

Autonomous overtaking maneuver under complex driving conditions

Jiyo Jolly Palatti



Thesis submitted for examination for the degree of Master of Science in Technology under the dual degree program of EIT Digital Master School.

Espoo 21.12.2020

Supervisors

Prof. Ville Kyrki

Prof. Sahin Albayrak

Advisors

Dr Andrei Aksjonov

Dr. Orhan Can Görür



Aalto University
P.O.BOX 11000, 00076 AALTO
www.aalto.fi



Technische Universität Berlin
Franklinstrasse 28-29, 10587 Berlin
www.av.tu-berlin.de

Copyright © 2020 Jiyo Jolly Palatti

Hereby I declare that I wrote this thesis myself with the help of no more than the mentioned literature and auxiliary means.

Espoo, 21.12.2020

.....
(Signature [Jiyo Jolly Palatti])

Author Jiyo Jolly Palatti

Title Autonomous overtaking maneuver under complex driving conditions

Degree programme Master's Programme in ICT Innovation (EIT Digital Master School)

Major Autonomous Systems

Code of major ELEC3055

Supervisors Prof. Ville Kyrki, Prof. Sahin Albayrak

Advisors Dr Andrei Aksjonov, Dr. Orhan Can Görür

Date 21.12.2020

Number of pages 68+2

Language English

Abstract

The thesis tackles the planning and control of autonomous overtaking and subsequent lane-keeping. While existing solutions attempt multi-lane overtaking involving simple and static scenarios, the focus here is on single lane overtaking which require minimal intrusion on to the adjacent lane in dynamically changing conditions. The method proposed utilizes a heuristic rule-based strategy to select optimal maneuvers and then uses a combination of safe and reachable sets to iteratively generate intermediate reference targets based on the desired maneuver. A nonlinear model predictive controller then plans dynamically feasible trajectories to these intermediate reference targets that avoid collisions. The proposed method was implemented and tested under 7 different scenarios that cover many complex lane-keeping and overtaking scenarios using the CARLA simulation engine with ROS (Robotic Operating System) framework for inter-component communication with model predictive controller developed using MATLAB. In every tested scenario, the proposed planning and control paradigm was able to select the best course of action (maneuver) and execute the same without collisions with other nearby vehicles.

Keywords Intelligent Control, Autonomous Vehicles, Behaviour Planning, Trajectory optimization, Predictive control, Overtaking, Robotic Operating System, MATLAB

Preface

The help and support of several people have contributed to the achievements of this project. I want to thank Dr. Andrei Aksjonov for his unwavering support and advice at every step. I extend my sincere gratitude towards Prof. Dr. Ville Kyrki for offering me the opportunity to perform my thesis at the Intelligent Robotics Group at Aalto University and providing valuable supervision and direction.

I would also like to thank my supervisor Dr. Şahin Albayrak and advisor Dr. Orhan Can Görür from TU Berlin, for their supervision and valuable review comments. I wish to thank Dr. Alcan Gokhan for providing his expert guidance on many topics during the course of the thesis.

Completion of this thesis would not have been possible without the help and encouragement of my friends and family, especially, my sister and my friend Krishnananda for their indispensable support during the writing phase.

Otaniemi, 21.12.2020

Jiyo Jolly Palatti

Contents

Abstract	4
Preface	5
Contents	6
Symbols & Abbreviations	8
1 Introduction	9
1.1 Motivation	10
1.2 Problem Statement	11
1.3 Objectives	12
2 Background	14
2.1 Autonomous Driving Systems	14
2.1.1 Sensors	14
2.1.2 Perception	14
2.1.3 Planning	15
2.1.4 Control	16
2.2 Planning for Autonomous Overtaking	17
2.2.1 Tackling Behavioural Planning	17
2.2.2 Tackling Trajectory Planning	21
2.2.3 Combined Behavioural/Trajectory Planning	25
2.2.4 Discussion	25
3 Autonomous Overtaking Method	27
3.1 Assumptions	27
3.2 Proposed Pipeline	27
3.3 Background Information	28
3.3.1 Conventions	28
3.3.2 Vehicle Model	28
3.4 Behaviour Planning	30
3.4.1 Generating Reachable Set	30
3.4.2 Generating Safe Set	31
3.4.3 Behaviour Selection	34
3.5 Trajectory Planning	38
3.5.1 Obstacle Avoidance Constraints	39
3.5.2 Nonlinear Model Predictive Controller Trajectory Planner	40
4 Experimental Design	43
4.1 Implementation Details	43
4.1.1 Robotic Operating System	43
4.1.2 CARLA Simulation Platform	43
4.1.3 Model Predictive Control Toolbox	44
4.1.4 Pipeline Development	44
4.2 Test Scenarios	46

5 Results	49
5.1 Scenarios with No Overtaking	49
5.2 Scenarios with Overtaking	52
5.3 Discussion	56
5.3.1 Limitations	57
5.3.2 Summary	58
6 Conclusions	59
6.1 Future Work	59
References	61
A Parameters Used	69

Symbols & Abbreviations

Abbreviations

AD	Automated Driving
AV	Autonomous Vehicle
CFTOC	Constrained Finite-Time Optimal Control
DOF	Degrees of Freedom
EV	Ego vehicle
ILQR	Iterated Linear Quadratic Regulator
LQR	Linear Quadratic Regulator
LV	Lead vehicle (Vehicles other than the ego car)
MPC	Model Predictive Controller
NMPC	Nonlinear Model Predictive Controller
PID	Proportional-Integral-Derivative
RL	Reinforcement Learning

Symbols

a	Acceleration [ms^{-2}]
B	Boundary of the Lead Vehicles –
β	Angle between velocity vector & longitudinal axis of the car. [rad]
δ_f	Front steering angle [rad]
δ_r	Rear steering angle [rad]
D_{max}	Maximum length of the vertex [m]
R	Reachable set [$\mathbb{R}^{m \times n}$]
S	Safe set [$\mathbb{R}^{m \times n}$]
S_{SR}	Safe and reachable set [$\mathbb{R}^{m \times n}$]
T_h	Planning Horizon [s]
p_{des}	Desired position [x, y]
p_{EV}	Position of Ego Vehicle [m, m]
p_{ref}	Intermediate reference position [x, y]
U_{LV}	Obstacle potential [–]
U_{road}	Road potential [–]
v	Velocity [ms^{-1}]
v_{des}	Desired Velocity [ms^{-1}]
v_{EV}	Velocity of Ego Vehicle [ms^{-1}]
v_{LV}	Velocity of Lead Vehicle [ms^{-1}]
x	Longitudinal position [m]
X_{des}	Desired reference state [x, y, ψ, v]
X_{ref}	Intermediate reference state [x, y, ψ, v]
ψ	Heading angle [rad]
y	Lateral position [m]

1 Introduction

Over the past century, automobiles have become the main mode of transportation. With the ability to mass-produce vehicles that are safe, reliable, and affordable, the automotive industry has seen exponential growth [1]. However, these vehicles still require a skilled human driver to be able to get from one location to another. Unprecedented advances in information technology have led to the transformation of old-fashioned and mechanical means of commuting to a smart and infotainment-rich mode of transportation. Integration of electronics into automobiles has enabled various features, namely Advanced Driver Assistance Systems (ADAS) and Infotainment-On The-Go. ADAS features help drivers to avoid drifting into adjacent lanes or making unsafe lane changes, or warn drivers of other vehicles behind them when they back up. ADAS systems are nowadays even capable of predicting collisions and automatically intervening to apply brakes and take evasive maneuvers. These features have already helped save lives and prevent injuries, thus, overall improving safety and convenience of the driver and passengers [2].

Even though ADAS features help to some extent, driving tasks are still primarily performed by a human. A brief review of the traffic accidents statistics [3] reveals that the amount of road traffic deaths has reached 1.35 million per year worldwide. Road traffic injuries are now the primary cause of death of people aged 5-29 years. Automobile-related accidents has also caused around 20-50 million injuries worldwide, often resulting in long-term disabilities. 94% of all automobile crashes were caused by human error [4]. On the other hand, the machines are not characterized by fatigue, boredom or distraction. Hence, automotive systems tend to minimize an impact of human factors on transportation safety. Thus, the end goal is to automate the whole process so as to avoid human intervention altogether. Autonomous Vehicle (AV), also known as a self-driving car, is a promising technology, which is able to sense its environment and perform the driving task without a human in control. Recent advances in computing power can enable machine-learning-based techniques to both process data from various sensors and take necessary control actions to perform the driving task autonomously. This has the advantages of improving safety, efficiency, comfort and convenience.

Improving Safety

Human beings are error-prone and easily distracted. Human drivers also experience driver fatigue and require restful sleep before performing optimally [5]. In many high stake situations, human drivers succumb to emotional reactions that could lead to irrational decisions and, thus, road accidents. AVs can be programmed to perform driving tasks without compromising safety. Since they are essentially machines, they do not feel fatigue and are able to always make rational decisions no matter the situation. AVs are equipped with sensors which have significantly better range and sensitivity. Hence, they are able to collect a lot more data about the environment compared to human drivers, who are able to only see in the visible spectrum.

Improving Efficiency

An AV can be efficient in numerous ways. (i) It can be programmed to optimize energy efficiency and time. (ii) It can follow optimal trajectories that are devoid of unnecessary and inefficient maneuvers. (iii) Once most of the vehicles on the road are automated,

significant improvements in traffic efficiency and lower congestion may be achieved. This is possible due to the fact that AVs can potentially communicate and coordinate with each other at a scale that is impossible for humans [6]. Also, private vehicles are only used 10% of the time while sitting idle for the rest. AVs can accelerate ride/car-sharing technologies changing the concept of ownership of private vehicles. This could mean that the personal mobility requirements can be met with approximately a third of the total number of personal vehicles that are on the roads today [6]. All of this may translate to a significantly lower impact on the environment and help us reach our climate change goals.

Improving Comfort & Convenience

Driving is a complex task that requires concentration and high levels of alertness to guarantee the safety of other drivers and co-passengers. Driving task can be physically and mentally exhausting to human drivers. Autonomous Driving (AD) can alleviate this burden and increase convenience and productivity of passengers on the go. Passengers can potentially, on-demand, request a pickup from any location, reach their destinations, and go about their way, not worrying about parking. AVs can also be programmed to adapt and provide a consistent driving experience to passengers based on their preferences. The concept of car-sharing discussed above may also lead to a reduction in the costs of car ownership and maintenance.

1.1 Motivation

Current AV technologies offer different levels of autonomy ranging from no autonomous features (which constitute most of the vehicles on today's roads) to recent offerings from Tesla, Inc. (Palo Alto, California) and Waymo LLC (Mountain View, California, U.S.), which enable autonomous driving in certain structured road conditions, though, still requiring partial human supervision [7]. These scenarios represent only a small subset of all the complex situations that a truly driverless system should be capable of handling. A useful classification is the 'SAE Levels of Driving Automation' which gauge the level of automation of a vehicle, ranging from "No Driving Automation (level 0)" to "Full Driving Automation (level 5)" [8]. Figure 1 illustrates this in detail. The holy grail of AVs will be a system that is capable of handling each and every driving situation that may arise without any human intervention, ensuring the safety of the passengers.

A Fully Automated System (Level 5) should be capable of handling multitudes of complex driving maneuvers that are involved in a typical driving situation, such as:

1. Lane Keeping
2. Lane Changing
3. Overtaking
4. Turns & Intersections
5. Roundabouts
6. Parking
7. Emergency maneuvers



SAE J3016™ LEVELS OF DRIVING AUTOMATION

	SAE LEVEL 0	SAE LEVEL 1	SAE LEVEL 2	SAE LEVEL 3	SAE LEVEL 4	SAE LEVEL 5
What does the human in the driver's seat have to do?	You <u>are</u> driving whenever these driver support features are engaged – even if your feet are off the pedals and you are not steering	You must constantly supervise these support features; you must steer, brake or accelerate as needed to maintain safety		You <u>are not</u> driving when these automated driving features are engaged – even if you are seated in “the driver’s seat”	When the feature requests, you must drive	These automated driving features will not require you to take over driving
What do these features do?	These are driver support features			These are automated driving features		
Example Features	• automatic emergency braking • blind spot warning • lane departure warning	• lane centering OR • adaptive cruise control	• lane centering AND • adaptive cruise control at the same time	• traffic jam chauffeur	• local driverless taxi • pedals/steering wheel may or may not be installed	• same as level 4, but feature can drive everywhere in all conditions

Figure 1: SAE J3016™: Levels of Driving Automation (extracted from **J3016B Taxonomy Definitions**).

Of these, overtaking is one of the hardest problems to solve as it is one of riskiest maneuvers for both drivers and riders. While overtaking, the vehicle is in the direct path of oncoming traffic, often at high speeds. In a head-on collision, the momentum of both vehicles contribute to create a severe impact. In 2018, 368,559 accidents with personal injury on rural roads were registered by the police in Germany [9]. These accidents also led to 3,275 deaths. About 4 percent of these accidents occurred while the driver was overtaking. In some regions, overtaking is banned on single lane roads to avoid accidents. The final goal in this thesis is to understand the innumerable intricacies involved in various overtaking scenarios, specifically those pertaining to single lane roads, identify the gaps in current approaches, and devise possible solutions.

1.2 Problem Statement

This thesis addresses the problem of planning and execution of overtaking maneuvers in an AV. Road layouts differ from single lane roads (common in suburban & country-side areas) to structured multi-lane highways that connect major cities. Road traffic consists of different types of vehicles ranging from bicycles and motorbikes up to huge trucks. The applicable road rules also factor into overtaking decisions. In highway driving scenarios with multiple lanes, vehicles can use them to execute multiple lane changes to overtake a slower moving vehicle. This is not an option on many roads that are single lanes especially those in suburban and country-side regions. Overtaking is allowed and sometimes unavoidable on these types of roads (which constitute the majority of the roads

in the world). Moreover, in many cases, one might encounter other vehicles that may be parked haphazardly on the roadside or curb. The automated driving (AD) system also might encounter situations where it might not be aware of the actual length of the vehicle/number of vehicles that it has to overtake at the start of the maneuver. Thus, it has to adapt to this unexpected situation. It is of great importance for AD systems to be able to execute overtaking maneuvers safely in all these scenarios to avoid a deadlock situation and reduce travel time. Many existing methods in the literature are suitable only for simple scenarios where the LV is static or moving at a constant velocity and the simulated road environment consists of multiple lanes for easy overtaking. These methods then are often tested in low-fidelity simulations which do not validate their applicability to real-world scenarios described above.

Thus, the specific problems, an AD system has to solve is as follows. Firstly, it has to understand the road geometry and keep track of other vehicles it might have to overtake. The subsequent planning decisions depend on an accurate and updated representation of the surroundings at each time step. Moreover, apart from the location, the footprint of these obstacles should also be considered for overtaking them without collisions. Secondly, the selection of the right maneuver has to be made. This decision is made based on the current location of the obstacles, road layout, and active road rules and regulations. Safety, comfort, and demands of the passengers also factor into this decision of the maneuver. It is a challenge to compute the optimal decision in every situation, which many existing methods in literature overlook. Finally, trajectories should be generated that correspond to the selected maneuver. These trajectories should be dynamically updated to reflect the changing road conditions and motion of the obstacle vehicles. It is also crucial that the generated trajectories be dynamically feasible, meaning vehicle mechanics can track them without fail. The trajectories should also be collision-free and relatively smooth to avoid discomfort to the passengers.

The AD system then needs to be validated using a driving simulator that can imitate real-world situations as close as possible. This can ensure that the gap between its performance on the test simulations and the real world can be significantly reduced.

1.3 Objectives

The primary objective of this thesis is to *develop a control architecture that performs overtaking maneuvers autonomously whilst ensures safety, and is able to handle dynamic and complex real-world scenarios.*

This control architecture should be able to:

- Form a representation of the immediate environment to identify road layouts, neighbouring vehicles, obstacles.
- Decide on actions that drive the vehicle closer to the final destination at the same time avoiding collisions and obeying road rules. Thus, the system has to decide if it is safe to overtake or keep following the lane.
- Generate trajectories based on decided maneuvers that are collision-free and feasible for the vehicle to follow.

This thesis proposes a control architecture that attempts to solve the problem of overtaking on single lane roads under complex situations by

1. Novel iterative selection of target point by using a combination of artificial potential fields and reachability set analysis that define the safe and reachable areas.
2. Adaptive planning of trajectory to the said target point by using an Nonlinear Model Predictive Controller (NMPC).

This approach has the following advantages.

1. Road geometries are better expressed in the form of potential fields which are more intuitive and computationally efficient for identifying safe areas.
2. Iterative selection of target points removes much of the environmental constraint handling from the purview of the NMPC.
3. Prediction horizons can be kept short which leads to reduction in computational requirements.

The proposed method, which draws inspiration from works of Dixit *et al.* [10], [11], chooses an appropriate maneuver (overtaking, lane keeping, etc.) based on situational factors and then selects safe, feasible intermediate reference targets that guide the subsequent trajectory generation using an NMPC to execute the chosen maneuver. It is also capable of detecting when overtaking is successful and switching to lane-keeping to resume journey to the final destination. The proposed method also includes a Proportional-Integral-Derivative (PID) controller that accurately tracks the generated trajectory by converting the corresponding optimal control actions to the form required by the simulator (Throttle, Brake and Steering angle).

The proposed architecture is then validated in simulation. Different scenarios that correspond to the challenges discussed in Section 1.2 will be setup. Factors like (i) success in selection of optimal maneuver, (ii) collision events, (iii) adherence to road layouts (iv) intrusion onto the adjacent lane during overtake, (vii) adaptability to change, (iv) violation of constraints, (v) quality of overall trajectory, will be used to evaluate the performance. Thus, the secondary objective of the thesis is to *develop a simulation setup which creates various scenarios to test the limits of this control architecture*.

The proposed control architecture was tested under 7 different simulated scenarios to evaluate its performance based on factors discussed. The results of these tests showed that the EV was able to select the appropriate maneuvers and, plan and execute them without colliding with any other vehicles.

The thesis is organized as follows. In Section 2, related literature is reviewed. Section 3 describes the proposed control architecture in detail. The experimental design and various scenarios that were used to test the proposed method are described in Section 4. The results of these test scenarios are discussed in Section 5. The final Section 6 concludes with a summary of the work and possible avenues to explore in the future.

2 Background

2.1 Autonomous Driving Systems

A system that performs AD is complex and consists of many individual modules that execute specific tasks in harmony to autonomously drive the car. These tasks include perception, planning and control of the AV. Each module is expected to perform specific sub tasks for successful execution of different driving maneuvers, such as overtaking or lane-keeping. A brief overview of a typical AD software stack is outlined in Figure 2.

