffic rules. The proposed method shows conservative behavior (i.e., the ego-vehicle waits unnecessarily long) when other road-users are driving away from the ego-vehicle, due to the current context information implementation. Future work should be aimed at reducing this conservative behavior by adding critical target and non critical target categorization and dynamical information of the other road-users with respect to the ego-vehicle. Furthermore, multi-lane intersections should be taken into consideration and also at least two autonomous vehicles using the proposed motion planning algorithm at the same moment in time and at the same intersection.

REFERENCES

- [1] Erik Eckermann. *World history of the automobile*. SAE, 2001.
- [2] Yves Page et al. “The evaluation of the safety benefits of combined passive and on-board active safety applications”. In: *Annals of Advances in Automotive Medicine/Annual Scientific Conference*. Vol. 53. Association for the Advancement of Automotive Medicine, 2009, p. 117.
- [3] CBS Doden en gewonden in het wegverkeer. <https://www.cbs.nl/nl-nl/maatschappij/verkeer-en-vervoer/transport-en-mobiliteit/mobiliteit/verkeersongevallen/categorie-verkeersongevallen/doden-en-gewonden-in-het-wegverkeer>. Accessed: 05-04-2020.
- [4] Santokh Singh. *Critical reasons for crashes investigated in the national motor vehicle crash causation survey*. Tech. rep, 2015.
- [5] S Weber et al. “Different types of distraction causing accidents”. In: *Presentation at the DDI* (2018).
- [6] Teimour Allahyari et al. “Cognitive failures, driving errors and driving accidents”. In: *International journal of occupational safety and ergonomics* 14.2 (2008), pp. 149–158.
- [7] Thomas A Dingus et al. “Driver crash risk factors and prevalence evaluation using naturalistic driving data”. In: *Proceedings of the National Academy of Sciences* 113.10 (2016), pp. 2636–2641.
- [8] Neville A Stanton and Paul M Salmon. “Human error taxonomies applied to driving: A generic driver error taxonomy and its implications for intelligent transport systems”. In: *Safety Science* 47.2 (2009), pp. 227–237.
- [9] A Molinero Martinez et al. “Accident causation and pre-accidental driving situations”. In: *Part 1* (2008), p. 176.
- [10] *Crash Factors in Intersection-Related Crashes: An On-Scene Perspective*. <https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/811366>. Accessed: 05-04-2020.
- [11] Caner Filiz. “Can Autonomous Vehicles Prevent Traffic Accidents?” In: *Accident Analysis and Prevention*. IntechOpen, 2020.
- [12] Elnaz Namazi, Jingyue Li, and Chaoru Lu. “Intelligent intersection management systems considering autonomous vehicles: a systematic literature review”. In: *IEEE Access* 7 (2019), pp. 91946–91965.
- [13] Kurt Dresner and Peter Stone. “A multiagent approach to autonomous intersection management”. In: *Journal of artificial intelligence research* 31 (2008), pp. 591–656.
- [14] John Houry and Joud Houry. “Passive, decentralized, and fully autonomous intersection access control”. In: *17th international IEEE conference on intelligent transportation systems (ITSC)*. IEEE, 2014, pp. 3028–3033.
- [15] Guni Sharon and Peter Stone. “A protocol for mixed autonomous and human-operated vehicles at intersections”. In: *International Conference on Autonomous Agents and Multiagent Systems*. Springer, 2017, pp. 151–167.

- [16] Pengfei Taylor Li and Xuesong Zhou. “Recasting and optimizing intersection automation as a connected-and-automated-vehicle (CAV) scheduling problem: A sequential branch-and-bound search approach in phase-time-traffic hypernetwork”. In: *Transportation Research Part B: Methodological* 105 (2017), pp. 479–506.
- [17] Xi Liu, Ping-Chun Hsieh, and PR Kumar. “Safe intersection management for mixed transportation systems with human-driven and autonomous vehicles”. In: *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE. 2018, pp. 834–841.
- [18] *Intelligent transportation systems Vehicle-to-Vehicle Technologies Expected to Offer Safety Benefits, but a Variety of Deployment Challenges Exist*. <https://www.gao.gov/assets/gao-14-13.pdf>. Accessed: 06-03-2021.
- [19] Gabriel Rodrigues de Campos, Paolo Falcone, and Jonas Sjöberg. “Autonomous cooperative driving: a velocity-based negotiation approach for intersection crossing”. In: *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. IEEE. 2013, pp. 1456–1461.
- [20] Xiangjun Qian et al. “Decentralized model predictive control for smooth coordination of automated vehicles at intersection”. In: *2015 European control conference (ECC)*. IEEE. 2015, pp. 3452–3458.
- [21] Peiqun Lin et al. “Autonomous vehicle-intersection coordination method in a connected vehicle environment”. In: *IEEE Intelligent Transportation Systems Magazine* 9.4 (2017), pp. 37–47.
- [22] Muhammed O Sayin et al. “Information-driven autonomous intersection control via incentive compatible mechanisms”. In: *IEEE Transactions on Intelligent Transportation Systems* 20.3 (2018), pp. 912–924.
- [23] Kazi Shah Nawaz Ripon, Jostein Solaas, and Håkon Dissen. “Multi-objective evolutionary optimization for autonomous intersection management”. In: *Asia-Pacific Conference on Simulated Evolution and Learning*. Springer. 2017, pp. 297–308.
- [24] Arda Kurt and Ümit Özgüner. “Hierarchical finite state machines for autonomous mobile systems”. In: *Control Engineering Practice* 21.2 (2013), pp. 184–194.
- [25] Scott Drew Pendleton et al. “Perception, planning, control, and coordination for autonomous vehicles”. In: *Machines* 5.1 (2017), p. 6.
- [26] Brian Paden et al. “A survey of motion planning and control techniques for self-driving urban vehicles”. In: *IEEE Transactions on intelligent vehicles* 1.1 (2016), pp. 33–55.
- [27] Edward F Moore et al. “Gedanken-experiments on sequential machines”. In: *Automata studies* 34 (1956), pp. 129–153.
- [28] George H Mealy. “A method for synthesizing sequential circuits”. In: *The Bell System Technical Journal* 34.5 (1955), pp. 1045–1079.
- [29] David Harel. “Statecharts: A visual formalism for complex systems”. In: *Science of computer programming* 8.3 (1987), pp. 231–274.
- [30] Steven Waslander and Jonathan Kelly. *Motion Planning for Self-Driving Cars*. <https://www.coursera.org/lecture/motion-planning-self-driving-cars/lesson-1-behaviour-planning-tPdVH>. [Online; accessed 18-Aug-2020]. 2020.
- [31] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
- [32] Wilko Schwarting, Javier Alonso-Mora, and Daniela Rus. “Planning and decision-making for autonomous vehicles”. In: *Annual Review of Control, Robotics, and Autonomous Systems* (2018).
- [33] Nikos Vlassis, Matthijs TJ Spaan, et al. “A fast point-based algorithm for POMDPs”. In: *Benelearn 2004: Proceedings of the Annual Machine Learning Conference of Belgium and the Netherlands*. 2004, pp. 170–176.
- [34] Filecloud. *Top 5 Limitations of Machine Learning in an Enterprise Setting*. <https://www.getfilecloud.com/blog/2018/06/top-5-limitations-of-machine-learning-in-an-enterprise-setting/\#.Xz0ejzVcKUI>. [Online; accessed 18-Aug-2020]. 2020.
- [35] *Reglement verkeersregels en verkeerstekens 1990 (RVV 1990)*. <https://wetten.overheid.nl/BWBR0004825/2020-01-01>. Accessed: 05-04-2021.
- [36] Davor Hrovat et al. “The development of model predictive control in automotive industry: A survey”. In: *2012 IEEE International Conference on Control Applications*. IEEE. 2012, pp. 295–302.
- [37] Tobias Kunz and Mike Stilman. “Time-optimal trajectory generation for path following with bounded acceleration and velocity”. In: *Robotics: Science and Systems VIII* (2012), pp. 1–8.
- [38] Michael S Floater. “On the deviation of a parametric cubic spline interpolant from its data polygon”. In: *Computer Aided Geometric Design* 25.3 (2008), pp. 148–156.
- [39] Marko Lepetič et al. “Time optimal path planning considering acceleration limits”. In: *Robotics and Autonomous Systems* 45.3-4 (2003), pp. 199–210.
- [40] Chuck Lewin. *Mathematics of motion control profiles*. In: *Performance Motion Devices, Inc.: Westford, MA, USA* (2007), pp. 1–5.
- [41] José Román García Martínez et al. “Assessment of jerk performance s-curve and trapezoidal velocity profiles”. In: *2017 XIII International Engineering Congress (CONIIN)*. IEEE. 2017, pp. 1–7.
- [42] Jorge Villagra et al. “Smooth path and speed planning for an automated public transport vehicle”. In: *Robotics and Autonomous Systems* 60.2 (2012), pp. 252–265.
- [43] Hong Chen and Frank Allgöwer. “A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability”. In: *Automatica* 34.10 (1998), pp. 1205–1217.

- [44] Frank Allgöwer et al. “Nonlinear predictive control and moving horizon estimation—an introductory overview”. In: *Advances in control*. Springer, 1999, pp. 391–449.
- [45] Michael A Henson. “Nonlinear model predictive control: Current status and future directions”. In: *Computers & Chemical Engineering* 23.2 (1998), pp. 187–202.
- [46] F Kühne. “Predictive control of nonholonomic mobile robots”. PhD thesis. Master thesis, Federal University of Rio Grande do Sul, Porto Alegre, RS, Brazil, 2005.
- [47] Pranjal Biswas. “Human-like Trajectory Generation for Autonomous Driving”. MA thesis. De Zaale Eindhoven: Eindhoven University of Technology, 2019.
- [48] Yiqi Gao. “Model predictive control for autonomous and semiautonomous vehicles”. PhD thesis. UC Berkeley, 2014.
- [49] A Galip Ulsoy, Huei Peng, and Melih Çakmakci. *Automotive control systems*. Cambridge University Press, 2012.
- [50] Rajesh Rajamani. *Vehicle dynamics and control*. Springer Science & Business Media, 2011.
- [51] Georg Schildbach and Francesco Borrelli. “Scenario model predictive control for lane change assistance on highways”. In: *2015 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2015, pp. 611–616.
- [52] Eric W Weisstein. “Simpson’s rule”. In: <https://mathworld.wolfram.com/> (2003).
- [53] Joao P Hespanha. *Linear systems theory*. Princeton university press, 2018.
- [54] Kendall E Atkinson and Weimin Han. *Elementary numerical analysis*. Wiley New York, 1993.

APPENDIX A

A. Intersection use-cases

In this section the five different intersection types are shown and those intersection types also function as the use-cases for this research.

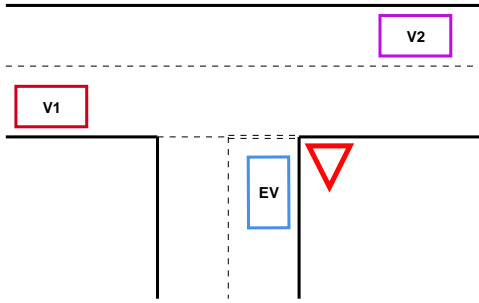


Fig. 12. Illustration of the T-intersection use-case, where the ego-vehicle (EV) is shown in blue.