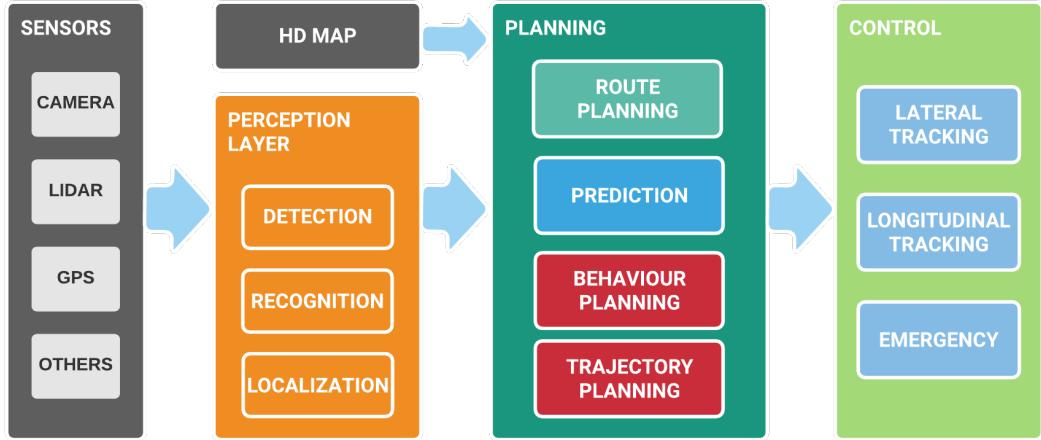


Figure 2: High-level Autonomous Driving Stack.

2.1.1 Sensors

Self-driving cars require a multitude of on-board sensors that stream information about the surrounding environment and state of the ego vehicle(EV). These include cameras, LIDARs, radars and, ultrasound sensors, which provide precise information about the environment. Global Positioning System(GPS) and Inertial Measurement Unit (IMU) are proprioceptive sensors that measure positions, velocities, and accelerations, crucial for planning and control. Current self-driving systems depend on several combinations of sensors, for example, Tesla’s (Tesla, Inc., Palo Alto, California) Autopilot [12] depends on cameras forgoing the use of LIDARs, whereas Google’s Waymo (Waymo LLC, Mountain View, California, U.S.) [13] utilizes both LIDARs and cameras. A hurdle is to make them reliable enough and easy to incorporate into the design of the car without affecting the aesthetics and aerodynamics of the car. Many of these sensors, especially traditional LIDARs, are expensive and are electromechanical in nature, relying on moving parts. Solid state LIDARs which are built entirely on a silicon chip are expected to solve these issues [14].

2.1.2 Perception

Information collected by the sensors requires processing and extraction of relevant features to create a coherent, consistent and reliable representation of the environment. Therefore, the perception module is crucial for any AD system. The module develops contextual

understanding of the environment, such as information about obstacles, road layouts, traffic signs and other road features. This understanding is hard to program classically and, thus, deep learning methods are particularly well suited, especially Convolutional Neural Networks, thereby making them the de facto standard for detection and recognition tasks [7]. The module also performs the important task of localizing the vehicle in its environment, generally using data from IMUs and GPS, combined with visual odometry. Although localization is predominantly done using classical algorithms such as Kalman filters and other similar methods [15], deep learning-based methods are being researched that can simultaneously perform localization as well as estimate and predict the motion of the surrounding environment (known as scene flow)[7]. Prediction of surrounding environment is of great importance in the planning stage, where it could influence the decision making process and the subsequent trajectories that are planned.

2.1.3 Planning

This module is responsible for generating a safe and optimal trajectory which utilizes perception information from the perception layer along with the current state of the EV, subject to any dynamic constraints that ensure the comfort and preferences of the passengers. The planning can be subdivided into the following modules described below.

Route Planning

The final goal of any AV system is to drive from one location to another. This requires planning a route from an initial location to the final destination using a map source that contains the road network information. Thus, a route planner provides the system with a high-level plan that the lower levels can use as a guide for planning decisions. Traditional “shortest path” algorithms such as Dijkstra [16] and A* [17] and its derivatives do not translate well to large road networks that span cities or even continents, as these graphs contain millions of edges. With latest advancements in route planning, algorithms can compute driving plans in milliseconds or even less at these continental scales [18]. Most proposed methods require a one-time pre-processing step after which they can return optimal routes. They are also able to incorporate real-time traffic information to account for time lost due to blocks or road congestions, thus, providing alternate routes that are faster [19].

Behaviour Planning

After obtaining a route plan, the AV must be able to judiciously navigate through it, meanwhile interacting with other vehicles, obeying traffic rules and following road conventions. This involves selecting between various driving behaviours. Take, for example, a scenario where the EV is stuck behind a slower moving LV. Overtaking might be possible only if there are no oncoming vehicles in the other lane. Otherwise, lane keeping behaviour must be continued keeping safe distance from the LV. Such a behaviour selection is made based on cautious inspection of immediate surroundings using the information from the perception module. Complexity is increased in a more dynamic setting where split-second decisions must be made. Rapidity, consistency, providentness and determinism are essential traits that any behaviour planning method for automated driving should possess. Section 2.2 will provide more information on various methods through which behaviour planning is achieved.

Trajectory/Path Planning

The driving maneuver chosen by the behaviour planning module needs to be expressed as a trajectory that the vehicle can track. It is also required that the generated trajectory ensures the passenger comfort, respects road geometry and rules meanwhile keeps safe distance from obstacles. This is made possible by the trajectory planner module, which plans the vehicle's transition from one feasible state to another while obeying the vehicle dynamics. Tracking of this trajectory should not require actuation commands that are not possible with the vehicle. Exact solutions to this problem are infeasible due to computational requirements and, therefore, approximate numerical methods are usually used. Section 2.2 below will go into more detail on these methods.

2.1.4 Control

The trajectory generated by the previous module should be tracked by effectively executing appropriate actuation commands, which is the responsibility of the control module. It consists of a closed-loop control system that uses an accurate vehicle model to control the steering, throttle and brake commands. Since an accurate state estimation is of prime importance, these controllers are also coupled with advanced state estimators such as Kalman filters. Controllers need to ensure robustness and stability. Hence, control methods with formal stability guarantees and capable of ensuring robustness are preferred. Classical methods like PID are used in commercially available ADAS capable vehicles. These, however, require expert tuning to have a good tracking performance and require more complicated formulations like Sliding Mode Control and Adaptive PID [20]. Advancements in vehicle modelling and onboard computational capability have enabled the use of optimal control methods, e.g., Model Predictive Control (MPC), Linear Quadratic Regulator (LQR) [20].

2.2 Planning for Autonomous Overtaking

The planning module is of utmost importance in autonomous overtaking maneuvers. This section goes into detail about control architectures in related works, especially the ones that focus on overtaking maneuvers. Some works have attempted to tackle the problem without adhering to the hierarchical approach discussed above and instead use a single control method to capture both behaviour and trajectory planning requirements. Works that have adopted the hierarchical layered approach are analysed in Section 2.2.

2.2.1 Tackling Behavioural Planning

Finite State Machines (FSMs)

A natural way to think about choosing between different driving behaviours is to consider them as states and using driving rules and heuristic-based conditions to transition between them in a FSM framework. This essentially means hand-engineering this rule-based state machine. This approach was frequently used in early attempts at AD, like DARPA Urban Challenge [21]. Ardelt *et al.* [22] described usage of hybrid state machines and decision trees to define the required driving behaviour depending on the situation and feasibility. The drawbacks of this approach is that they cannot handle all edge cases that arise in driving. It is very difficult to manually capture all possible behaviours and transitions in a state machine and is also prone to error [23]. Furthermore, usage of heuristic rules may lead to sub-optimal decisions regarding the behaviour. Another challenge is to handle uncertainty in intentions and trajectories of other participants like vehicles and pedestrians [18].

Fuzzy Logic Based

Fuzzy logic based behaviour planning proposed by Safiotti [24] is more flexible as they allow combining different recommendations from concurrent behaviours. Hagras [25] proposed a hierarchical Type 2 fuzzy controller to better handle uncertainties in the environment. It was then applied to indoor robot navigation where decision making is required for navigating through unstructured environments. Jaafar *et al.* [26] proposed a novel action selection method using fuzzy logic. Based on sensor inputs, different behaviours are assigned weights and a final action is selected using Huwicz criterion. These also suffer from the same lack of generalization that plague FSM based approaches as encoding all possible behaviours is cumbersome.

Markov Decision Processes (MDP)

The MDPs are a general framework to model decision making and planning in situations, where outcomes are partly stochastic in nature and partly dependent on decisions (actions). They are essentially an extension of Markov chains, where transition to future states depends only on current state. The fundamental issue of an MDP is to identify an optimal policy that stipulates the action to take at the current state to maximize the cumulative discounted reward over an infinite horizon [27].

Physical systems are never fully observable in practice. The measurements used to determine the state of the system inherently contain uncertainty due to noise. The states

may not be directly measurable and, hence, have to be estimated indirectly. The perception systems cannot fully ascertain the information of the surroundings due to occlusions and sensor limitations. These also inject uncertainty into planning [23]. Thus, Partially Observable Markov Decision Processes or POMDP[28] provides a superior framework that is better suited to characterize the evolution of the EV and environment consisting of other actors as a result of any decision. The states can be either continuous or discrete depending on formulation. The planning framework solves for the optimal sequence of actions(maneuvers) or policy that maximize a reward function. This reward function can be formulated to optimize for time or effort, penalize collisions-prone situations.

Brechtel *et al.*[29] attempts to optimize behaviour planning for intersection merging scenarios, by employing a continuous POMDP. This allows for a continuous representation of the traffic (pose, velocities etc.) compared to relying on a discretized symbolic representation for every special driving situation. The process/prediction model that defines the transition between states is formulated using a hierarchical Bayesian model, hand engineered from underlying vehicle physics and road agent interaction knowledge.

The most challenging aspect is the development of a prediction model that captures the transition probabilities between states. Computational complexity in finding optimal policy is also another hurdle that limits their applicability to real-time situations, especially those that require states and actions to be modeled as discrete values. Ulbrich *et al.* [30] explores a similar POMDP based method which tries to solve many of these issues, opting for mixed-integer spaces instead of purely continuous states. It also tries to reduce computational complexity in finding optimal policy by keeping planning horizons short, having sparse action choices at each state and limiting planning accuracy. Significant breakthroughs in research on better POMDP solvers and prediction models make these more viable for use in behaviour planning [31]. However, these works still do not cover all driving situations and focus only on certain scenarios e.g. lane changing behaviour or intersection merging. Any attempts to generalize would mean exponentially increasing complexity in the prediction model.

Deep Reinforcement Learning

One way to side-step formulation of the prediction model is to use Deep Reinforcement Learning (DRL). It enables learning the suitable behaviour from data. The basic idea of a RL agent planning method is that it observes some state, which represents the vehicle as well as the environment. Actions are selected based on the policy and then applied, after which the agent receives rewards corresponding to action taken. The cumulative reward is used as feedback to tweak the policy. Similar to MDP-based methods, this reward can be designed based on desired policy that needs to be learned. The approximation using deep neural nets (DNNs) avoids the need of explicitly designing prediction models. Reinforcement learning (RL) has confirmed the capability of solving for the optimal policy. It can map various observations to corresponding actions in complicated scenarios [32]. You *et al.* [33] describes one such work where optimal policy learned using RL techniques specifically Q-learning [34]. Qiao *et al.* [32] describes a hierarchically structured RL-based method with two levels similar to the structure of heuristic rule-based methods. The top-level RL structure chooses an option that represents an action subset corresponding to a task. The lower-level RL structure then chooses an action from this subset. A hybrid reward mechanism is also proposed that prioritizes the completion of the current task.

The principle difficulty of these methods is that the agent will only be able to solve the type of situations it encounters during the training process. Hence the quality of data used for training is really important. It should capture as many edges as possible. It is difficult to guarantee functional safety of an RL agent’s decisions due to the black-box nature of DNNs. Using an underlying safety layer that makes sure that the planned trajectory is safe before execution by the vehicle control system, is a means of preventing this problem. Most of these methods are only validated using simulated data and are yet to be implemented and tested in a high-fidelity simulator let alone in a real world scenario.

Predicting the intent of other vehicles is crucial for any behaviour planning paradigm. To solve the problem of uncertainty of intent estimation, Mousavi *et al.* [35] uses a Kalman filter to track obstacles in the surrounding. A superior probabilistic learning-based method uses Gaussian mixtures and Gaussian process regression that are proposed in [36], [37]. These methods are able to anticipate the behaviours of other vehicles and obstacles. This information can then be incorporated into the decision making process. Table 1 gives an overview of the various behavioral planning approaches in the existing literature that was discussed.

Planning Strategy	Strengths	Weakness
Finite State Machines (FSMs)	<ul style="list-style-type: none"> • Easy to implement. • Computationally cheap. • Existing FSM frameworks available 	<ul style="list-style-type: none"> • Failure to generalize. • Depends heavily on heuristics and lacks optimality. • Development requires in-depth knowledge about system and road rules.
Fuzzy-Based Logic	<ul style="list-style-type: none"> • Possibility to combine different behaviours 	<ul style="list-style-type: none"> • Similar drawbacks to FSMs
MDP-based	<ul style="list-style-type: none"> • Better suited for decision making. • Better generalization. • Optimal solution is possible. 	<ul style="list-style-type: none"> • Requires explicit process model definition. • Extent of generalization depends on prediction model. • MDP problem is large and complex in autonomous driving context. • Existing MDP solvers computationally inefficient.
Deep Reinforcement Learning(DRL)	<ul style="list-style-type: none"> • Enables learning from data. • Explicit prediction model not required. • Capable of finding an optimal policy. • Existing RL frameworks can be leveraged. 	<ul style="list-style-type: none"> • Highly data-dependent (data needs to cover all driving situations). • Prone to bias. • Cannot ensure optimal solution every time. • Cannot ensure safety.

Table 1: Comparison of different Behavioural Planning methods.

2.2.2 Tackling Trajectory Planning

Once an optimal behaviour is selected, a trajectory that conforms to the behaviour has to be found; which is termed Trajectory Planning. Trajectory Planning can be further described as the process of real-time planning of the vehicle's transition from one viable state to the next while avoiding obstacles and at the same time satisfying its kinematic limits based on vehicle dynamics and constrained by occupant comfort, lane boundaries and traffic rules [20]. Existing planning algorithms that originate primarily from the field of indoor mobile robotics have been applied with varying degrees of success. Most methods require modification to fit into the paradigm of AD. The vehicle is a non-holonomic system and the environment is highly dynamic and the speeds involved are significantly higher.

Sampling-Based Methods

One way to approach trajectory planning is to sample the state space using rapidly exploring trees(RRTs) or probabilistic sampling (e.g. Probabilistic Road Maps). They work well with high-dimensional spaces. Modification [38] to the original RRT formulations can enable them to handle any vehicle constraints. Khakar *et al.* [39] describes one such approach that is able to guide the AV in performing overtaking maneuvers in complex situations. The main drawback with these methods is that the paths are often jagged and non-optimal which are detrimental to passenger comfort, unless high sampling resolution is used [40]. This leads to high computational complexity. High traffic density and road curvatures also prove to be a Achilles heel of these methods.

Potential Field-Based Methods

The core idea behind potential-field based algorithms is to use obstacles and goals as attractive and repulsive differential fields analogous to electromagnetic or gravitational fields [41]. These forces are utilized to develop the gradient of a potential field. These algorithms compute trajectories along the steepest potential gradient in the resulting field [42]. Khatib [43] describes one such algorithm for indoor navigation and obstacle avoidance. Vector field histograms [44] and elastic band algorithms [45] also fall under the same family. It is certain that the computed path will follow the lowest potential, resulting in collision-free trajectories. The efficiency of the generated potential field depends largely on its safety and accuracy, which is not possible in AD situations. Wolf *et al.* [46] extended this idea for high-speed highway scenarios by conditional buffer zones (that depend on headway time, velocities) around obstacles and, thus, constrained the search space to ensure safety in these dynamic environments. Chipade [47] proposed an overtaking controller utilizing vector field histograms taking into consideration the intent of the LV and ensuring no collisions. One major drawback of these methods is that the computed trajectories have the tendency to get stuck at local minima in these fields [42]. It is also not possible to handle vehicle kinematic constraints, thus, can compute infeasible trajectories that demand unrealistic actuations.

Optimal Control Methods

Optimal control methods try to minimise a performance index (energy, jerk, etc.) or a cost/objective that capture various kinematic and environmental constraints to obtain

an optimal trajectory. A general form of this method for a finite horizon is known as the Constrained Finite-Time Optimal Control (CFTOC) problem [48]. A special case of a general nonlinear optimal control problem without considering constraints is a LQR. This works well for linear systems with quadratic costs. In AD though, the model is nonlinear and costs are almost always represented using non-quadratic functions. It should also be able to handle constraints on input and control. Iterated Linear Quadratic Regulator (ILQR) [49] approach for solving nonlinear and non quadratic equations uses the same process as the LQR solution, but the dynamics and objective function are linearized and quadratized locally and the LQR solution is iterated to increasingly get better approximations of the optimal trajectory of the system. ILQR cannot handle complex constraints which come in the form of state and input for occupant safety and comfort that should be ensured by an AD system. Constrained Iterative Linear Quadratic Regulator (CILQR) proposed by [50], [51] is modification to ILQR to directly handle constraints and ensure the constraints are met for the whole trajectory. Chu *et al.* [52] proposed a different approach that selects an optimal path from a finite number of path candidates that is generated from splines from waypoints that minimizes a cost function based on path safety cost, path smoothness, and path consistency. Optimal control based methods are open loop in nature. Trajectories obtained by such open-loop single stage optimisation do not account for uncertainties in a dynamic environment and, therefore, these trajectory planning methods have limited potential unless used in either extremely controlled or structured environments [20].

Model Predictive Control Methods

MPC [53] also tries to solve the CFTOC problem but in a receding time window, hence MPC is also known as Receding Horizon Control (RHC). This means that an optimal trajectory is found for the whole prediction horizon, but only the first control input from the sequence is applied. The solution is then recomputed after every step unlike in LQR-based control, where there is a single (optimal) solution for the whole time horizon. This can be desirable due to the highly dynamic nature of autonomous driving. MPC has seen widespread adoption in most industrial control problems due to the slower sampling requirements [54]. For application in autonomous driving where MPC should be able to handle (i) nonlinear vehicle dynamics, and (ii) time-varying state and input constraints while navigating in a dynamic environment. MPC-based formulations can easily manage issues in nonlinear multiple-input multiple-output system control, and clearly take both hard and soft system constraints into consideration [55]. This can also lead to high computational complexity hindering real-time execution. Feasibility and uniqueness of the optimisation cannot be guaranteed. This may become an concern of the past as recent developments in highly efficient algorithms for implementing MPC controllers are showing promise [20].