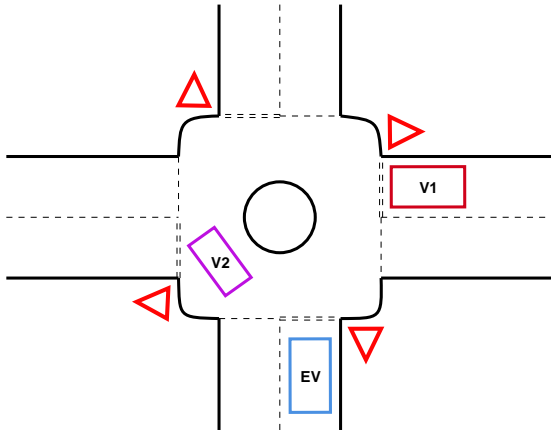


Fig. 13. Illustration of the roundabout use-case, where the ego-vehicle (EV) is shown in blue.

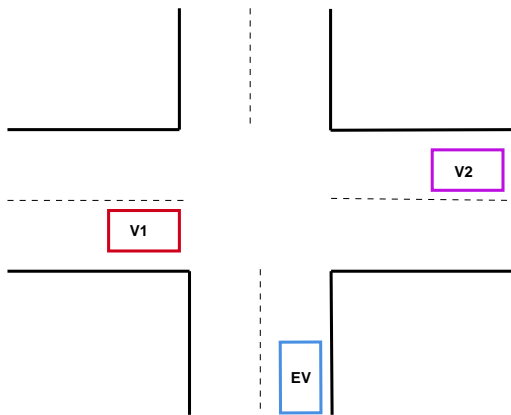


Fig. 14. Illustration of the no priority pre-assigned intersection (NPA fourway) use-case, where the ego-vehicle (EV) is shown in blue.

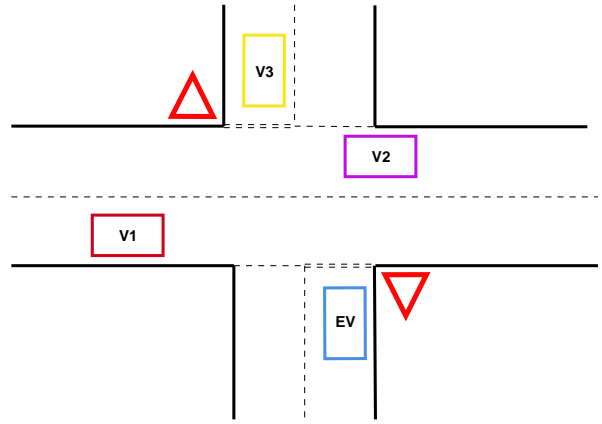


Fig. 15. Illustration of the priority pre-assigned intersection (PA fourway) use-case, where the ego-vehicle (EV) is shown in blue.

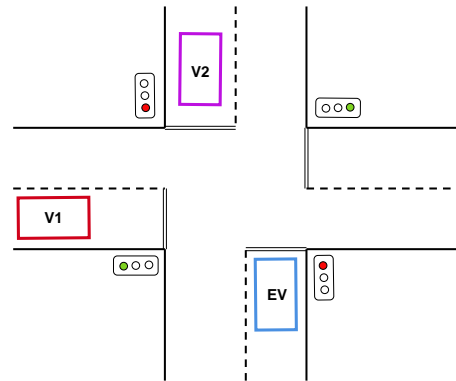


Fig. 16. Illustration of the traffic light intersection use-case, where the ego-vehicle (EV) is shown in blue.

B. Context information categorization

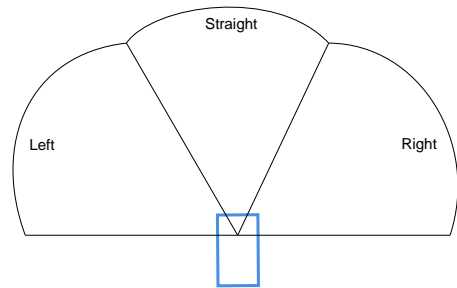


Fig. 17. The three detection zones of the radar sensors available on the ego-vehicle.

C. Hierarchical FSM

In this section level two and three of the hierarchical FSM are shown. The figures show the state transition conditions.

1) Hierarchical FSM level 2: High or Low priority

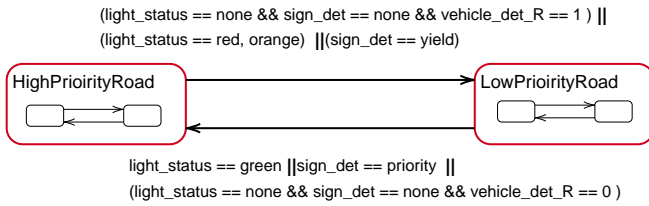


Fig. 18. Second level of the Level 3 hierarchical FSM, two states high priority and low priority road types.

2) Hierarchical FSM level 3 left turn: Stop or Go

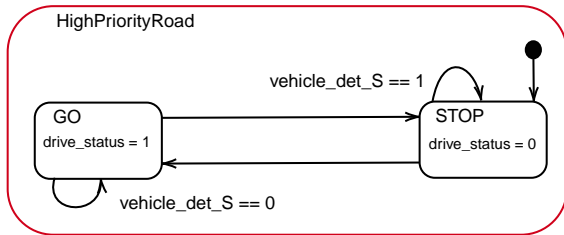


Fig. 19. Level 3 hierarchical FSM high priority left turn.