Mousavi [35] proposed a stochastic path planning method to keep track of uncertainties on the environment. The state estimation of EV and obstacles is done using Kalman filters. Linear-Time Varying Model Predictive Control (LTV-MPC) method is used by successive linearization of the system model to generate required optimal trajectories and control inputs. Use of a point-mass system model severely limits the usage of this control architecture for autonomous driving. Murgovski and Sjöberg [56] employs the MPC to develop a predictive cruise control system that is able to overtake a slow-moving LV. The constraints are in the form of mixed-integer lateral limits for guiding the overtake

maneuver.

Molinari [57] proposes a similar but more efficient mixed-integer based formulation by reducing the number of binary variables in the obstacle constraints and also considers the scenario of an oncoming vehicle in the other lane. These methods however do not account for LVs with variable speeds and curvy roads. The model used is also again a point-mass model which is undesirable in real-world application since the planner does not take the non-holonomic system dynamics into consideration. Complex nonlinear system models that better represent the actual systems like e.g. The Kinematic Bicycle Model [58], helps predict more realistic trajectories. Linear MPCs are plagued by approximation error due to local linearization [54] in both system and cost functions. Li *et al.* [59] proposes a Nonlinear Model Predictive Control (NMPC) to dynamically adjust an initial reference trajectory that is predefined using a sigmoid function in accordance with road conditions. The proposed controller only optimizes the lateral movement of the EV. It might be necessary to modify the longitudinal position or velocity to account for any dynamic obstacles which the method does not. However, the method is unique in the way the predicted motion of the dynamic obstacles is accounted for in the optimized trajectory. To summarize, MPCs provides a promising approach for trajectory planning due to its ability to: (i) include system dynamics constraints, (ii) include both hard and soft constraints, and (iii) perform receding horizon control which allows it to plan feasible trajectories over a larger operating range [20].

Interdisciplinary Methods

These approaches assimilate features of several planning methods to take advantage of desirable features of individual methods. For example, Dixit [10],[11] first utilizes artificial potential fields to generate riskmaps that take into consideration the road layout and obstacles. Target points are then selected keeping in mind the aforementioned riskmaps and the reachability of the vehicle which guide the overtaking maneuver. A linear MPC (using reduced, linear system model) generates trajectories that perform overtaking scenarios. The obstacle avoidance constraints are simple lines that divide the whole state sub-space into safe/unsafe regions. The overtaking maneuver is interpreted as a series of consecutive lane changes that is only suitable for structured highway scenarios. This method will not be able to manage situations consisting of multiple vehicles (oncoming or otherwise), and any dynamicity from the LV. The utilization of a simplified, linearised model may lead to computed trajectory having wide discrepancies from the real-world behaviour. Motivated by the results presented in this method [11], in this thesis, the reachability and safety sets combination is also used. However, the method proposed by this thesis significantly enhances the method presented in [11] in multiple manners, mainly, the additional ruled-based maneuver selection block and the modified intermediate target selection. This allows the proposed method to chose the best maneuver for the current situation be it overtaking or lane keeping, and select a intermediate target that is safe and reachable, to which a trajectory planner based on an NMPC can generate feasible trajectories.

Interdisciplinary methods like these, moreover, lack a systematic design procedure and necessitate extensive experimental tests to be proven road worthiness [20]. Table 2 gives an overview of the various trajectory planning approaches in the existing literature that was discussed.

Planning Strategy	Strengths	Weakness
Sampling-based methods (e.g. RRT, PRM)	<ul style="list-style-type: none"> Collision free trajectories. Existing efficient algorithms 	<ul style="list-style-type: none"> Jagged and non-optimal paths Increased computational complexity with higher sampling resolution and complexity No mechanism to handle uncertainties
Potential fields	<ul style="list-style-type: none"> Optimal trajectories guaranteed. Intuitive 	<ul style="list-style-type: none"> Trajectories can get stuck in local minima. Inability to handle system constraints (vehicle kinematics not considered) No mechanism to handle uncertainties
Optimal control (e.g. LQR, CILQR)	<ul style="list-style-type: none"> Ability to handle constraints Optimal trajectories guaranteed Handle uncertainties through stochastic formulations 	<ul style="list-style-type: none"> Open-loop control (no replanning based on state feedback) Requires local linearization resulting in approximation errors. Not suitable for highly nonlinear systems. No explicit handling of hard or soft constraints.
Model Predictive Control (MPC)	<ul style="list-style-type: none"> Explicit handling of hard or soft constraints Suitable for highly nonlinear systems Closed-loop control Receding horizon results in reduction of computational complexity Stochastic formulations similar to optimal control possible 	<ul style="list-style-type: none"> High computation complexity due to high-order system models, nonlinearity, and non-convexity of constraints Feasibility and uniqueness of optimization cannot be guaranteed (NMPC)

Table 2: Comparison of different Trajectory Planning methods.

2.2.3 Combined Behavioural/Trajectory Planning

Some approaches try to tackle both the selection of maneuver and trajectory planning with a single control approach. Pan *et al.* [60] uses a CILQR framework discussed earlier but with an analytic function that adaptively calculates the weights of the cost function based on some heuristics. This paradigm enables the controller to be scenario-aware to encourage different behaviours (overtaking vs lane-keeping) based on current situation. Reinforcement learning is also a possible candidate which can learn a unified policy that selects the optimal control inputs. Liu *et al.* [61] describes one such control strategy where a Long Short Term Memory (LSTM) is utilized to forecast motion of vehicles and then a RL formulation is used to guide the EV to perform overtaking. However, the development of end to end RL methods are still in its infancy.

2.2.4 Discussion

After an exhaustive review of the various methods that contribute to solving the problem of autonomous overtaking, following observations were made.

1. Autonomous overtaking is complex and the environment is constantly changing.
The planning methodology should be able to adapt to it.
2. Proposed methods in the existing literature solve only a subset of challenges.
3. Most methods only consider structured road environments with multiple lanes for easy overtaking.
4. MPCs are best suited for adapting to sudden changes in the environment.
5. However, most MPC-based methods rely solely on the MPC itself to meet all environmental and vehicular constraints.
6. Utilization of linear MPCs for planning is inadequate as they do not scale well to all speeds and maneuvers.
7. MPCs can be effective planners if the number of constraints and optimization horizons are kept small.

Many existing methods discussed above are suitable only for simple scenarios where the LV is static or moving at a constant velocity and the simulated road environment consists of multiple lanes for easy overtaking by executing two simple lane changes. The test scenarios themselves are set up in a way that maneuvers require only a modification of the lateral position to complete the overtake maneuver. Even works that consider dynamically changing environments, is tested in low-fidelity simulations and use LQR based methods that optimize for the whole maneuver; thereby requiring long optimization time horizons and failing to account for deviations of the actual vehicle from the system model. In contrast, this thesis proposes a multi-level planning algorithm that utilizes a rule-based decision making system which enables selection of maneuvers that are safe and efficient. The target selection algorithm is then able to adaptively select an intermediate reference state that is safe and reachable and at the same time helps to keep optimization horizons small. Such intermediate references ensure that the overtaking maneuver demands minimal intrusion onto the adjacent lane, which is crucial for single lane overtaking. An NMPC is further able to plan a trajectory that avoids collisions and adapt to any dynamic changes

in the environment. The NMPC is also able to correct for any deviations of the actual trajectory from the planned trajectory resulting from system model/vehicle mismatch, since state is fed back at every time step.

3 Autonomous Overtaking Method

This chapter describes the proposed pipeline and the various sub-modules that facilitate overtaking. First section provides necessary background information such as conventions followed, and system model used throughout the control architecture. Second section discusses the behavioural planning stage in detail covering creation of risk maps and selection of intermediate reference targets. The final section formulates the NMPC used for trajectory planning.

3.1 Assumptions

The focus of this thesis is on execution of the overtaking maneuver, after a decision to overtake has been made. Thus, in all scenarios that is considered, overtaking is assumed to be the best maneuver possible at that instant. In actual real world situations, this decision is made after careful analysis of the environment. All roads are assumed to be straight infinite lines and, hence, curved roads are not present in any of the tested scenarios. The kinematic model of the system used does not include tire interactions. Hence, the performance of the proposed method is not tested on roads with different coefficients of friction, e.g., slippery roads. The EV state is assumed to be known with completely at all times. It is also assumed that no uncertainty exists in the information regarding the road environment and the other obstacle vehicles (LVs) and is completely known. All tested scenarios are conducted on single lane roads where conventional overtaking (that consists of 2 consecutive lane changes) is not feasible. The competence of the proposed method to adapt to sudden changes in the behaviour of LV (vehicle in front of the EV) will be explored thoroughly. The following sections will go into detail on the proposed pipeline and experimental results of the various scenarios that were tested on.

3.2 Proposed Pipeline

The control architecture is broadly split into two main modules that handle behaviour planning and trajectory planning. The behaviour planning module identifies safe and reachable regions and makes the decision to overtake based on simple heuristics, such as closeness of the LV, its relative speed implemented as a FSM. The environmental information regarding the location of the obstacles and road geometry along with the current EV state is used to select a suitable reference state (target) that best suits both the current active behaviour (overtaking or lane keeping) and high level route plan. The trajectory planning module primarily consists of a NMPC-based trajectory planner. The reference state (target) serves as an input to the NMPC along with any relevant obstacle information. The NMPC then plans the optimal trajectory that moves the vehicle from current state to the desired reference state. The reference state is updated at every control interval to reflect the progress in the execution of the maneuver and any change in the desired behaviour. The dynamic obstacle information is also updated so that the NMPC can modify the path accordingly, to avoid collisions. The current EV state is fed back to the NMPC to achieve continuous dynamic planning and tracking control. Finally, a PID-based tracking controller is used for tracking the optimal control inputs a , δ_f , generated by the NMPC is translated into throttle and brake commands. Experiments were conducted using the Carla AD simulator. The entire planning and control pipeline is shown in Figure 3.

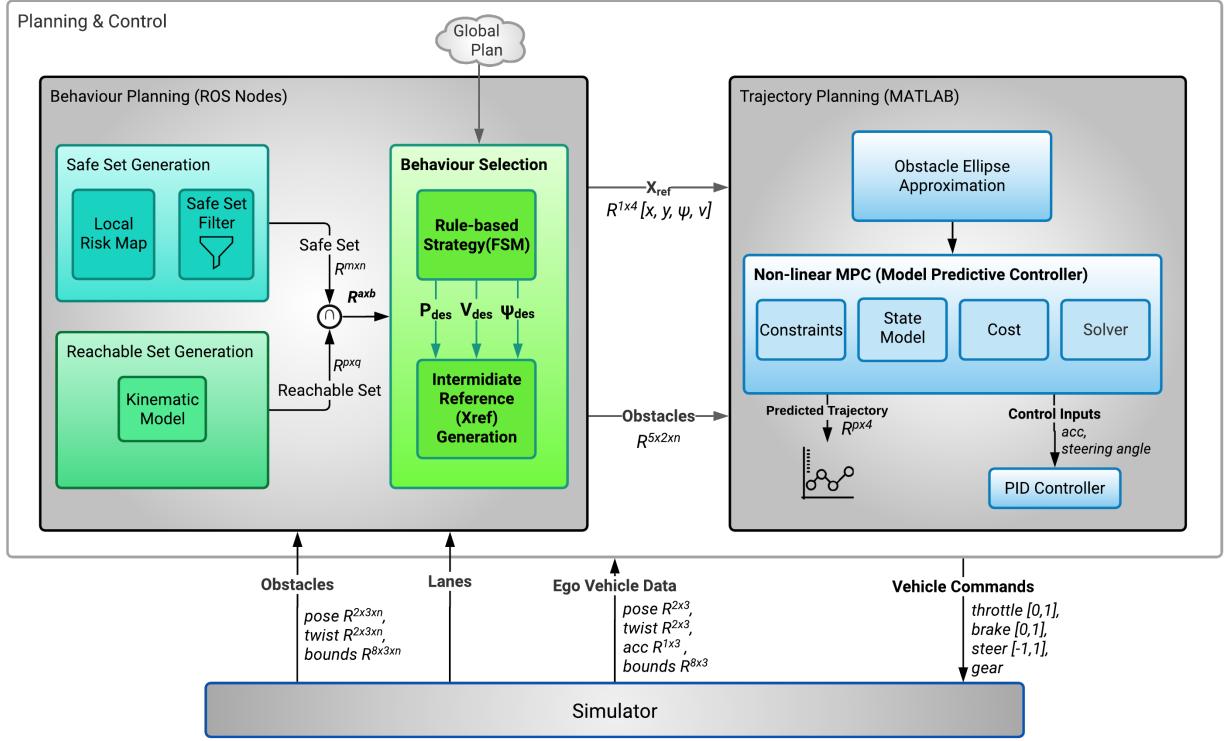


Figure 3: Proposed Pipeline.

3.3 Background Information

3.3.1 Conventions

Figure 4 shows the frame of references that is followed in different modules. The World Frame (W), which is inertial, is located at a fixed point defined by the map used in the simulator. All other frames of reference are derived from the World Frame. The EV Frame (E) and the LV Frame (L) are located at the center mass of the respective vehicles. The simulator (Carla) uses a left-hand cartesian coordinate system (whereas a right-hand cartesian coordinate system is common in AD applications). It is important to note that the Tait-Bryan convention [62] (Roll - Pitch - Yaw) of rotational transformations are to be applied here. Therefore, all information from the simulator is transformed accordingly for uniformity and conformity.

3.3.2 Vehicle Model

The reachability analysis and the NMPC requires an accurate vehicle model. Low dimensional point mass-based models are inadequate as they neglect the fact that the vehicle is non-holonomic due to the 4 degrees of freedom (DOF) that are available for motion versus the 2 DOF that are controllable. Complex dynamic models based on forces that act on the vehicle and also incorporate tire dynamics are available and commonly used [63]. These models normally demand a high computational cost for acceptable real-time performance. For trajectory planning purposes kinematic bicycle models offer the best compromise between performance and accuracy as shown by Kong *et al.* [64]. Hence, a nonlinear bicycle model [64] is used in this work. The left and right side wheel of each

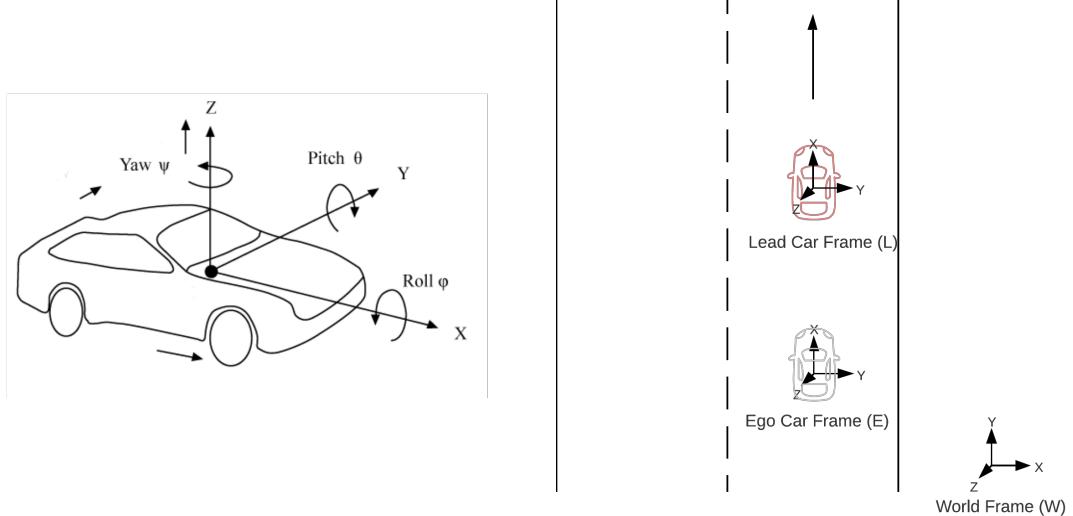


Figure 4: Conventions and Frames of Reference.

axle are simplified to a single wheel at the center of the vehicle as shown in Figure 5. Equations of motion:

$$\dot{x} = v \cos(\psi + \beta) \quad (1)$$

$$\dot{y} = v \sin(\psi + \beta) \quad (2)$$

$$\dot{\psi} = \frac{v}{l_f + l_r} \cos(\beta) \tan(\delta_f) \quad (3)$$

$$\dot{v} = a \quad (4)$$

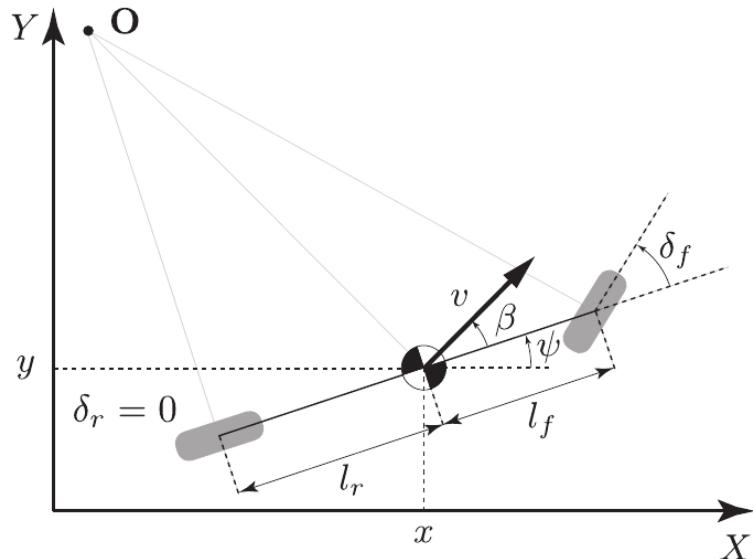


Figure 5: Kinematic Bicycle Vehicle Model.

$$\beta = \tan^{-1} \left(\frac{l_r}{l_f + l_r} \tan(\delta_f) \right) \quad (5)$$

where x and y are the coordinates of the center of mass in an inertial frame (W). ψ is the inertial heading and v is the speed of the vehicle. These 4 variables together define the state of the system. Only 2 system parameters are required for identification (l_r, l_f) which represent the distance from the center of the mass (CoM) of the vehicle to the front and rear axles, respectively. β is the angle of the current velocity of the center of mass with respect to the longitudinal axis of the car. In most vehicles, the rear wheels cannot be steered, thus, δ_r is assumed to be 0. a is the acceleration of the CoM in the same direction as the velocity. Therefore, the control inputs are the front steering angle δ_f , and acceleration a .