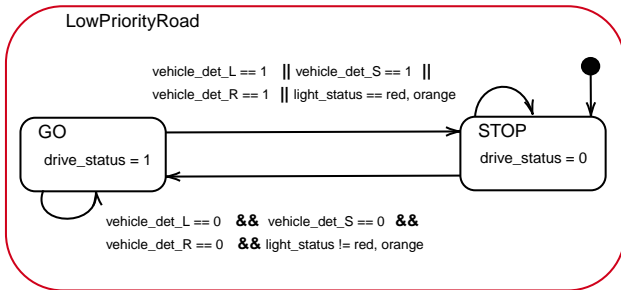


Fig. 20. Level 3 hierarchical FSM low priority left turn.

3) Hierarchical FSM level 3 straight: Stop or Go

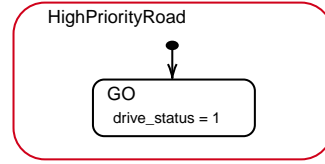


Fig. 21. Level 3 hierarchical FSM high priority straight.

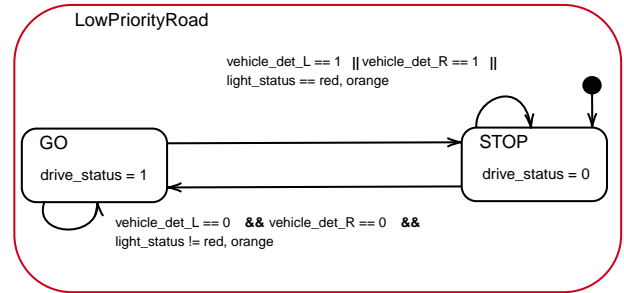


Fig. 22. Level 3 hierarchical FSM low priority straight.

4) Hierarchical FSM level 3 right turn: Stop or Go

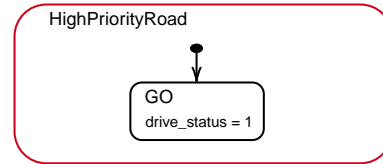


Fig. 23. Level 3 hierarchical FSM high priority right turn.

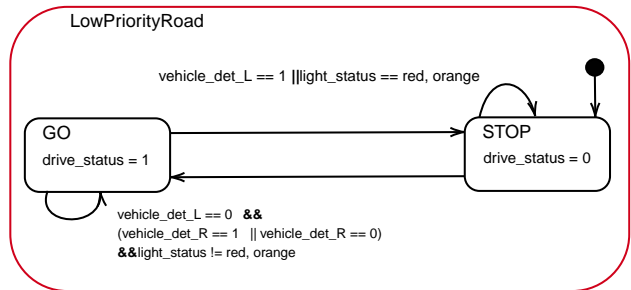


Fig. 24. Level 3 of hierarchical FSM low priority right turn.

D. Closest reference point calculation

The closest reference spline point is found by computing the hypotenuse (black lines in Fig. 25) between the current ego-vehicle state (blue point in Fig. 25) and all spline reference points (yellow points in Fig. 25). The spline point with the shortest hypotenuse (green line in Fig. 25) distance (*Euclidean distance*) is chosen as the first point for the linear time interpolation $\xi_{0|k}^{ref}$, in order to calculate the reference prediction states ($\xi_{i|k}^{ref}$) for the entire prediction horizon ($i = 0, \dots, N_p$).

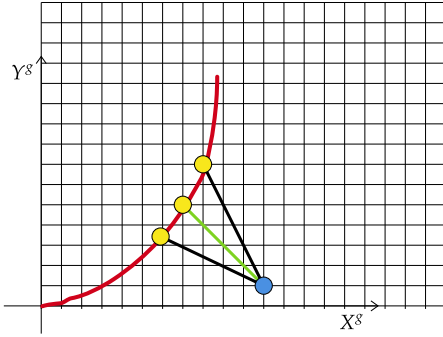


Fig. 25. Closest reference point calculation using the Euclidean distance, between the reference points (yellow points) and the current position of the ego-vehicle (blue point).

E. Kinematic bicycle model

Different coordinate frames are required to define the states of the ego-vehicle, the reference path and the vehicle inputs. Two main coordinate frames are used, which are shown in Fig. 26. Each of the coordinates frames are defined as follows:

- *Global coordinate frame* ($O^g X^g Y^g$): The origin of the global coordinate frame, at any time, is a fixed point in space and is denoted by the g superscript.
- *Ego-vehicle coordinate frame* ($O^e X^e Y^e$): The origin of the ego-vehicle coordinate frame, at any time, lies at the instantaneous Centre of Gravity (COG) of the ego-vehicle, with its X-axis aligned with the instantaneous heading of the ego-vehicle with respect to the global coordinate frame and is denoted by the e superscript.

The MPC algorithm needs a model in order to predict future states. In this section the choice of the model is explained. Extensive studies have been performed regarding vehicle dynamic modelling over the years [48–50]. Since the research in this paper is limited to low speed ($< 50km/h$) in urban intersection scenarios and not at the limits of vehicle handling, the vehicle model for the MPC algorithm will be a kinematic bicycle model shown in Fig. 27.