3.4 Behaviour Planning

Identifying areas that are valid and safe to drive is a crucial part of AD. Different lane configurations, road boundaries, and other vehicles or obstacles must be considered while driving. In an AD system, data from various sensors like LIDARs, cameras are processed by the perception module to extract information like the location of road boundaries, lane markers, and other obstacles. Since perception processing is out of the scope of this work, it is assumed that this information is already available. It is also assumed that the perception layer can detect and track road features and obstacles in a 20 m radius, which is a reasonable assumption [65]. The maneuver selection is performed by an FSM that is designed to switch between various maneuvers like lane-keeping, overtaking, and lane changing based on heuristic rules. The trajectory planning can be then performed that conforms to the chosen maneuver using the NMPC. It is advantageous to restrict the planning space to safe and reachable areas thus reducing the overall complexity of the NMPC. The approach to identifying these areas is discussed in the following sections. All boundaries and obstacles are represented using polytopes.

3.4.1 Generating Reachable Set

Every vehicle has kinematic and dynamic constraints that limit the configurations in state space that it can reach. This is known as reachability analysis. It provides a powerful tool to assess the capabilities of the vehicle and ensure that the state configuration demanded is within the reach of the system. This reachable space can be computed and represented in various ways. Many tools exist that perform this reachability analysis. CORA [66], SpaceEx [67], Flow* [68] are a few of the methods that were considered initially. They use zonotopes, ellipsoids, Taylor models, or similar set representations to over-approximate the reachable space. However, these toolboxes require a special setup and do not provide a straightforward way to integrate into the pipeline. They also do not provide any means to compute the reachable set online, hence, a simpler quicker way was devised. The reachable set defines all points the vehicle can reach in the entire time horizon T_h (planning horizon). The reachable set $R \subset \mathbb{R}^{m \times n}$ for time T_h can be represented using polytopes that bound the reachable space. Since the position of the EV is more important for trajectory planning and collision avoidance, the reachable set is limited to 2 DOF; x, y , which are longitudinal and lateral positions, respectively. The kinematic model (described in section 3.3.2) is utilized to find the bounds. Using extremes of actuation, namely, front steering angle,

δ_f , and acceleration, a , possible with the vehicle, the system model is simulated for time T_h . The reachability set also excludes regions of space reachable only by exceeding the desired velocity. Desired velocity v_{des} is limited by speed limits and comfort preferences. Boundary of reachable set $R \subset \mathbb{R}^{m \times n}$ is given by

$$\delta_{fmin} < \delta_f < \delta_{fmax} \quad (6)$$

$$v \leq v_{des} \quad (7)$$

$$a_{min} \leq a_x \leq a_{max} \quad (8)$$

Even though these are the bounds for the reachable set, the actual computation was further simplified by just considering the points reachable with the velocity v_{des} without accelerating as every other limit give reachable sets that are a subset of this set. Reverse driving is not considered in this reachability analysis since it is not required for any of the maneuvers that are tackled in this thesis, thus, velocity is always 0 or positive and acceleration in the negative direction refers only to a deceleration in the ego car frame (E). Reverse driving will be considered if this control paradigm is expanded to other maneuvers. An example of the reachable set R generated is shown in Figure 6 represented using a yellow polytope.

3.4.2 Generating Safe Set

It is possible to express the road boundaries and lane markers as constraints in the NMPC during trajectory planning. This approach can lead to a complex control law that may result in increased computational requirements and chances of optimization failure. So, it

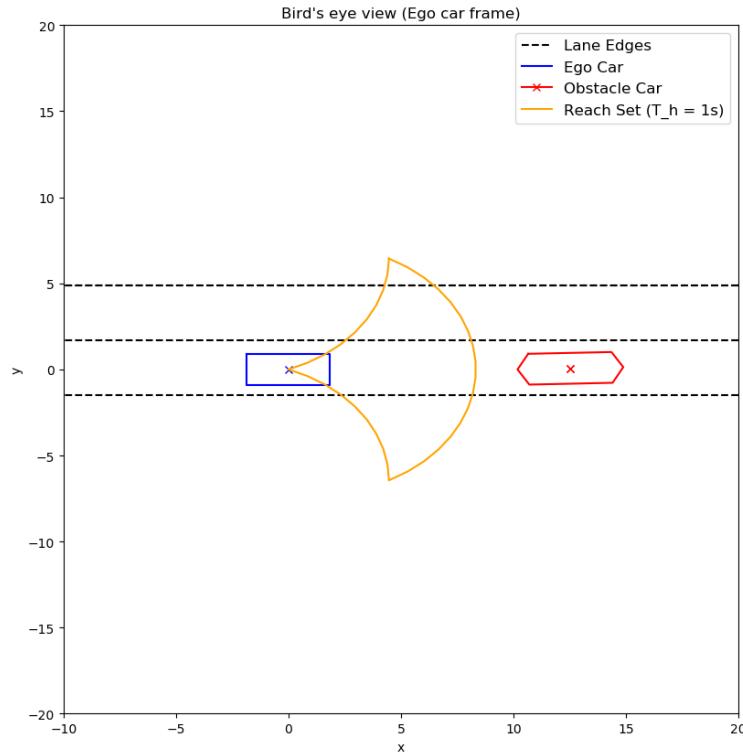


Figure 6: Polytope representing reachable set.

was decided to identify the safe driving zones beforehand and limit the trajectory planning to these regions. Artificial potential fields as described in Wolf *et al.* [46] was adapted for this purpose. It is employed to create a local risk map of the surroundings, which can be used to identify safe driving zones. Every road feature and obstacle is mapped to a potential function that assigns a risk value to all points in space. All roads are assumed to straight infinite lines but these fields can be easily adapted to curved roads. A grid of 20x20 m with a resolution of 0.5 m with the EV at the center is used for mapping these fields. The set of all points that are in this grid is represented by G . All potential fields are calculated with respect to the EV in E frame.

Road Potential Function

The road potential function is given by an exponential function so that the risk value approaches infinity at road edges. This ensures that the road edges and the space beyond is marked as unsafe. The road potential [46] U_{road} is given by:

$$U_{road} = 0.5\eta \exp \left(\frac{1}{\sin(\psi)(x - y_i) + \cos(\psi)(y - y_i)} \right)^2 \quad (9)$$

where η is a scaling factor and y_i is the i^{th} road edge y-coordinate, $i \in 1, 2$. ψ is the heading of the car used to account for the heading of the car with respect to the roads.

Car Potential Function

A repulsive field is generated around all obstacle vehicles to mark them as unsafe areas, thus, enabling the EV to keep a safe distance from them. These fields are based on Yukawa potential [69] which are exponential and approach infinity near the boundaries of the obstacle vehicles. This only embeds the position and orientation of the obstacles in the potential fields. It is beneficial to inflate the obstacle bounds to include velocity information. This can ensure that the EV keeps a safe distance from LVs while overtaking or lane-keeping. This is similar to the approach by Wolf *et al.* [46], though in that case, triangular wedges were used to guide computed path in such a way as to encourage lane change as the EV gets closer to the LV. In the proposed method, the triangular wedges serve as safety margins for the safe set generation. To this effect, the original obstacle bounding box is augmented with triangular wedges in the front and back of the obstacle vehicles. A modified sigmoid function based on the relative and absolute velocities of the obstacle vehicles is proposed to calculate the vertices of the triangles. The use of a sigmoid function limits the size of the triangular wedges at very high velocities.

The position of triangular vertex in front of the vehicle is given by:

$$T_{fvertex} = \left(\frac{D_{max}}{1 + e^{-\gamma(v_{LV}-C)}} \right) \quad (10)$$

It only depends on the velocity of the LV, thus, high velocities of the LV translate to a high safety margin in front of the LV. Thus it ensures EV keeps enough distance in front of LV during the final phase of the overtaking maneuver.

The position of triangular vertex back of the vehicle is given by:

$$T_{bvertex} = \left(\frac{D_{max}}{1 + e^{-\gamma(v_{LV}-C)}} \right) + \left(\frac{D_{max}}{1 + e^{-\gamma(v_{EV}-v_{LV}-C)}} \right) \quad (11)$$

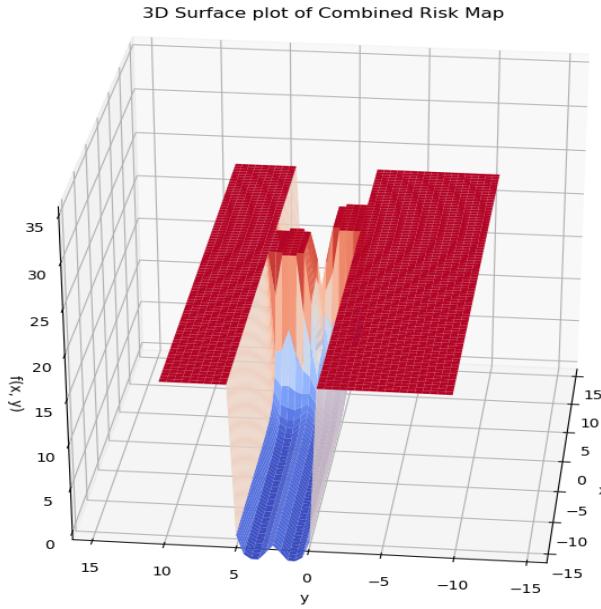


Figure 7: 3D risk map showing the $U(p)$.

where v_{EV} is velocity of the EV and v_{LV} is the velocity of the LVs. The second term that depends on the relative velocity of the EV with LV, is added to expand the safety margin to account for the difference in velocities of the two vehicles. D_{max} is the maximum length of the vertex, thus, limiting the size of the triangle. K_{abs} , γ and C are parameters that determine the scaling of the vertex length with respect to velocities.

After the triangular wedges are added to the LV polygon as described above, the final polygon B is created for each LV. The minimum distance from the EV to this polygon B is given by K_d :

$$K_d = \min_{b \in B} \|p_{EV} - b\|_2 \quad (12)$$

where B is the set of points comprising the boundary of the LV. p_{EV} is the current position of the EV in E frame.

The potential function that exponentially increases as the distance to LVs decrease U_{LVs} is given by the following expression:

$$U_{LVs} = A_{car} \exp\left(\frac{-\alpha K_d}{K_d}\right) \quad (13)$$

where α is a exponential scaling factor that determines the steepness of the potential field towards the boundary of the LV. A_{car} is the amplitude of the Yukawa amplitude. The final combined potential field $U(p)$ (illustrated in Figure 7) is given by:

$$U(p) = U_{road} + U_{LVs} \quad (14)$$

The safe set is then given by all points in G that have total risk values below a certain safe threshold $U_{threshold}$. Safe set S is given by:

$$S = \{p \in G : U(p) \leq U_{threshold}\} \quad (15)$$

This safe set is updated at each time interval so that the risk map reflects the dynamic changes in the environment. Since the shape of the triangular wedges that were used

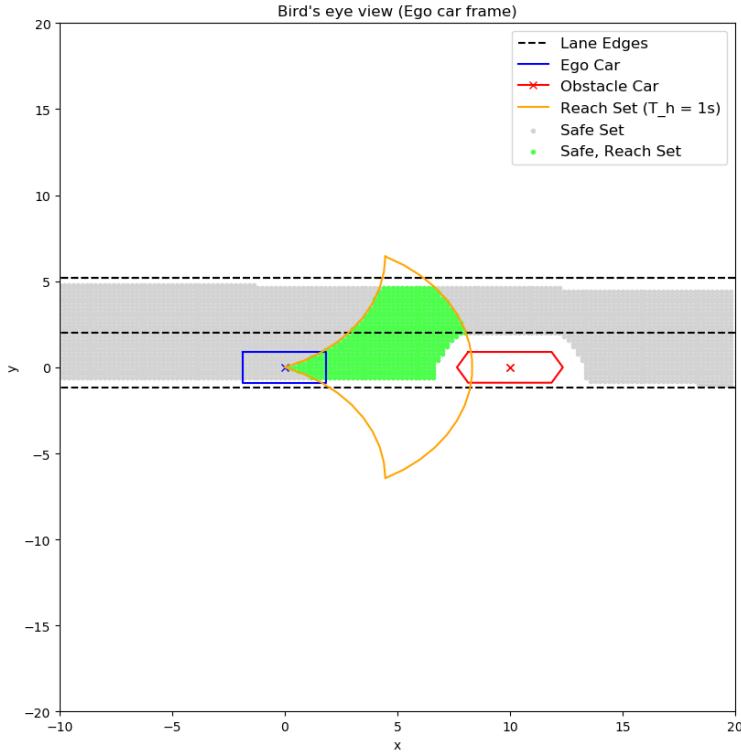


Figure 8: Plot showing the safe and reachable area (shown in green).

augment the LV polygons depends on the relative and absolute velocities of EV, this directly influences the safe set at every time step. The final safe and reachable set S_{SR} is obtained from the intersection of safe set and reachable set.

$$S_{SR} = S \cap R \quad (16)$$

Thus, the safe and reachable set S_{SR} is generated, which is shown in Figure 8.

3.4.3 Behaviour Selection

Rule-based Strategy

After identification of safe and reachable areas, a suitable maneuver has to be chosen by the AD system. The process that goes into deciding the best maneuver is complex and depends on many factors such as the current road layout, applicable traffic rules, and intentions of the passenger. The decision-making process is also affected by the existence of vehicles other than the LV. Since the thesis focuses primarily on a subset of all possible maneuvers, mainly overtaking, the approach was kept simple by using a finite state machine (FSM) for the selection of maneuver. The proposed FSM uses heuristic rules to dictate state transitions. This approach was selected due to its relative ease of implementation and deterministic decision making capability. The FSM has 3 modes:

1. Overtaking
2. Lane Changing
3. Lane Keeping.

FSM selects the desired longitudinal and lateral positions $p_{des} = [x_{des}, y_{des}]$, which is the end target on completion of the maneuver.

Overtaking Mode

When the EV detects a vehicle in front, overtaking mode is selected if overtaking is allowed. The desired position is selected as the point at the center of the lane, $d_{overtake}$ distance in front of the LV (L frame) to facilitate overtaking. The position is calculated in the LV frame (L) which eliminates the necessity of updating it. This also ensures that the EV keeps certain safe distance from the previous LV when the overtaking maneuver is complete.

Lane-Keeping Mode

When the overtaking maneuver is complete and the EV is at safe distance in front of the current LV, the behaviour selection logic switches to lane-keeping mode. This is achieved by selecting a desired position p_{des} that is distance d_{cruise} away from the current position of the EV (p_{EV}).

Lane Changing

In certain situations, overtaking may not be permitted by the active road rules and traffic conventions. In these situations, a lane change may be possible if EV is travelling on a multi-lane road and there are no other vehicles in the adjacent lane. Consequently, as the EV approaches the LV, p_{des} is chosen as a point in the center of the adjacent lane.

The above rule-based strategy is illustrated as a flow chart in Figure 9. This module also decides on desired heading ψ_{des} and desired velocity v_{des} based on the behaviour. ψ_{des} is selected such that the final heading after maneuver aligns with the curvature of the road. The vehicle should not violate any speed limits and or cause any sort of discomfort to passengers due to excessive speeds. Thus, v_{des} is decided based on active speed limits and passenger comfort. The p_{des} chosen above is currently in the LV frame (L) and is transformed to the EV frame (E) before generating the final desired state X_{des} .

$$X_{des} = [x_{des}, y_{des}, \psi_{des}, v_{des}] \quad (17)$$

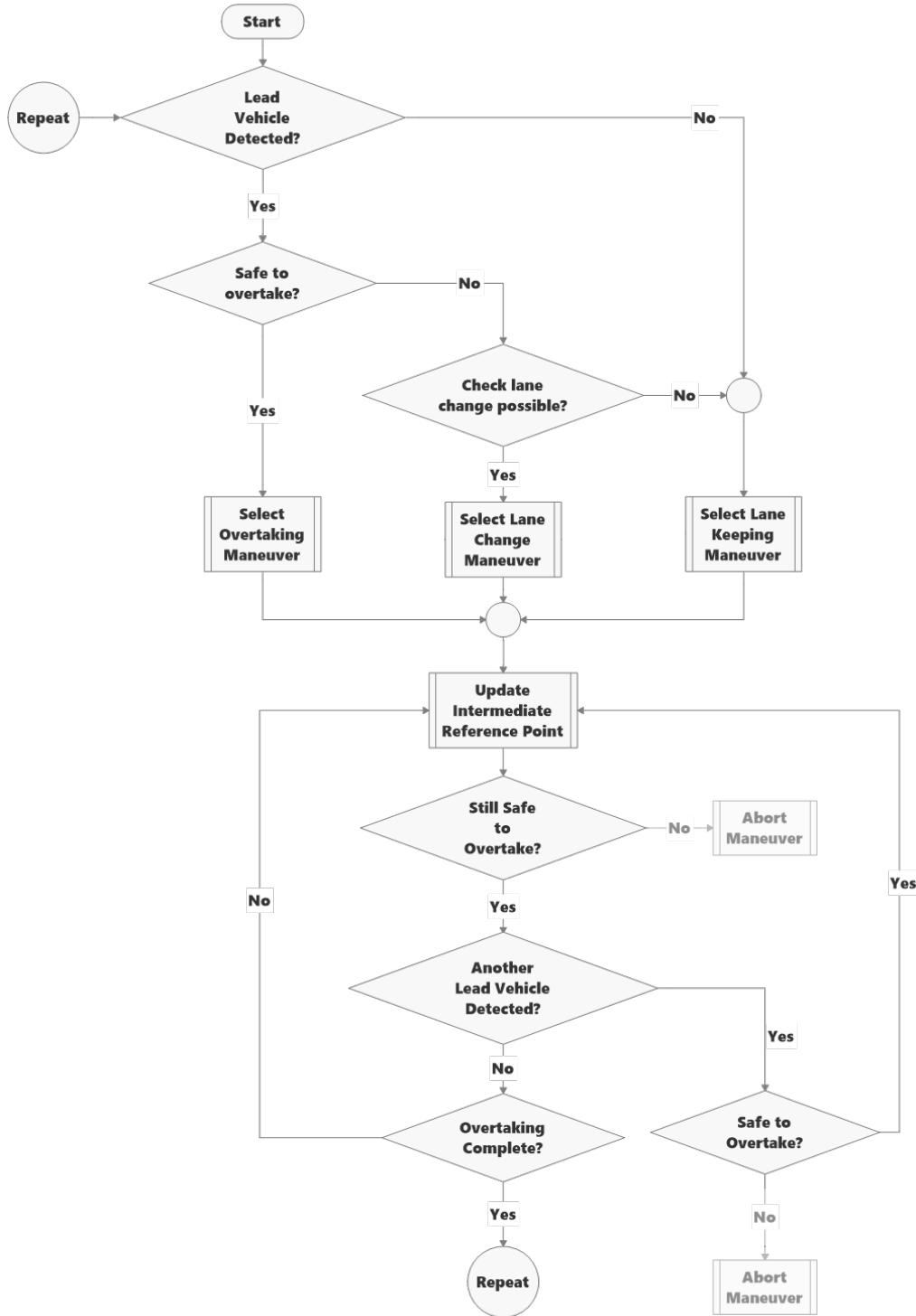


Figure 9: Flow chart illustrating the rule-based logic of the FSM.¹

¹Remark: All the possible scenarios and transitions, e.g. Aborting Maneuver, were not considered in the final rule-based strategy that was implemented due to lack of time and for the sake of simplicity

Intermediate Reference Point Generation

This is a crucial step in the control process where a suitable intermediate state X_{ref} is selected from the safe and reachable set S_{SR} described in the previous section, which conforms with the maneuver chosen by the behaviour selection FSM. This approach is similar to one taken by Dixit *et al.* [10], [11], though, it has been modified to improve the selection of intermediate reference point for single lane overtaking. The modifications allow for minimal intrusion onto adjacent lanes while overtaking. In practice, this is achieved by selecting a point p_{ref} that is in the reachable and safe set S_{SR} and minimizes the distance to the final desired position p_{des} .

$$p_{ref} = \underset{p \in S_{SR}}{\operatorname{argmin}} (\|p - p_{des}\|_2) \quad (18)$$

This is repeated at every time step, after which p_{ref} moves closer to final target p_{des} until it coincides with it. The intermediate reference point selection process remains the same regardless of the maneuver selected. This iterative process is able to guide the NMPC to perform the required maneuver, consequently, reducing the complexity in the design of the NMPC. This also captures any sudden changes in the environment or in the state of the LV, thus, ensuring that the X_{ref} is always safe (lies in the safe and reachable set S_{SR}). The overall behaviour selection and intermediate target selection is illustrated in the Figure 10. Thus, the intermediate target selection ensures that:

- The NMPC always receives a reference target state that is safe and reachable.
- The length of trajectories planned by the NMPC are kept short, thus, requiring a short prediction horizon.

The p_{ref} was computed from the safe and reachable set and is in the EV frame (E). Since, the next stage of planning (NMPC Trajectory Planning) operates in the world frame (W), necessary transformations are applied beforehand. Finally, the intermediate reference target X_{ref} provided as input reference to the NMPC is:

$$X_{ref} = [p_{ref}, \psi_{des}, v_{des}] = [x_{ref}, y_{ref}, \psi_{des}, v_{des}] \quad (19)$$

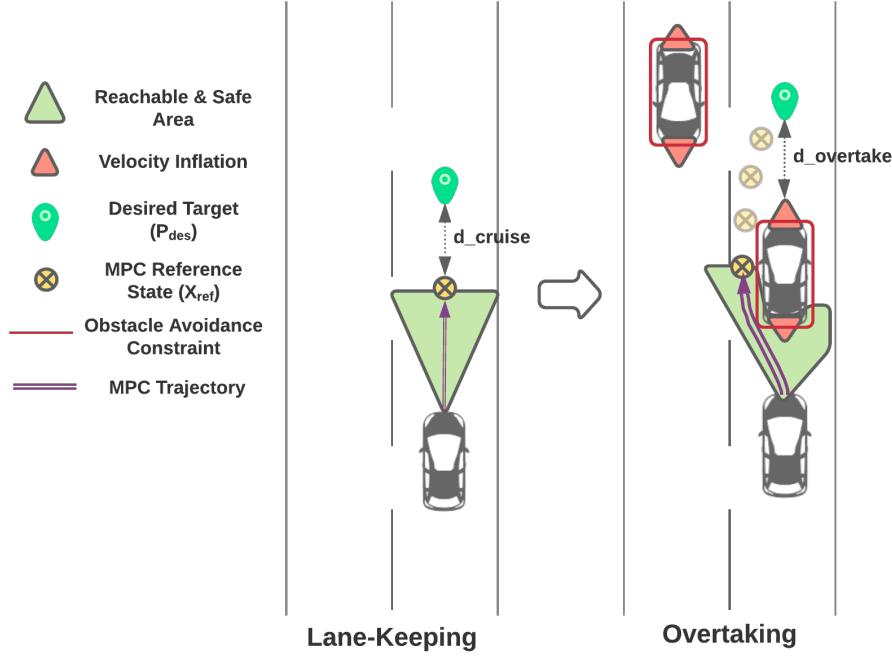


Figure 10: Iterative selection of reference point.

3.5 Trajectory Planning

The next phase of planning involves generating a trajectory that follows the system dynamics, devoid of collisions, obeys state and control limits. Since this problem can be easily formulated as an optimal control problem (CFTOC), an NMPC can be used for solving it. An NMPC plans an optimal trajectory from the current state $x^k = [x_0, y_0, \psi_0, v_0]$ to the reference state $X_{ref} = [x_{ref}, y_{ref}, \psi_{des}, v_{des}]$ and generates necessary optimal control inputs to the system in form of acceleration a , and front steering angle δ_f . To review, a discrete CFTOC problem [70] is:

$$\min_u J = \sum_{k=0}^{N-1} l_0^k(x^k, u^k) + L^N(x^N) \quad (20)$$

where $N = T_h$ is the planning horizon, l_0^k is the running cost and L^N is the terminal cost. x^k is the state vector and u^k control input to the system, defined by system model at time step k .

This is subject to system model equation,

$$x^{k+1} = f^k(x^k, u^k), \quad (21)$$

and input constraints, state constraints and other custom constraints

$$\begin{aligned} u_{min} \leq u^k \leq u_{max}, & \quad \text{input constraints} \\ x_{min} \leq x^k \leq x_{max}, & \quad \text{state constraints} \\ g^k(x^k, u^k) \geq 0 & \quad \text{other constraints} \end{aligned} \quad (22)$$

For trajectory planning, the system model $f^k(x, u)$ is the nonlinear kinematic bicycle model discussed in the previous section. Input constraints are limits on the control

inputs acceleration a , and front steering angle δ_f . This avoids control actions that are unreasonable and impossible due to physical system constraints. State constraints are hard limits that restrict the state space. $g^k(x, u)$ constitutes a family of nonlinear functions that can express various constraints as functions of both state and control inputs. This is later used for specifying collision avoidance constraints. In a NMPC, the above CFTOC problem is solved at each time step as a receding horizon optimal closed loop control problem. Since the current state is fed back and replanned at every time step, NMPC is able to compensate for model/plant mismatch and adapt to dynamic changes in the environment. A suitable solver is used for solving the optimization problem $J(u, x)$ at every time step, e.g. sequential quadratic programming(SQP) or interior point solver [70].

3.5.1 Obstacle Avoidance Constraints

Even though the reference state X_{ref} is safe, it is necessary to ensure that the trajectory planned by the NMPC is also safe and free from collisions. To achieve this, the obstacle avoidance is expressed as state constraints in the form of constraint function $g^k(x, u)$. For example, in Mousavi *et al.* [35], the obstacle avoidance constraint is a set of simple lines that divide the space into safe/unsafe regions. The intersection of these individual regions give the final safe space for vehicle trajectory. Hence, the constraints are linear. The disadvantage is that the unsafe region is not restricted to the obstacle and requires updating at every time step. In [51], the collision avoidance is expressed as the minimum distance from the polygon that encompasses the obstacle vehicle. This gives the best representation of the obstacle and the safe region is only devoid of the obstacle polygons. The drawback is that this function is neither continuous nor easily differentiable. Many CFTOC solvers like SQP require that the constraint functions be continuous and twice differentiable. To achieve both, the following approach is adopted in this thesis. The obstacles are represented using higher order ellipses. The equation for a higher-order general ellipse is given by

$$\left(\frac{(x - x_e) \cos(\phi) - (y - y_e) \sin(\phi)}{a} \right)^n + \left(\frac{(x - x_e) \sin(\phi) - (y - y_e) \cos(\phi)}{b} \right)^n - 1 = 0 \quad (23)$$

where a is the sub axis (radius) of the X axis of the ellipse given by $a = f \cdot \frac{(L_o + L_e)}{2\alpha} \alpha$ and b is the sub axis (radius) of the Y axis of the ellipse given by $b = f \cdot \frac{(W_o + W_e)}{2\alpha}$. L_o is length of the obstacle vehicle, L_e is length of the EV. n is the order of the ellipse, α is the inflation factor, W_o is width of the obstacle vehicle, W_e is width of the EV. x_e is the center at the X axis of the obstacle ellipse given by x -coordinate of the center of mass of the vehicle. y_e is the center at the Y axis of the obstacle ellipse given by y -coordinate of the center of mass of the vehicle. ϕ - orientation in radians of the ellipse given by the heading of the obstacle vehicle.

Higher-order ellipses are used because they represent the vehicles, which are usually rectangular in shape more accurately, as shown in Figure 11. The parameters of the ellipse encompassing the rectangular obstacle vehicle are calculated using the length and breadth of the obstacle rectangle. The ellipse is also padded with the length and breadth of the EV, and an inflation factor α to ensure that collisions do not occur during the entirety of the maneuver. It is important to note that the safety constraints formulated for the NMPC are different from that which was considered for behaviour selection and intermediate reference point selection. In the latter, velocity inflation margins are used to augment the obstacle footprint while generating the safe set. However, in the purview of the NMPC,

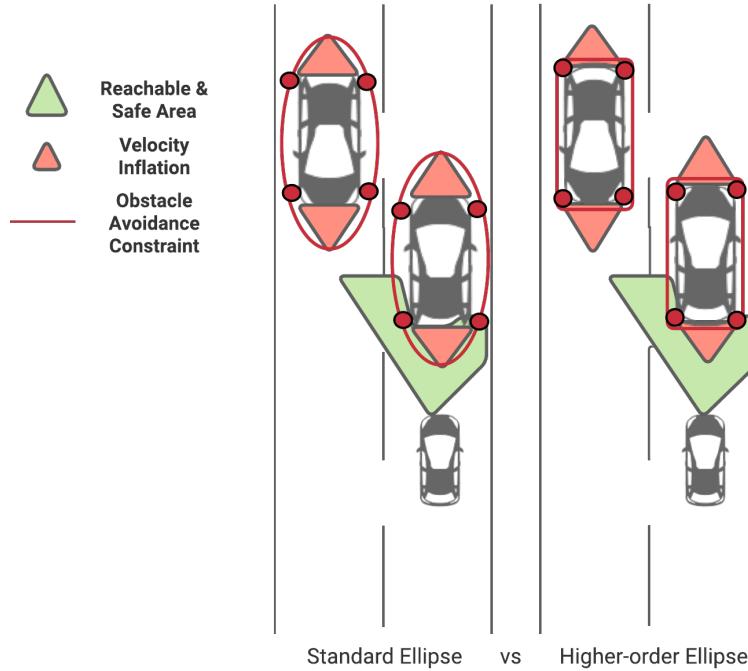


Figure 11: Comparison between using standard ellipses vs higher-order ellipses for LV obstacle avoidance constraint.

this was avoided, instead only using the bounds of the obstacle vehicle to form the safety constraint as discussed above. This was done to simplify the approximation of an obstacle using ellipses. The inclusion of triangular wedges that embedded the velocity information will lead to unnecessary restrictions and reduce the available space for trajectory planning. The velocity inflation is artificial and will not actually cause collisions if violated. Moreover, intermediate reference points are anyway chosen keeping these safety margins in mind. The future trajectory of the LV is not currently considered in the obstacle avoidance constraints. Instead, the NMPC will dynamically change the trajectory to account for the motion of the LVs at each time step.

Thus, each obstacle is represented using ellipses as discussed above and the constraint is formulated as below:

$$g_i^k(x) = \left(\frac{(x - x_{e,i}) \cos(\phi_i) - (y - y_{e,i}) \sin(\phi_i)}{a_i} \right)^n + \left(\frac{(x - x_{e,i}) \sin(\phi_i) - (y - y_{e,i}) \cos(\phi_i)}{b_i} \right)^n - 1 \leq 0 \quad (24)$$

where i indicates the index of the obstacle, $i = 1, 2, 3 \dots N$, $N = \text{Number of obstacles under purview of the NMPC}$.

3.5.2 Nonlinear Model Predictive Controller Trajectory Planner

The following section describes in detail the specifics of the NMPC formulation and the control loop that computes the optimal trajectory and control inputs u_k at every time step k (Illustrated in Figure 12). The NMPC planning is carried out in the world frame (W). Therefore, all inputs to the NMPC such as reference target X_{ref} , state feedback, obstacle polygons are transformed to the world frame (W).

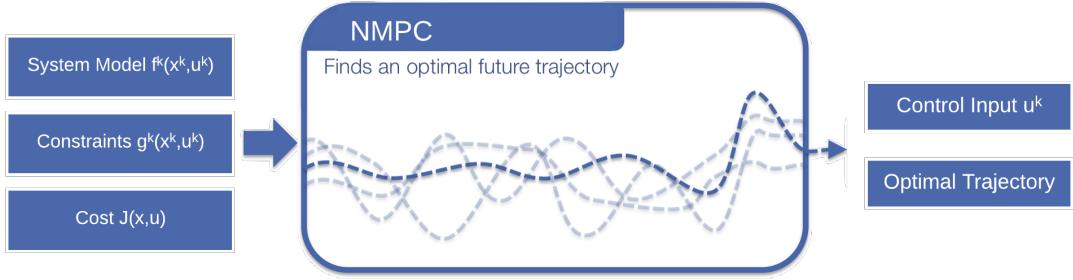


Figure 12: Overview of a MPC/NMPC.

System Model

As stated earlier, the system model used is the nonlinear kinematic bicycle model discussed in section 3.3.2. Since the proposed pipeline is tested in simulations, it is assumed that the system is completely observable and no uncertainty exists in the state feedback. Thus, no measurement model was considered. In practical applications, though, it is necessary to have a measurement model as all states may not be observable and might have plant and measurement noise that should be accounted for. A state estimator such as Kalman filter can be used for this purpose [71].

Constraints

Apart from the obstacle avoidance constraints, control inputs are also constrained. This is in order to limit the acceleration and steering angle inputs to what is capable by the vehicle. The acceleration in the positive direction is limited by power of the engine, mass of the vehicle, given by a_{max} . The acceleration in the negative direction is limited by the braking capabilities of the vehicle, given by a_{min} . The steering angle is also limited to δ_{fmin} and δ_{fmax} .

Cost

The cost function J_k for every control step k is a standard quadratic cost with weights w_x for state variables is shown below:

$$J_k = \sum_{i=0}^{N-1} w_x \|r_{k,i|k} - x_{k,i|k}\|_2 + \sum_{i=0}^{N-1} w_u \|\Delta u_k\|_2 \quad (25)$$

where $N = T_h$ is the planning horizon, $r_{k,i|k}$ is the reference value for i prediction horizon step, $x_{k,i|k}$ is the predicted state vector at i prediction horizon step, Δu_k is the change in the control inputs from $i-1$ to i prediction horizon steps.

w_x penalise deviation from state reference X_{ref} and w_u penalise sudden, large changes in the control inputs. During the overtaking maneuver, it is crucial to keep the lateral position y , in check where as heading ψ and velocity v can be more flexible. Therefore, the weights w_x are chosen to facilitate this. The steering angle and acceleration cannot instantaneously change in a real vehicle due to the physical constraints. Hence, appropriate penalization on u^k are also applied for discouraging this behaviour. The numerical values of the weights along with other simulation parameters can be found in Appendix A

Solver

A general purpose SQP(Sequential quadratic programming) [70] solver is used for solving the CFTOC problem (20 - 22). Many implementations of this algorithm are available e.g. MATLAB Optimization Toolbox [72], Python - SciPy [73] etc. The overall planning algorithm from behaviour planning to the trajectory planning stage is described in Algorithm 1.

Algorithm 1: Planning Algorithm.

```

initialize:
 $k \leftarrow 0$ 
 $u^{k-1} \leftarrow u^0$ 
 $N \leftarrow predictionHorizon$ 
 $w_k, w_u \leftarrow initializeWeights$ 
while  $TRUE$  do
    begin Behaviour Planning Phase
         $U \leftarrow updateRiskMap()$ 
         $R \leftarrow generateReachableSet(N)$ 
         $S \leftarrow generateSafeSet(U, U_{threshold})$ 
         $S_{SR} = S \cap R$ 
         $x_{des} \leftarrow selectBehaviour()$ 
         $x_{ref} \leftarrow getIntermediateReference(x_{des}, S_{SR})$ 
    begin Trajectory Planning Phase
         $x^k \leftarrow getCurrentState()$ 
         $g_i^k \leftarrow getCollisionAvoidanceConstraints()$ 
         $u^k \leftarrow solveNMPCstep(x^k, u^{k-1}, x_{ref}, N)$ 
         $applyOptimalControlInput(u^k)$ 
     $k \leftarrow k + 1$ 

```

4 Experimental Design

This chapter explains in detail the development process that went into setting up the proposed pipeline and the simulation environment. Various simulated scenarios used to test the proposed solution are also discussed.

4.1 Implementation Details

The proposed pipeline, except for the NMPC, was developed in Python. The NMPC was developed using the Model Predictive Control Toolbox [72] from Mathworks. The Robotic Operating System (ROS) [74] framework was used to integrate and facilitate the inter-module communication. The Carla simulation engine [75] was used to create test scenarios and simulate the vehicle physics. Some background information regarding these tools is given below.

4.1.1 Robotic Operating System

ROS is an open-source middleware software framework that serves as a platform for different software components to communicate with each other. These components spawn processes, known as nodes, that communicate through topics and services using ROS messages which are defined data structures that carry different data or message types. Message transfer between these nodes follow a publisher-subscriber system. Nodes that want to send messages publish it to a topic and the nodes that wish to receive these messages subscribe to the respective topic. A node can also advertise services. Upon invoking a service request, the node performs certain actions at the end of which it returns a result similar to client-server paradigm. The ROS master, which is the main node, keeps track of all other registered nodes and facilitates information exchange between them.

4.1.2 CARLA Simulation Platform

Several simulation platforms such as CarSim, LGSLV, IPG Carmaker. were considered for testing the proposed method. Ultimately, CARLA (Figure 13), a simulator for AD research, was selected since it has a strong community, is open source as well as having built-in ROS integration. It provides state-of-the-art rendering quality, realistic physics and a basic NPC logic. CARLA is designed as a server-client system where the server runs the simulation and renders the scene, and the clients connect to the server and requests for information or changes in the simulation via TCP/IP socket communication. The client sends commands and meta-commands to the server and receives sensor readings in return. A Python API is provided at the client side, which can be utilized to develop code that controls the simulation, and administers commands to control the EV and other non-player characters e.g. LVs.



Figure 13: Carla Simulator.