The kinematic bicycle model can be extended by also considering the body side-slip of the vehicle with respect to the COG of the ego-vehicle as done in [51]. Leading to the following equations of motions for the kinematic bicycle model with respect to the COG of the ego-vehicle

$$\dot{x}^g = v_x^e \cos(\phi^e + \beta^e), \quad (4a)$$

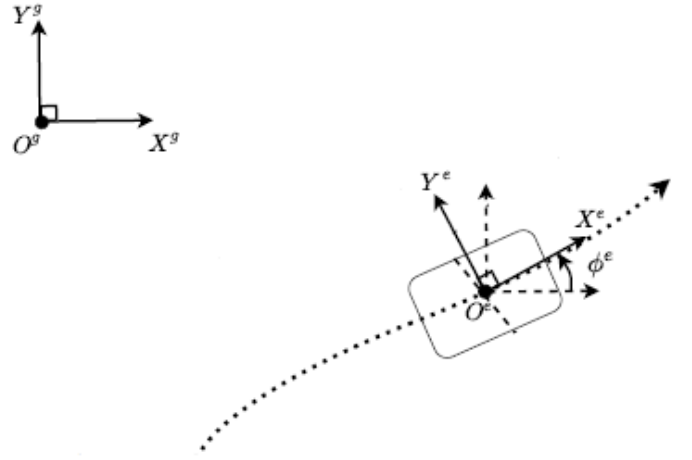


Fig. 26. Illustration of both world and local ego-vehicle coordinates [47]

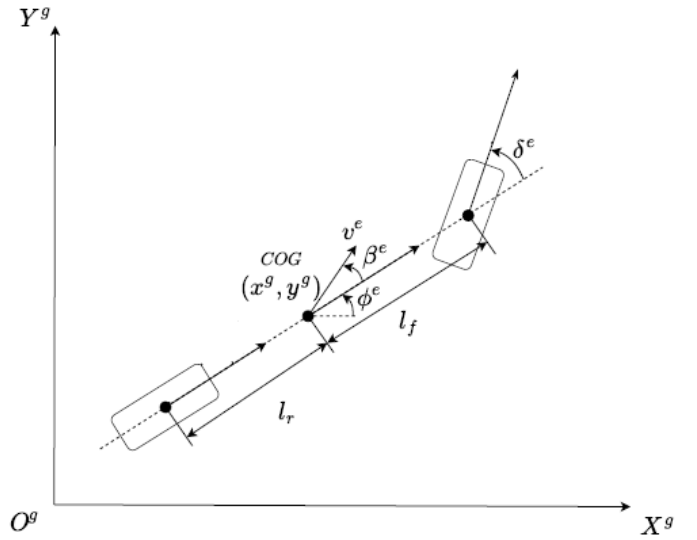


Fig. 27. Kinematic bicycle model [47]

$$\dot{y}^g = v_x^e \sin(\phi^e + \beta^e), \quad (4b)$$

$$\dot{\phi}^e = \frac{v_x^e \cos(\beta^e)}{L} \tan(\delta^e), \quad (4c)$$

$$\beta^e = \arctan\left(\frac{l_r}{L} \tan(\delta^e)\right), \quad (4d)$$

$$\dot{v}_x^e = a_{x,u}^e, \quad (4e)$$

where v_x^e is the longitudinal velocity in $[m/s]$, ϕ^e is the heading angle in $[rad]$, δ^e is the steering angle of the front wheel in $[rad]$, L is the wheelbase in $[m]$, β^e the body side-slip angle, l_r in $[m]$ is the distance of the rear axle from the COG and $a_{x,u}^e$ is the local longitudinal acceleration of the vehicle in $[m/s^2]$. The equations above are compactly represented by

$$\dot{\xi} = m^{KB}(\xi, u) \quad (5)$$

where $\xi = [x^g, y^g, \phi^e, v_x^e]^T$ is the state vector, while $u = [a_{x,u}^e, \delta^e]^T$ is the input vector.

F. Successive linearization

As explained in Section V-B, the MPC algorithm requires a LTV prediction model. This linear prediction model is calculated using successive linearization of the non-linear bicycle model explained in Section A-E. Fig. 28 shows the interaction between, the reference generator, successive linearization block, non-linear plant model and the MPC. The math behind Successive Linearization block is done for a simulation step k and is repeated at the next simulation step $k + 1$.

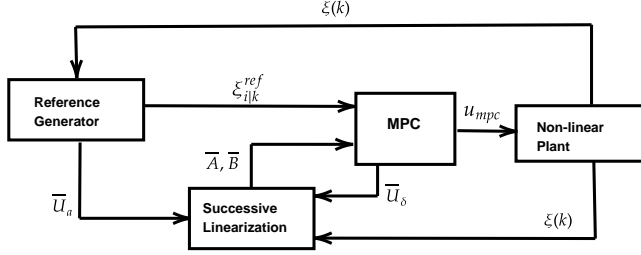


Fig. 28. Illustration of the closed-loop LTV MPC control diagram, showing the interaction between the reference generator, MPC controller and the plant model.

The goal of the successive linearization is to provide the MPC algorithm with discretized linear state-space matrices \bar{A} , \bar{B} as shown in Fig. 28 and given by,

$$\begin{aligned} \bar{A} &= (A_i, \dots, A_{i+N_p}) \\ \bar{B} &= (B_i, \dots, B_{i+N_p}) \end{aligned} \quad (6)$$

where i represents the current prediction step. In order to get the discrete matrices A_i, B_i , the continuous time linear state-space matrices $A_t(i)$ and $B_t(i)$ need to be discretized. This is done using the *Simpson's Rule* (as explained in detail in [52]). The continuous time linear state-space matrices $A_t(i)$ and $B_t(i)$ at each prediction step i are generated by evaluating the Jacobian matrices [53],

$$\begin{aligned} A_t(i) &= \frac{\partial m^{KB}(\bar{\xi}(i), \bar{U}(i))}{\partial \xi}, \\ B_t(i) &= \frac{\partial m^{KB}(\bar{\xi}(i), \bar{U}(i))}{\partial u}, \end{aligned} \quad (7)$$

where m^{KB} represents the kinematic bicycle model (5), $\bar{\xi}(i)$ is an approximation of the prediction state, $\bar{U}(i)$ is an approximation of the input sequence. $\bar{U}(i)$ is given by,

$$\bar{U}(i) = \begin{bmatrix} \bar{U}_\delta \\ \bar{U}_a \end{bmatrix} = \begin{bmatrix} \bar{u}_\delta(i), \dots, \bar{u}_\delta(i + N_p - 1) \\ \bar{u}_a(i), \dots, \bar{u}_a(i + N_p - 1) \end{bmatrix} \quad (8)$$

where \bar{U}_δ is the previous calculated $(k-1)$ steering angle input sequence by the MPC algorithm. This sequence is initialized to zeros for the initial simulation step $(k=1)$. \bar{U}_a is a linear approximation of the acceleration input sequence which is calculated by

$$\bar{u}_a(i) = \frac{v_x^{ref}(i+1) - v_x^{ref}(i)}{T_s}, \forall i = 1, \dots, i + N_p - 1, \quad (9)$$

where v_x^{ref} is longitudinal reference velocity generated by the velocity profiler, T_s is the sample time. The next approximation of the prediction state $\bar{\xi}(i+1)$ is done by numerical approximation using the *forward Euler method* [54], which is given by,

$$\bar{\xi}_n(n+1) = \bar{\xi}_n(n) + \frac{T_s}{h} m^{KB}(\bar{\xi}_n(n), \bar{U}(i)), \quad \forall n = 1, \dots, h, \quad (10a)$$

$$\bar{\xi}_n(1) = \bar{\xi}(i) \quad (10b)$$

$$\bar{\xi}(1) = \xi(k) \quad (10c)$$

$$\bar{\xi}(i+1) = \bar{\xi}_n(h) \quad (10d)$$

where i is the current prediction step, $(h=10)$ is the amount of increment steps, $(T_s=0.1)$ is sample time of the simulation. The initial approximation $\bar{\xi}(1)$ is equal to initial state of the current simulation step $\xi(k)$. All steps above are done for the entire prediction horizon N_p (i.e., $i=1, \dots, N_p-1$).

APPENDIX B

A. MPC parameter tuning results

1) *Prediction horizon tuning*: Firstly, the prediction horizon N_p is tuned, the values for the MPC parameters Q, R, N_p are given in Table VII.

TABLE VII
PREDICTION HORIZON TUNING VALUES

| | N1 | N2 | N3 | N4 |
|-------|---------|---------|---------|---------|
| Q | 1 1 1 1 | 1 1 1 1 | 1 1 1 1 | 1 1 1 1 |
| R | 0 0 | 0 0 | 0 0 | 0 0 |
| N_p | 3 | 5 | 10 | 20 |

TABLE VIII
RMS ERROR VALUES FOR DIFFERENT PREDICTION HORIZON LENGTHS

| N_p | RMSE value | | | |
|-------|------------|---------|--------------|-------------|
| | x [m] | y [m] | ϕ [rad] | V_x [m/s] |
| 3 | 2.120 | 2.568 | 0.363 | 1.436 |
| 5 | 0.015 | 0.033 | 0.052 | 0.028 |
| 10 | 0.014 | 0.036 | 0.052 | 0.028 |
| 20 | 0.013 | 0.036 | 0.052 | 0.028 |

The RMS error (RMSE) of the different states (shown in Table VIII) show that a longer prediction horizon $N_p > 10$ does not lead to a significant change in RMS error. So in order to reduce the computation time a prediction horizon of $N_p = 10$ is chosen.

2) *State cost matrix tuning*: Secondly, the state cost matrix Q is tuned, the values for the MPC parameters Q, R, N_p are given in Table IX.

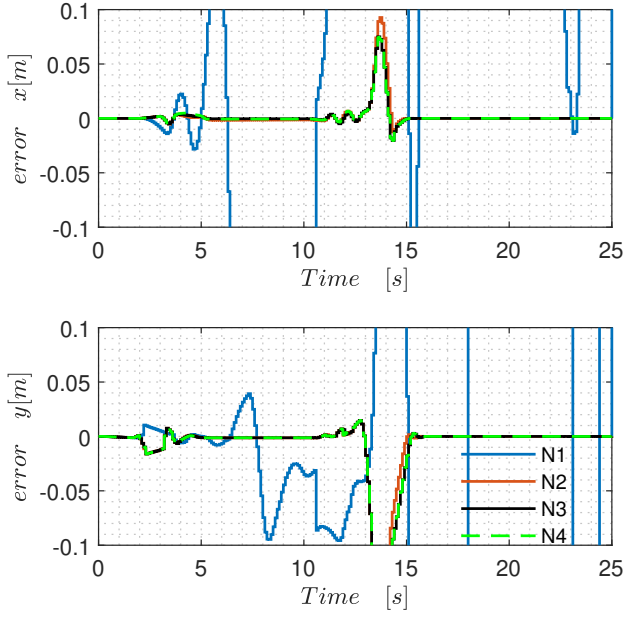


Fig. 29. Longitudinal and lateral position error against time for different prediction horizon lengths.

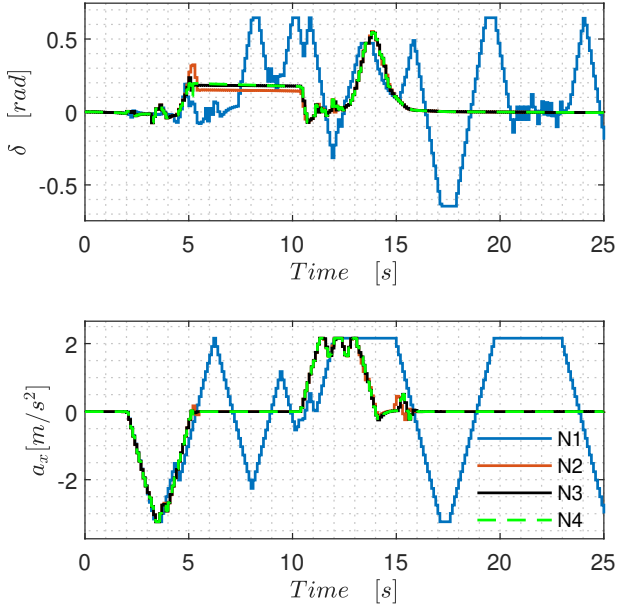


Fig. 30. Actuator inputs against time for different prediction horizon lengths.