4.1.3 Model Predictive Control Toolbox

The trajectory planner was formulated as NMPC as discussed in Section 3. Thus, a suitable model predictive control library was required for implementation. Some existing Python libraries such as SciPy and Do-mpc were considered initially as viable candidates. SciPy offers SQP optimization but lacks the framework for an easy application of MPC paradigm in control applications. Do-mpc offers an open-source toolbox in robust model predictive controls and supports optimization of nonlinear systems. Ultimately, the Model Predictive Toolbox from Mathworks was chosen since it offers an easy integration with Simulink, thereby enabling GUI-based development and has a more comprehensive documentation. Integration of the trajectory planner developed using this toolbox with the overall pipeline was straightforward since MATLAB offers ROS support. Though there was an option to specify the analytical Jacobians of the system model and constraint for better performance of the SQP solver, initial attempts to implement it failed and later abandoned due to lack of time. The SQP solver was eventually configured to compute the Jacobians numerically.

4.1.4 Pipeline Development

As mentioned earlier, the components were developed in Python and integrated as ROS packages. The implemented pipeline and their inter-component communication is shown in Figure 14. The data comprising of vehicle and environment state were queried from the simulator using the CARLA Python API. Information regarding road boundaries and

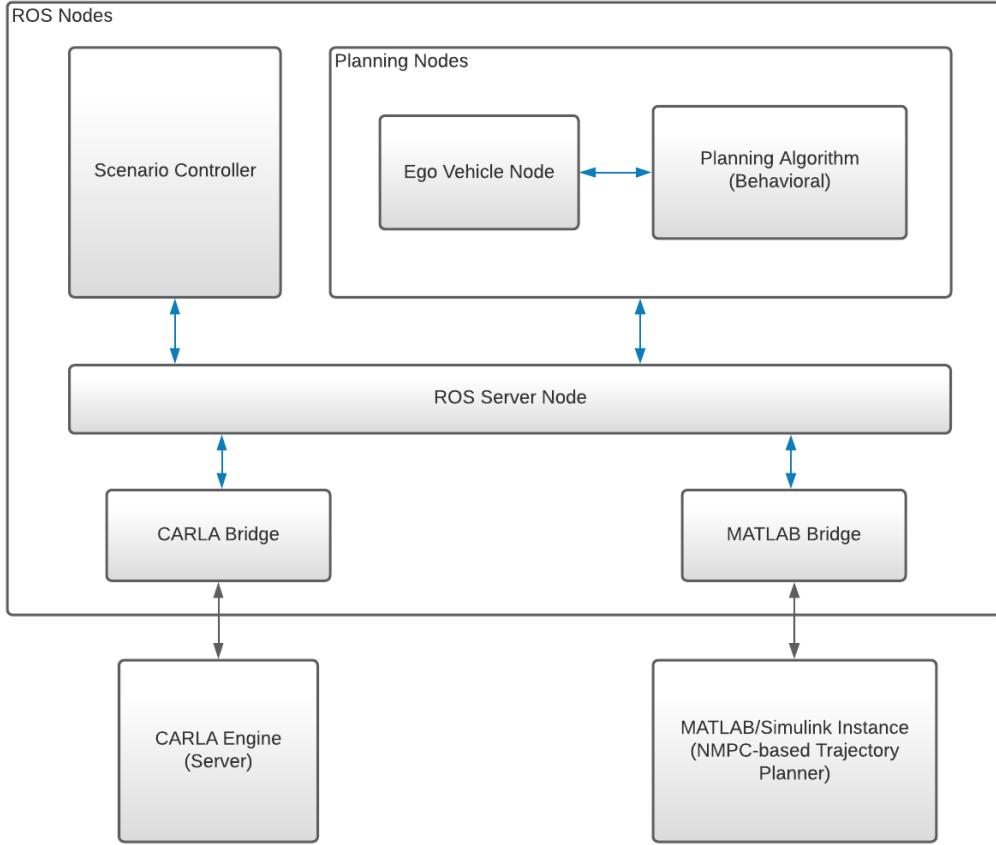


Figure 14: Block diagram showing the overall simulation setup.

lanes was not directly available from the simulator, therefore, an algorithm was developed to generate it using the provided CARLA API. In a real AD stack, this information is be available from the Perception Layer. The state information of other vehicles and obstacles was also directly available by querying the simulator. This information was used to generate the safe set and the reachable set. Shapely [76], a manipulation and analysis of planar geometric objects library, was extensively used to represent the vehicle bounding rectangles and boundaries of the safe and reachable set. It was also used to analyse and filter out non-drivable regions. The generation of reachable sets required solving of differential equations with different initial conditions. This was done with the help of the SciPy library [73]. The intermediate reference state selected along with EV state and obstacle information was published as ROS messages in the next step of planning.

The NMPC was developed as a Simulink model in MATLAB. MATLAB supports ROS integration and is able to subscribe or publish ROS messages through the ROS Toolbox, that enables Simulink models to interface with ROS server. The toolbox comes with built-in support for common ROS message types and also supports addition of custom messages types if required. These features were necessary to subscribe and publish messages to CARLA ROS bridge. The simulink model subscribes to these messages so it is able to plan the optimal trajectory and compute corresponding control commands. It then publishes these final commands that control the EV in the simulation. Since the NMPC generates control commands in the form of acceleration a and front steering wheel angle values δ_f , they require translation to throttle, brake and steering wheel values. Acceleration can be positive, indicating that throttle has to be adjusted, or negative,

indicating brakes have to be applied. This was achieved using a bank of PID controllers that map positive acceleration values to throttle command that varies from [0, 1] and negative acceleration values to brake command that varies from [0, 1]. The PID was also tuned to achieve optimal response. These were the commands that were finally published by the model. The steering angle values δ_f was mapped to CARLA steering commands which range from $[-1, 1]$ using simple scaling and output saturation. To ensure that the simulation and MATLAB NMPC planner runs in step, the CARLA simulator was configured to run with fixed time-step and in synchronous mode.

4.2 Test Scenarios

Various driving scenarios were developed in the CARLA simulator to test the proposed pipeline. The aim of these test scenarios was to evaluate the proposed pipeline in terms of its capability to control the vehicle optimally in multiple overtaking and lane keeping situations. The scenarios were developed keeping in mind the complex situations that can arise in a real world driving situation. Different vehicle types such as motorcycles, bicycles, and cars with different dimensions were used as LVs in these scenarios to evaluate the ability of the planning algorithm to adapt.

The test scenarios were broadly classified to scenarios where overtaking was allowed and where it was not.

Scenarios with No Overtaking

These scenarios focus on driving situations where the EV is supposed to keep to the center of the current lane and maintain a constant speed. Overtaking is not allowed under any circumstance. This is done to simulate road conditions/rules that do not allow overtaking, for example, single lane roads with continuous lane markings, overtaking forbidden by law. The vehicle can either change lanes in case of a multi-lane road or follow the LV keeping safe distance.

- Scenario 1 - Simple Lane keeping - In this scenario, the EV which is at rest is supposed to accelerate and eventually maintain the desired velocity v_{des} , at the time, keep to the center of the lane adjusting for any slight curvature of the road.
- Scenario 2 - Lane Change, Static Vehicle - In this scenario, overtaking is forbidden but lane change is allowed, if safe. It simulates multi-lane roads where lane change can be a viable option. The obstacle is static and represents malfunctioning vehicle or any other obstruction. The EV is expected to change lanes if it is safe. After the lane change, the EV is expected to switch to lane keeping behaviour while maintaining v_{des} .
- Scenario 3 - Lane Change, Moving Vehicle - This scenario is similar to the previous one except the obstacle in front is moving. Overtaking is not allowed and since the desired velocity v_{des} of the EV is higher than that of LV $v_{obstacle}$, it has to execute a lane change. After the lane change, the EV is expected to switch to lane keeping behaviour while maintaining v_{des} .

Scenarios with Overtaking

In these test scenarios, the overtaking is allowed. All scenarios are on a single lane roads with dashed lane markings. Thus, the EV is allowed to overtake the LV or obstacle with minimal intrusion on to the opposite lane. LVs can be static, moving at fixed speed or even have dynamic speeds. Overtaking is allowed and encouraged in the following subset of scenarios.

- Scenario 4 - Static Vehicle - In this scenario, the EV is expected to overtake the static LV in front of it, all the while avoiding collisions. After the overtaking maneuver, the LV is supposed to resume lane-keeping behaviour.
- Scenario 5 - Small Static Vehicle - This scenario is set up in a similar manner to the previous one except for the size of the LV. It tests overtaking scenarios where the obstacle might be a small vehicle like a bicycle or motorbike. The planning algorithm must adapt and plan a trajectory that does not significantly encroach on to the other lane, thereby, increasing the risk of head-on collisions with oncoming traffic.
- Scenario 6 - Moving Vehicle - This scenario is similar to scenario 6, except the LV is moving at constant velocity. Since the $v_{des} = 8.33m/s$ is higher than the $v_{lead} = 3m/s$, the EV should overtake the LV and then resume lane keeping mode.
- Scenario 7 - 2 Vehicles - In some situations, the EV might not be aware of a second obstacle vehicle in front of the LV. This may be due to perception limitations and sensor obstruction by the LV. Thus, the planning algorithm has to adapt to the existence of a second obstacle in the middle of the overtaking maneuver. Also, EV should be able to seamlessly switch between lane keeping and overtaking mode, and, thus, overtake vehicles as and when they are encountered. The LVs can potentially occupy any position on the road, e.g., parked haphazardly on the curb. These situations are tested in the following scenarios. The EV is supposed to overtake both vehicles while avoiding any collisions.

Each scenario was also repeated 3 times using the same parameters to reduce the probability of a chance success. The following factors were used to evaluate the performance of the proposed method in each test scenario.

- Success in selection of optimal maneuver
- Number of collision events
- Adherence to road layouts
- Intrusion onto the adjacent lane during overtake
- Adaptability to change
- Violation of state and control input constraints
- Quality of overall trajectory w.r.t. unnecessary heading and speed changes
- Sudden changes in acceleration and steering inputs

Each simulation was closely observed during execution to detect any collisions. The data consisting of positions, velocities, headings, control inputs of the EV, and positions, velocities of LVs over the course of each scenario simulation was gathered after each simulation and graphically plotted for further analysis.

The simulations were carried out on a laptop configured with Core i7 (8th gen) processor with 16GB RAM and a Nvidia Geforce 1050 Ti (4 GB) graphics card running both the ROS server and MATLAB instance. The results of the various test scenarios are discussed in the following section. The various parameters used for the simulations like the NMPC weights, vehicle parameters etc. are given in the Appendix for reference. The prediction horizon of the NMPC, given by $T_{horizon}/\Delta t_s$, is 20.

5 Results

In all figures, the desired reference point p_{des} selected by the rule-based strategy is shown in green. The intermediate reference point X_{ref} finally selected by the behaviour selection module (BSM) is shown in yellow. The EV bounds are represented using blue polygons where as obstacles are represented using red shaded polygons. The obstacle vehicle in front of EV is referred to as the LV. It is to be noted that plots that show the trajectories (Top view of the maneuver) has different x and y scaling so as to fit the entire evolution of the EV trajectory in one plot. There is a delay (1s) between initial application of acceleration to point where the velocity starts to increase in every scenario. The acceleration commands are mapped to the engine throttle command using a PID, as discussed in the Section 4. Thus, the inertia and frictional forces of the simulated EV has to be overcome before the EV can start moving. Also, the PID introduces certain amount of lag, contributing to the delay.

5.1 Scenarios with No Overtaking

Scenario 1 - Simple Lane keeping

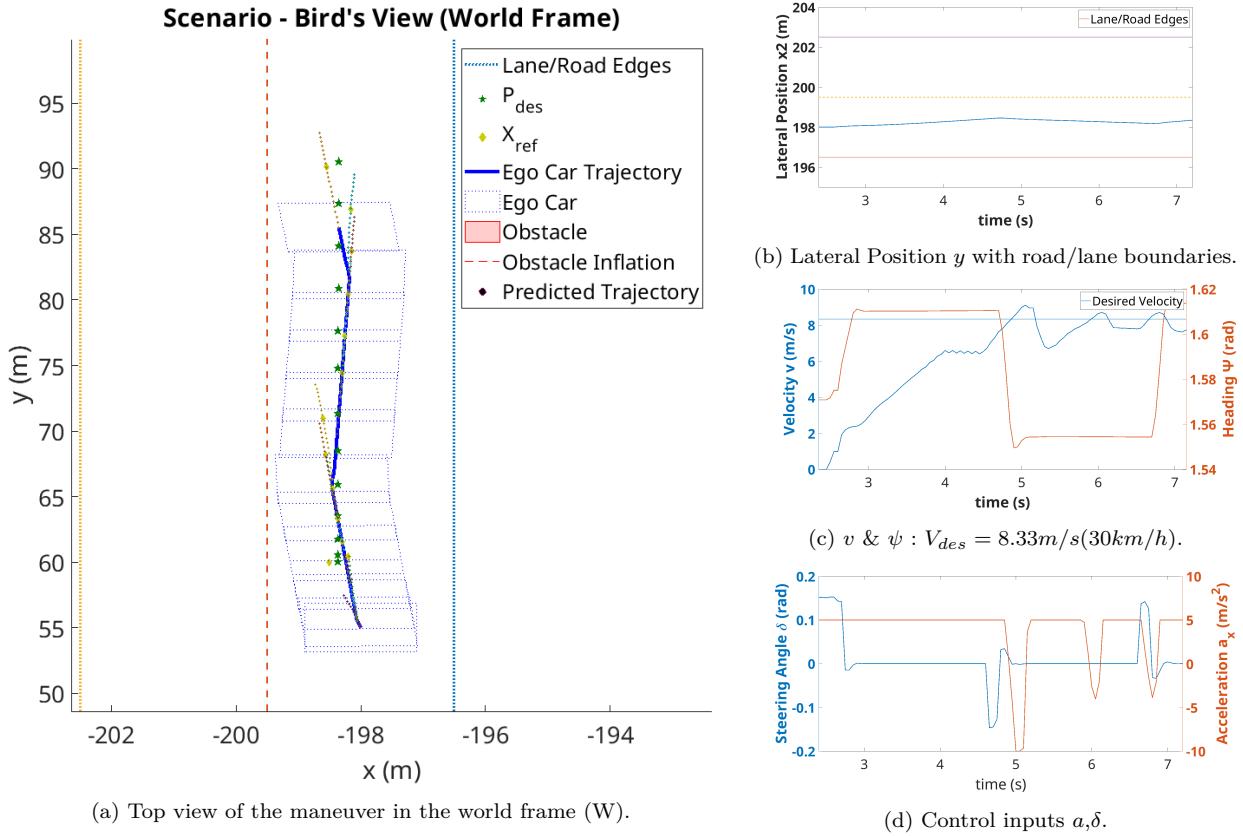


Figure 15: Simulation Results: Scenario 1 - Simple Lane Keeping.

Since in this scenario there are no obstacles in the form of any LVs, the lane-keeping maneuver is successfully selected by the rule-based BSM. p_{des} is selected as point, d_{cruise} distance ahead of the EV. EV was able to keep to the center of the lane while achieving the desired velocity as shown in Figure 15. The chosen reference points X_{ref} seem to

sway from side to side, leading to a slight oscillation in the overall trajectory. The control inputs that were generated by the NMPC do not violate any of the hard constraints, always keeping them between control input limits $[a_{min}, a_{max}]$ and $[\delta_{min}, \delta_{max}]$. However, sudden changes in acceleration input were observed which is detrimental to the comfort of the passengers.

Scenario 2 - Lane Change, Static Vehicle

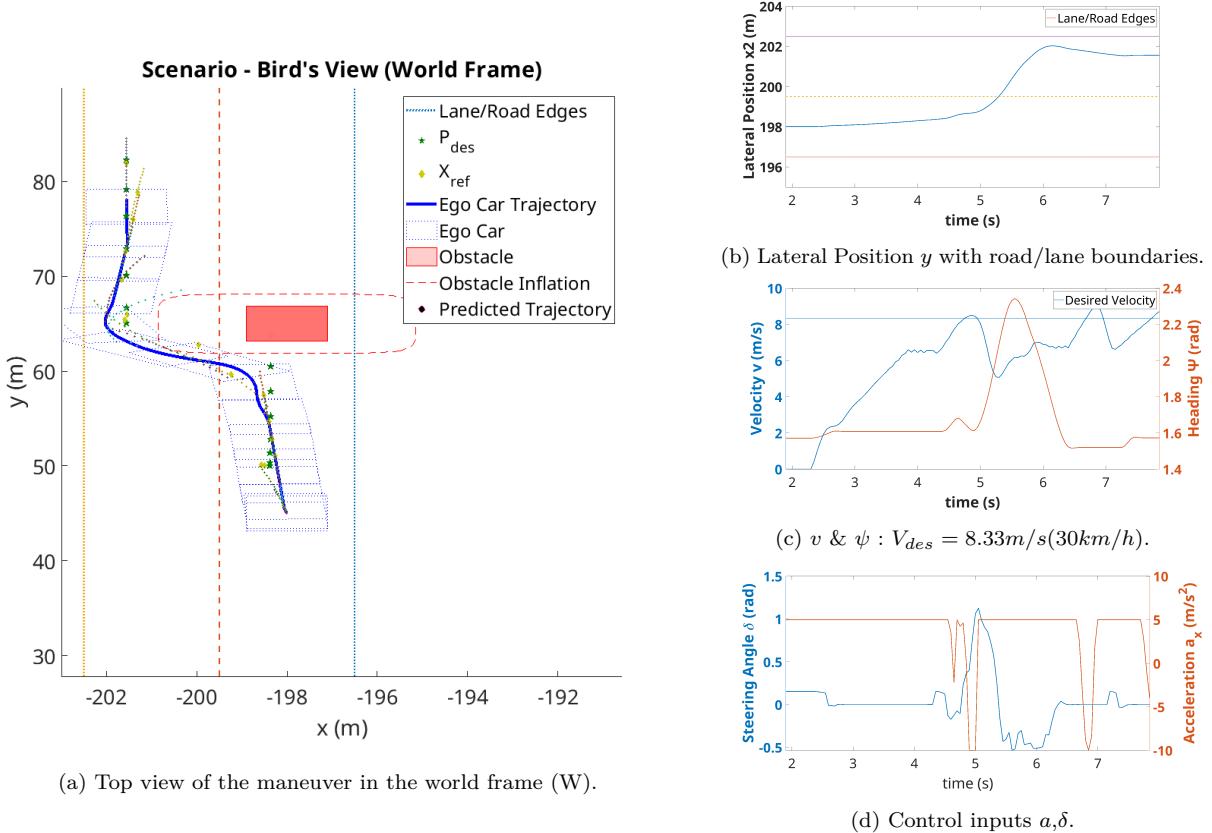


Figure 16: Simulation Results: Scenario 2 - Lane Change, Static Vehicle.