TABLE IX
STATE COST MATRIX TUNING VALUES

| | Q1 | Q2 | Q3 | Q4 |
|-------|---------|---------|---------|---------------|
| Q | 1 1 1 1 | 1 1 0 1 | 0 0 1 1 | 0.2 0.2 0 0.8 |
| R | 0 0 | 0 0 | 0 0 | 0 0 |
| N_p | 10 | 10 | 10 | 10 |

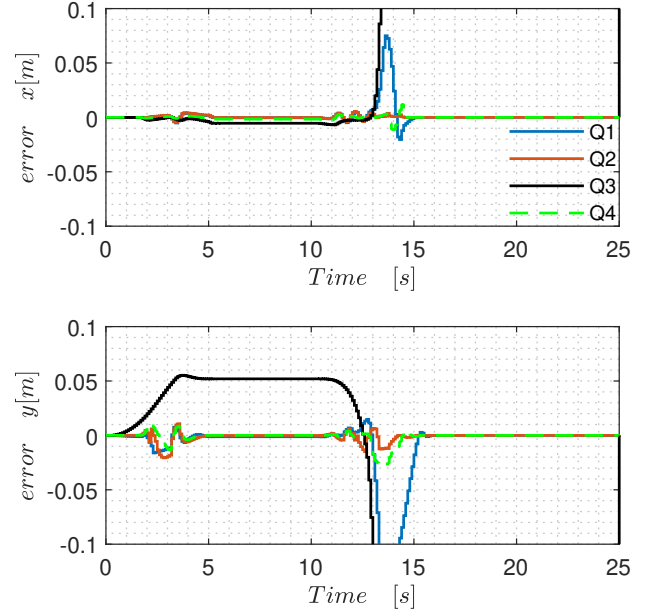


Fig. 31. Longitudinal and lateral position error against time for different state cost matrices.

TABLE X
RMS ERROR VALUES FOR DIFFERENT STATE COST MATRICES

| Q | RMSE value | | | |
|---------------|------------|---------|--------------|-------------|
| | x [m] | y [m] | ϕ [rad] | V_x [m/s] |
| 1 1 1 1 | 0.011 | 0.04 | 0.052 | 0.028 |
| 1 1 0 1 | 0.002 | 0.005 | 0.054 | 0.010 |
| 0 0 1 1 | 0.716 | 0.678 | 0.007 | 0.003 |
| 0.2 0.2 0 0.8 | 0.002 | 0.006 | 0.054 | 0.009 |

Fig. 31 shows that the state cost matrix of Q4 (i.e., $Q = \text{diag}(0.2 \ 0.2 \ 0 \ 0.8)$) leads to an acceptable RMS error of the different states (as shown in Table X) and does not lead to an uncomfortable steering inputs (as shown in Fig. 32).

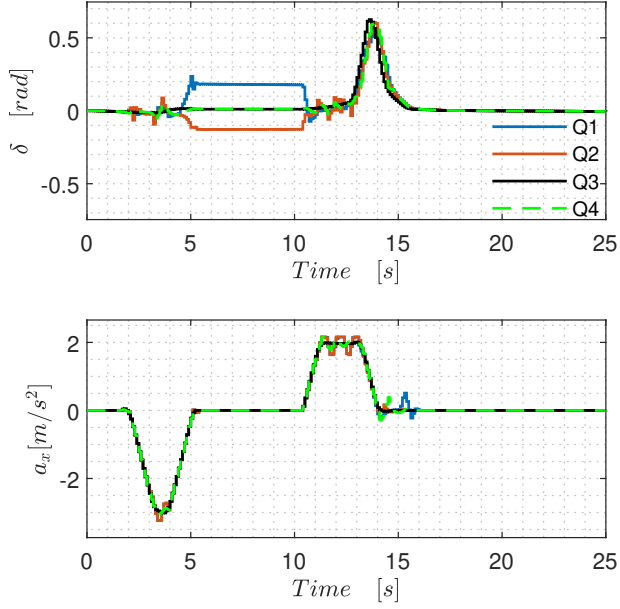


Fig. 32. Actuator inputs against time for different state cost matrices.

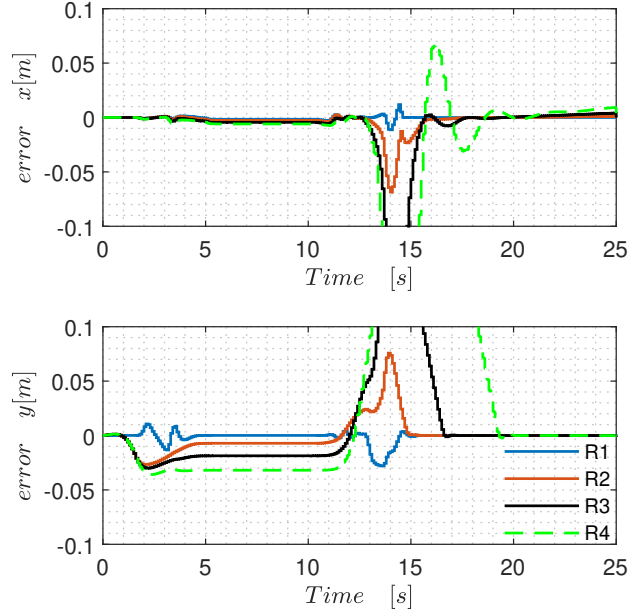


Fig. 33. Longitudinal and lateral position error against time for different input cost matrices.

3) *Input cost matrix tuning*: Thirdly the input cost matrix R is tuned, the values for the MPC parameters Q, R, N_p are given in Table XI.

TABLE XI
INPUT COST MATRIX TUNING VALUES

| | R1 | R2 | R3 | R4 |
|-------|---------------|-----------------|---------------|---------------|
| Q | 0.2 0.2 0 0.8 | 0.2 0.2 0 0 0.8 | 0.2 0.2 0 0.8 | 0.2 0.2 0 0.8 |
| R | 0 0 | 0.05 0 | 0.1 0 | 0.15 0 |
| N_p | 10 | 10 | 10 | 10 |

TABLE XII
RMS ERROR VALUES FOR DIFFERENT INPUT COST MATRICES

| R | RMSE value | | | |
|--------|------------|---------|--------------|-------------|
| | x [m] | y [m] | ϕ [rad] | V_x [m/s] |
| 0 0 | 0.002 | 0.006 | 0.054 | 0.009 |
| 0.05 0 | 0.012 | 0.016 | 0.057 | 0.017 |
| 0.1 0 | 0.047 | 0.066 | 0.065 | 0.035 |
| 0.15 0 | 0.108 | 0.172 | 0.080 | 0.053 |

In order to reduce the steering wobble at $t = 4s$ and $t = 12s$ in Fig.34 and to have an acceptable RMS error on the states (as shown in Table XII the input cost matrix will be equal to $R = \text{diag}(0.05 \ 0)$).

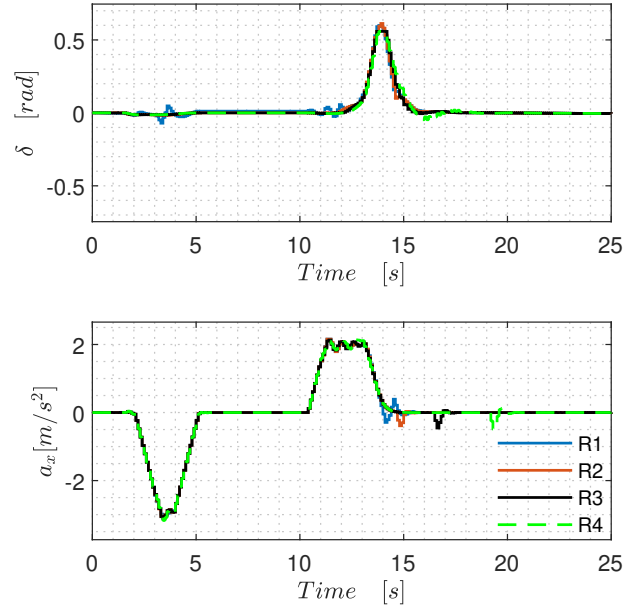


Fig. 34. Actuator inputs against time for different input cost matrices.

B. Additional simulation results

In this section additional results for the ego-vehicle driving straight through at a NPA fourway are given, as explained in Section VI-D.

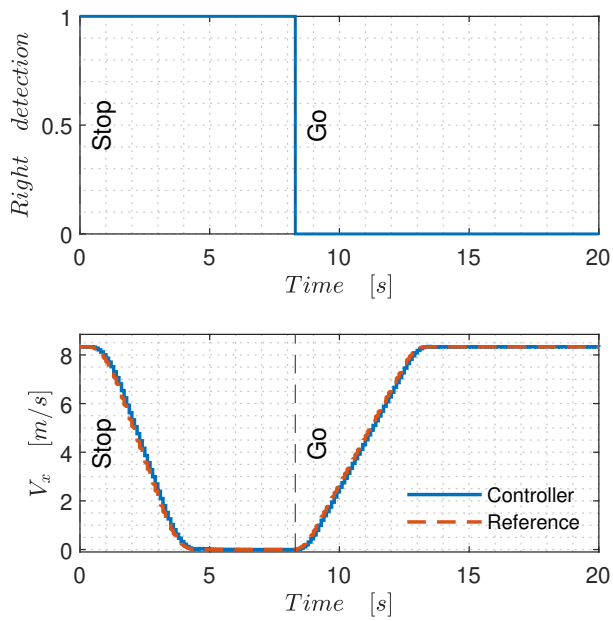


Fig. 35. The longitudinal velocity of the ego-vehicle for the NPA fourway test-case as a function of time.

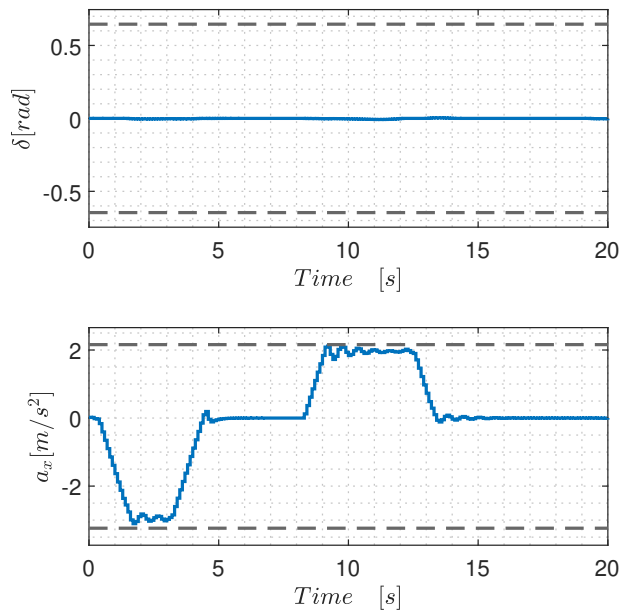


Fig. 36. Actuator inputs calculated by the MPC algorithm for the NPA fourway test-case, the horizontal dotted lines show the actuator limits