As seen in Figure 16, the EV approaches the LV and proceeds to attempt lane change to circumvent it. The p_{des} is selected as the center of the adjacent lane on the left. The EV then proceeds to complete the lane change maneuver meanwhile keeping a safe distance from the vehicle. No collisions were observed during the whole scenario. It then resumes lane-keeping and maintains v_{des} . The selection of X_{ref} shows a slight deviation from the lane center when lane-following, similar to the swaying nature observed in the previous scenario. At the moment just before the switch to lane change mode, the chosen X_{ref} starts to deviate to the left of the lane center even though the p_{des} is still at the center of the lane. This is due to the triangular velocity safety margins behind the LV that influence the safe set (not shown in Figure). The bounds of the EV violated the road boundaries for a short period during the lane change maneuver. This was due to the large inflation radius of the LV obstacle avoidance constraint as is evident from Figure 16a. This indicates that the inflation radius requires more fine-tuning. The control inputs that were generated by the NMPC do not violate any of the hard constraints, always keeping

them between control input limits $[a_{min}, a_{max}]$ and $[\delta_{min}, \delta_{max}]$. However, sudden changes in acceleration input were still observed.

Scenario 3 - Lane Change, Moving Vehicle

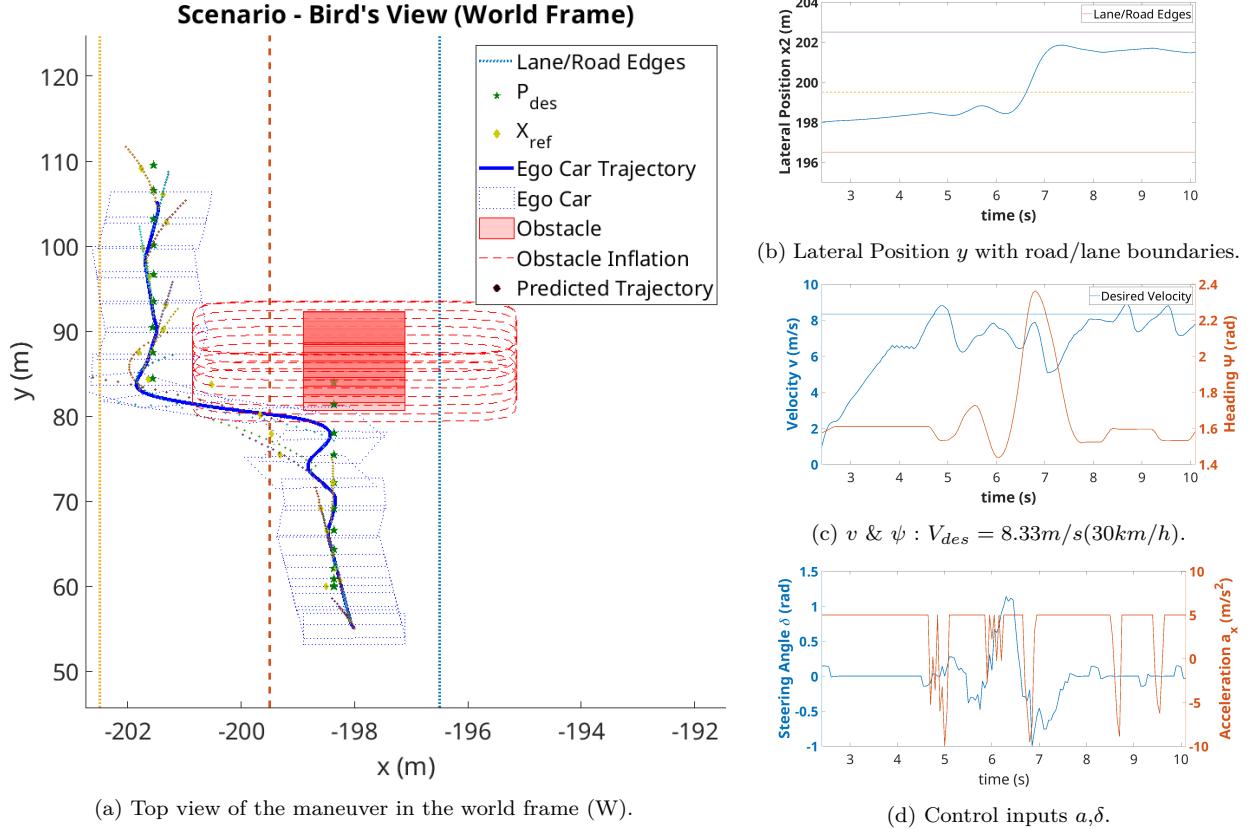


Figure 17: Simulation Results: Scenario 3 - Lane Change, Moving Vehicle.

As seen in Figure 17, the EV approaches the LV and proceeds to attempt lane-change to circumvent it. The LV is constantly moving forward and as the EV approaches the LV from behind, the triangular velocity safety margins cause the chosen X_{ref} deviate to the left, similar to the previous static lane-change scenario. Since the LV is still moving to influence the safe set, it causes the chosen X_{ref} to oscillate between the keeping the center of the lane and deviating to the left for short period. Eventually, EV approaches the LV close enough to trigger the lane-change and proceeds to complete the maneuver successfully. The EV then resumes lane-keeping afterward and maintains v_{des} . No collisions were observed during the entire scenario, though, road boundaries were violated by a corner of the EV for a very short period. The control inputs that were generated by the NMPC again do not violate any of the hard constraints, always keeping them between control input limits $[a_{min}, a_{max}]$ and $[\delta_{min}, \delta_{max}]$. However, sudden changes in acceleration input were more pronounced during this scenario.

5.2 Scenarios with Overtaking

Scenario 4 - Static Vehicle

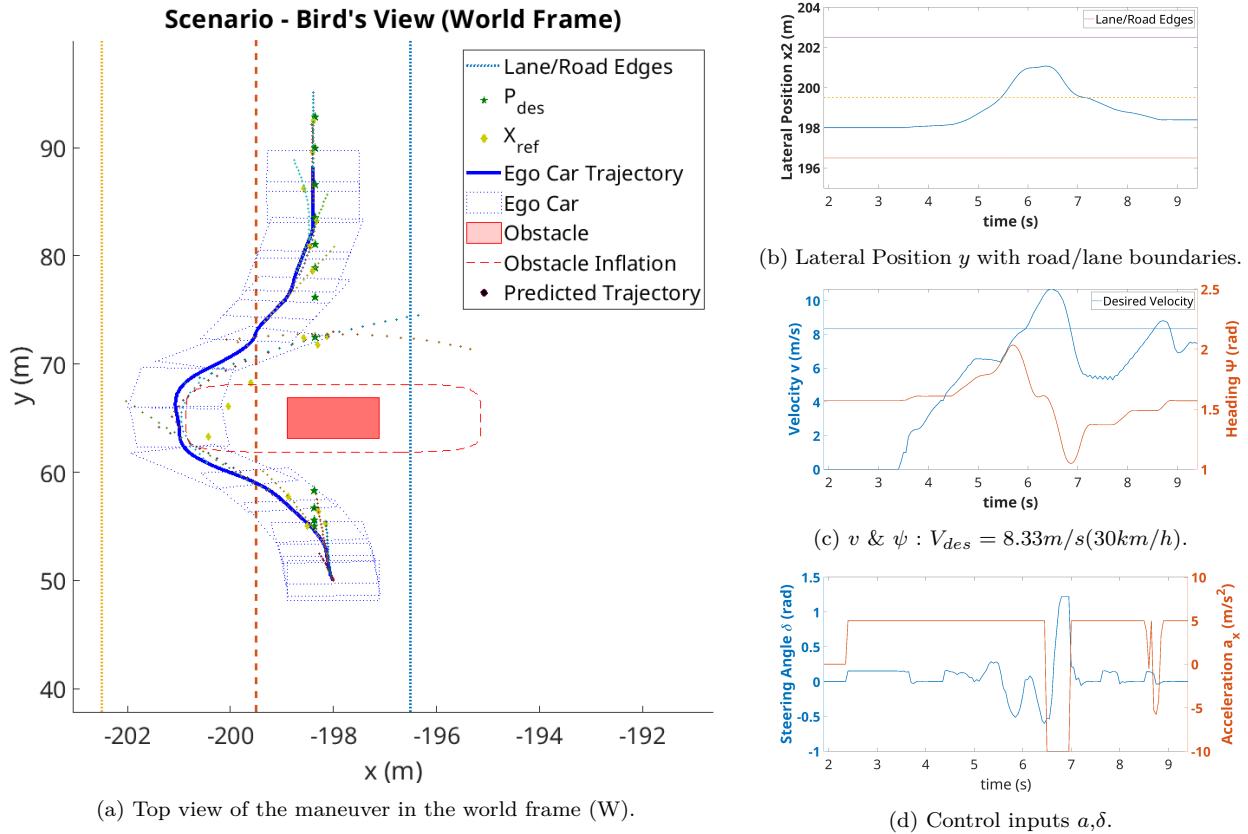


Figure 18: Simulation Results: Scenario 4 - Overtake - Static Vehicle.

Figure 18 shows the EV executing the overtaking maneuver. As it approaches the LV, the BSM detects the obstacle in front and switches to overtaking mode successfully. The p_{des} is chosen $d_{overtake}$ in front of the obstacle vehicle. The intermediate reference points X_{ref} that are chosen, guide the NMPC to perform the overtaking. The overall trajectory planned by the NMPC is smooth without any sudden heading changes. The overtaking maneuver causes the velocity to briefly fluctuate significantly above v_{des} as there are no hard constraints on the velocity of EV. No collisions were observed during the entire scenario and unlike previous scenarios, road boundaries were not violated by the bounds of the EV. The intrusion onto the adjacent lane was just enough to ensure no collisions and the time spent on the adjacent lane was also minimal (2s). The control inputs that were generated by the NMPC again do not violate any of the hard constraints, always keeping them between control input limits $[a_{min}, a_{max}]$ and $[\delta_{min}, \delta_{max}]$. However, sudden changes in acceleration input can still be observed. The steering angle inputs also seem to contain significant oscillations that might be undesirable to the comfort of the passenger, though, this does not translate to any oscillations in the actual heading of the vehicle. This may be due to the simulated inertia and limits on the rate of change of steering commands.

Scenario 5 - Small Static Vehicle

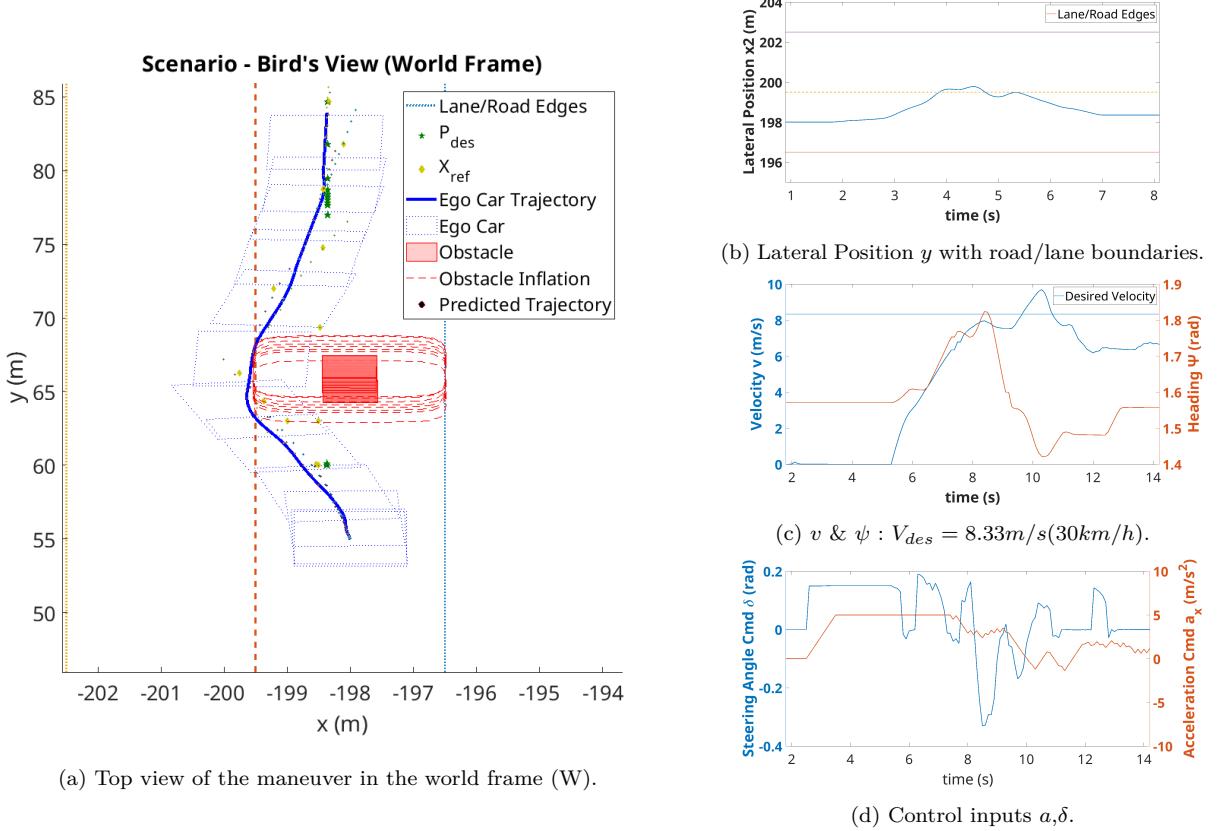


Figure 19: Simulation Results: Scenario 5 - Small LV.

As Figure 19 shows, the EV approaches the LV, completes the overtaking maneuver, and then resumes lane keeping. The graph showing the lateral position of the EV indicates that the planned trajectory has very minimal intrusion on to the adjacent lane without requiring any changes in the parameters. The EV did take longer to resume lane change, as p_{des} was chosen with the same $d_{overtake}$ distance from LV as in the previous scenario. No collisions were observed during the entire scenario. The control inputs that were generated by the NMPC again did not violate any of the hard constraints, always keeping them between control input limits $[a_{min}, a_{max}]$ and $[\delta_{min}, \delta_{max}]$. The steering angle inputs again seem to contain significant oscillations that are undesirable. Moreover, in this scenario, it also seems to affect the actual heading of the vehicle causing brief fluctuations in the heading.

Scenario 6 - Moving Vehicle

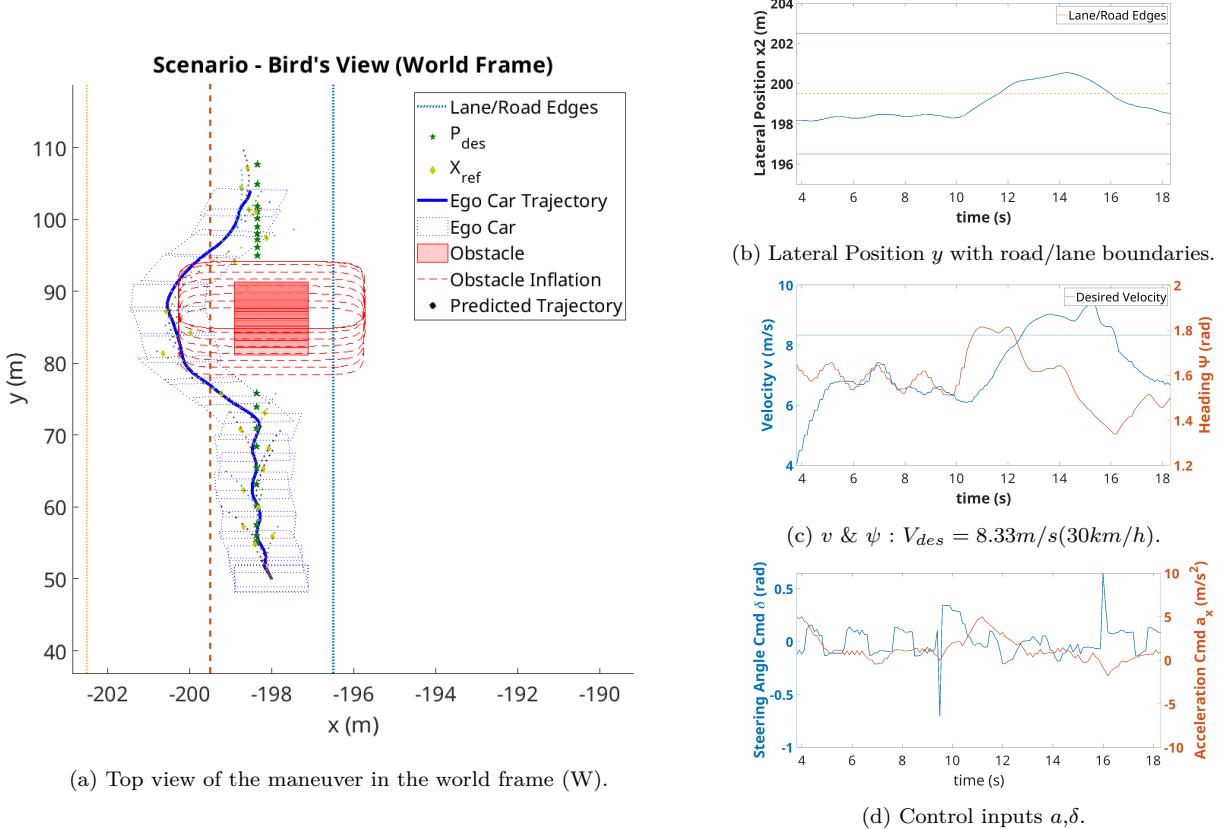


Figure 20: Simulation Results: Scenario 6 - Moving LV(3m/s).

As expected, the EV can overtake the LV successfully without any collision events. Figure 20 illustrates the trajectory in detail. The EV had to travel more distance in the adjacent lane to complete the overtake as the LV was moving; as a result, time spent on intruding on the adjacent lane was higher (4s). The chosen reference points X_{ref} again seem to sway from side to side, leading to the oscillation at the start of the maneuver. The same test scenario was also attempted with different velocities ($\leq v_{des}$) for the LV and the planning algorithm was able to successfully execute the overtaking maneuver every time. No collisions were observed during the entire scenario and road boundaries were not violated by the bounds of the EV at any point of time. The control inputs that were generated by the NMPC again do not violate any of the hard constraints, always keeping them between control input limits $[a_{min}, a_{max}]$ and $[\delta_{min}, \delta_{max}]$. Sudden changes in acceleration input were still observed. The steering angle inputs again seem to contain large oscillations especially during the end of the overtaking maneuver.

Scenario 7 - 2 Vehicles

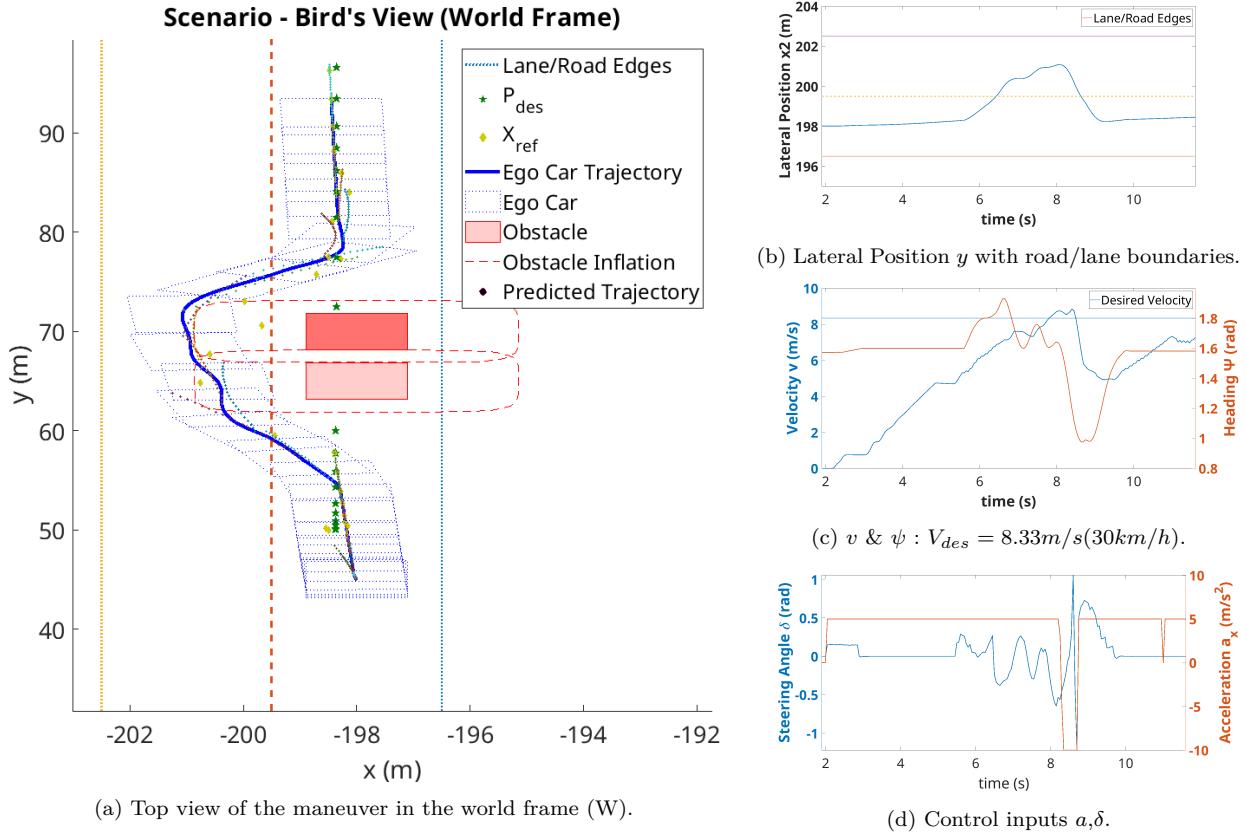


Figure 21: Simulation Results: Scenario 7 - 2 Vehicles, one in front of the other.

Figure 21 shows the EV executing the overtaking maneuver when there are 2 vehicles (one in front of the other). The EV approaches the first obstacle vehicle and the BSM selects the overtaking mode. p_{des} is selected as the point $d_{overtake}$ distance ahead of the first LV. As the overtaking maneuver is ongoing, the planning algorithm is notified of the second vehicle in front of the first which was initially occluded. The BSM then updates the p_{des} position to a point $d_{overtake}$ distance ahead of the second vehicle. The NMPC is only aware of one obstacle at a time. This is a limitation due to the implementation of NMPC in MATLAB rather than that of the proposed method. The collision avoidance constraints can be modified to include multiple obstacles in the future. The intermediate points X_{ref} chosen are still safe and reachable, hence, the overtaking maneuver is completed successfully. The EV then resumes lane keeping behaviour and tries to maintain v_{des} . No collisions were observed during the entire scenario and, again, road boundaries were not violated by the bounds of the EV at any point in time.

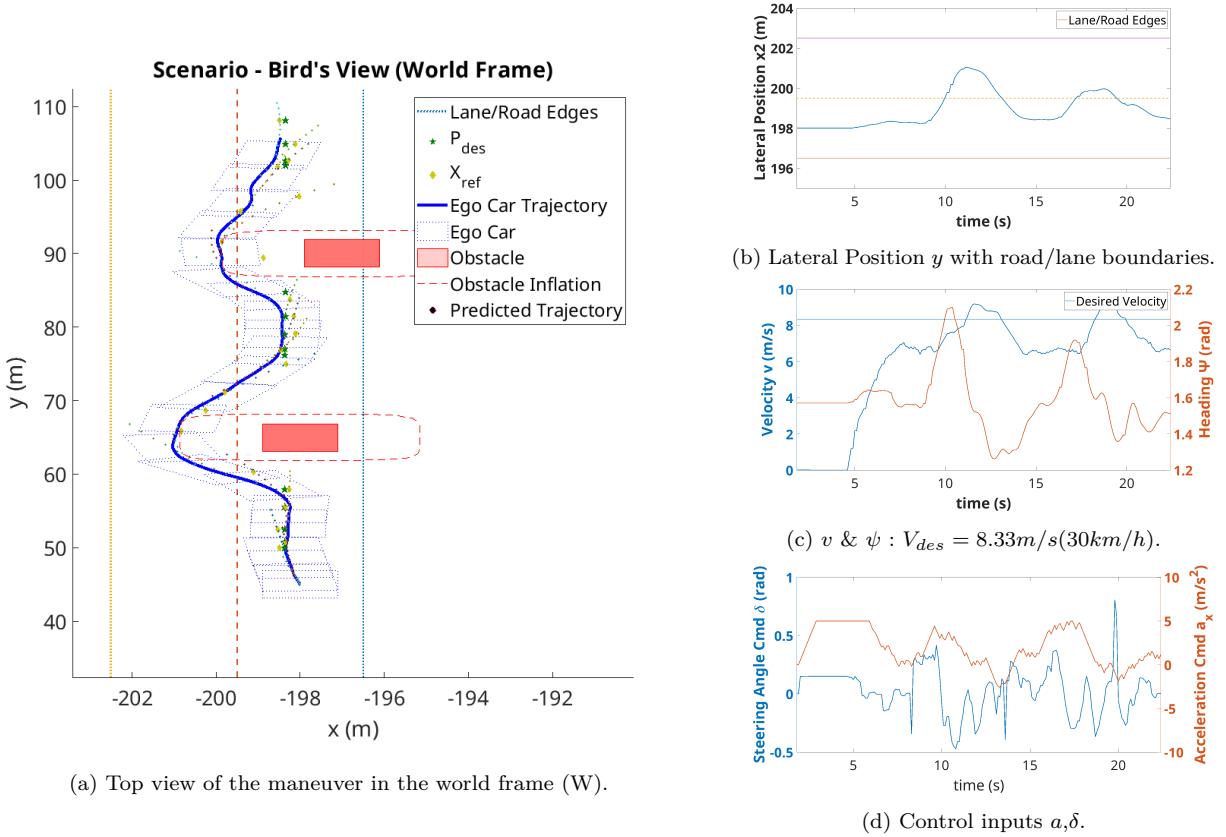


Figure 22: Simulation Results: Scenario 7 - 2 Vehicles, second LV parked to the side of the road.

The scenarios in Figure 22 is similar to the previous one, except the second LV is encountered after the EV resumes lane-keeping after overtaking the first LV. Since the second LV is parked to the side of the road, the EV does not need to encroach much to the oncoming lane and smoothly overtakes it to resume lane keeping. Both these scenarios show the adaptability of the proposed method to different road situations.

5.3 Discussion

The chosen reference points X_{ref} seem to sway from side to side, leading to oscillations primarily during lane-keeping maneuver. This is due to the grid resolution chosen for the safe set computation. Since X_{ref} is always chosen from this set, the point may not always coincide with the center of the lane causing the deviations. This can be resolved by higher grid resolutions but at the cost of computational requirements. In some scenarios, EV violated the road boundaries to account for the large inflation radius of the LV. This can be resolved by tuning the inflation radius of the LV. Another major issue that was observed was large oscillations in the acceleration and steering angle control inputs. This could cause heavy strain on an actual vehicle and prove to be detrimental to passenger comfort. This can be remedied by introducing limits on the rate of the change of these inputs. For example, the rate of change of acceleration (jerk) could be limited to ensure passenger comfort and reduce the strain of vehicle actuators.

5.3.1 Limitations

The proposed method is limited by the rule-based strategy which is fairly rudimentary at this stage. For example, a scenario was run where the LV was moving unpredictably with sudden changes in its heading and velocity after the overtaking maneuver started. Even though the NMPC was able to dynamically update the planned trajectory to avoid collisions, the EV was unable to recover fully and complete the overtaking maneuver. In situations similar to this, the BSM should have the option to abort the overtaking maneuver and fallback on lane-keeping mode. Thus, in situations where the overtaking and lane change can potentially be dangerous and unsafe, the current rule-based strategy should be extended to have the option to fallback and keep a safe distance from the LV. The MATLAB NMPC was also not well optimized, thus, ran considerably slow. The NMPC solver also had to compute the necessary Jacobians for system model and constraint functions through numerical perturbation since an analytical version of the Jacobians was not specified. In some simulation runs, the NMPC failed to optimize, consequently, failing to execute the selected maneuver. These optimization failures were rare and not repeatable and may be due to the ROS communication delays in state feedback. Only one obstacle avoidance constraint was active at every time step, thus, keeping the computational load quite low. This might not be feasible in more complex situations where the NMPC might need to avoid multiple obstacles while overtaking, for example, an oncoming vehicle on the other lane. These scenarios were not tested and the effectiveness of the behavior selection logic and NMPC under these situations is unknown.

The CARLA simulator and ROS framework proved problematic while testing the proposed method. The CARLA simulator lacks a solid vehicle dynamics engine and the parameters of the simulated vehicle were hard to determine. CARLA-ROS communication bridge introduced delays that are not ideal for testing real-time control applications. The tracking of the control commands a and δ_f by the PID and CARLA simulator is also poor, evident from Figure 23. This issue is also exacerbated by the communication delays and drops due to the ROS communication framework.

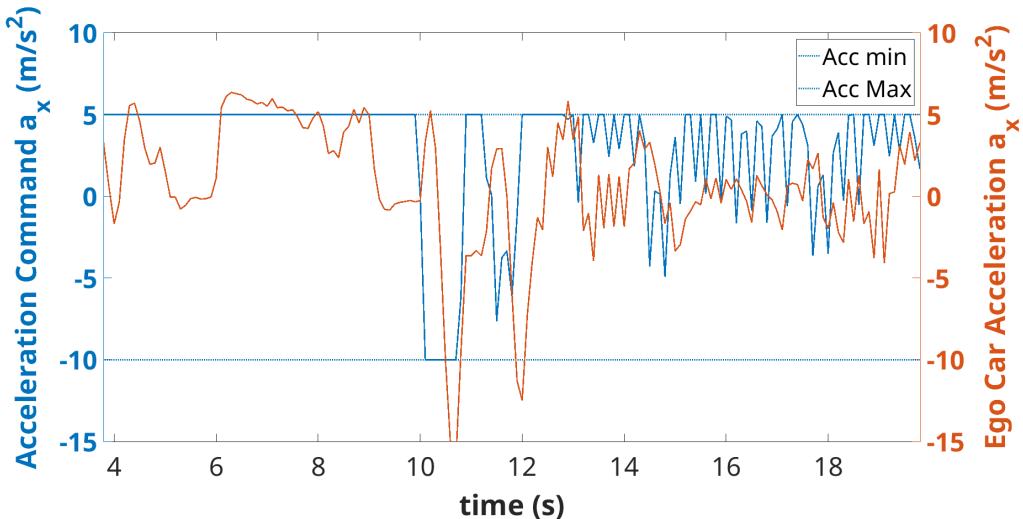


Figure 23: A comparison: Input Acceleration vs Actual vehicle acceleration (Max and min acceleration allowed is also shown.)

Limitations of Experimental Study

Even though the experimental study was designed to cover many complex scenarios and validate the proposed method with respect to many factors, some gaps were still identified. To ensure reliability, each scenario was run at least than 3 times, thus making sure the results were repeatable. The experiments were designed to include situations with more than one vehicle. However, the experimental study failed to cover some situations, primarily, ones that contain an oncoming vehicle in the opposite lane. The proposed method has a high chance of failure in these situations leading to potential collisions as there is no abort strategy. Scenarios with curved roads were omitted from the experimental study. However, the proposed method has shown promise in adapting to these road situations but requires validation through specifically designed test scenarios. The experimental study also failed evaluate the real-time performance of the proposed method as the experiments were conducted with the simulation step locked in (in step) with execution of the planning algorithm.

5.3.2 Summary

The proposed method was tested under different scenarios to gauge its effectiveness and generalization to many complex overtaking situations. In all of the scenarios discussed above, the planning algorithm successfully met its objectives which were to select an appropriate maneuver (lane keep, lane change, or overtake) and execute the maneuver avoiding collisions and obeying the system dynamics. The results show the versatility of the proposed method to adapt to the complex situations that are common in real-world driving. In comparison to other related works proposed previously, the behaviour planning module consisting of the rule-based maneuver selection and intermediate target selection was able to choose between various predefined maneuvers and select the optimal reference target for the NMPC suitable for the selected maneuver. This enabled the NMPC to concentrate on generating short, safe, dynamically feasible paths.

The NMPC, on the other hand, was also able to compensate for the deviations from the expected trajectory due to vehicle/system model mismatch and complete the maneuver successfully. This shows the robustness and advantage of having state feedback compared to the conventional optimal control methods that were discussed in the literature survey (Section 2). Also, compared to the MPC chosen for trajectory planning in related works, an NMPC can use a more accurate nonlinear system and can be subject to nonlinear constraints. To conclude, the results clearly show the merits of the proposed method in executing overtaking and related maneuvers in complex, dynamic scenarios.

6 Conclusions

AD has the potential to revolutionize the automobile industry by improving the safety, comfort, and convenience of passengers. When the majority of vehicles are driven by autonomous systems, traffic efficiency could be significantly improved. There are numerous challenges to overcome before a fully AD system becomes a reality. An AD system should be capable of controlling the vehicle without any human intervention. It must understand its environment, make the right decisions regarding maneuvers that move the vehicle closer to the destination whilst obeying relevant traffic rules, and finally, execute these maneuvers safely and efficiently. Overtaking and related maneuvers are a subset that the AD system should be capable of executing.

This thesis proposes a planning and control paradigm that can plan and execute overtaking maneuvers under complex scenarios, especially, those pertaining to single lane roads. The combination of safe and reachable sets to iteratively generate intermediate reference targets based on the desired maneuver, enabled the proposed method to re-plan and respond to the dynamically changing environment. It also allowed for the use of short prediction horizons, which can translate to better computational performance. This planning algorithm was implemented and then tested to validate its effectiveness under different overtaking scenarios. The experimental results have shown that the proposed method was successful in selecting the best maneuver and executing it optimally in most of the test scenarios, requiring minimal intrusion on to the adjacent lane. The hierarchical approach combining behavioral planning and trajectory planning allows for use of the best-suited method for any particular task. This allows for easy design and implementation since the control architecture can be divided into individual modules. However, it is hard to integrate them and can potentially lead to compatibility issues.

AD is a hard problem to solve. The task of developing a method that only handles a few select maneuvers itself proved to be challenging, even with the crutch of testing in a simulation environment. Development of an AD stack that can perform real-world driving, making the right decisions, and executing all necessary maneuvers without human intervention maybe even more challenging. Moreover, ensuring that the AD stack works as designed in all situations, even which are highly improbable, is of utmost importance for the general acceptance of AVs in the future. Thus, it is necessary to test the system in complex scenarios to gauge its effectiveness and identify areas that are problematic or might have been overlooked. Another observation was that existing methods in the control engineering toolbox, e.g. MPC, can be effective in AD development, especially in planning and control. These methods have strong stability and robustness theory backing, which is of great importance for ensuring that the AD controllers built using these methods are safe, stable, and robust. They may prove to be the backbone of the AD stack, on which machine learning-oriented methods (e.g. RL) can be applied for higher-level planning tasks and to enable learning from data.

6.1 Future Work

The test scenarios that were run showed that the current heuristics rule-based behaviour has to be significantly expanded to handle all edge cases, e.g., aborting of the maneuver if unsafe. Identifying all possible cases and then developing heuristic-based rules for optimal decision making requires in-depth domain knowledge and skill. Even then, it may fail in highly improbable but possible situations. A possible replacement to heuristic

rule-based FSMs is MDP or RL based methods alluded to in Section 2. These methods are better suited in capturing the probabilistic nature of the decision-making process and also enable learning from data. The NMPC can also be improved in many ways. The SQP solver used can be further optimized by providing analytical Jacobians, which can dramatically increase the computational efficiency. The collision avoidance constraints can be augmented to account for the predicted trajectory of the LV, which means less dynamic modifications to the initial planned trajectory are required at every time step. The current algorithm assumes deterministic values for the vehicle state and environment information. Moreover, the system is assumed to be fully observable, which is not possible in real practice. A state estimator that accounts for process and measurement noise and also computes unobserved states from measurements can be incorporated. The simulation environment better suited for vehicle dynamics together with an improved test setup can help eliminate the issues faced due to communication delays and sub-par performance of the PID controller in controlling the EV in CARLA simulation.

References

- [1] D. Pojani and D. Stead, "Sustainable Urban Transport in the Developing World: Beyond Megacities," *Sustainability*, vol. 7, no. 6, pp. 7784–7805, 6 Jun. 2015. DOI: [10.3390/su7067784](https://doi.org/10.3390/su7067784). [Online]. Available: <https://www.mdpi.com/2071-1050/7/6/7784> (visited on 12/15/2020).
- [2] D. Watzenig and M. Horn, "Introduction to Automated Driving," in *Automated Driving: Safer and More Efficient Future Driving*, D. Watzenig and M. Horn, Eds., Cham: Springer International Publishing, 2017, pp. 3–16, ISBN: 978-3-319-31895-0. DOI: [10.1007/978-3-319-31895-0_1](https://doi.org/10.1007/978-3-319-31895-0_1). [Online]. Available: https://doi.org/10.1007/978-3-319-31895-0_1 (visited on 10/12/2020).
- [3] W. H. Organization, *Global Status Report on Road Safety 2018*, ser. Nonserial Publication. World Health Organization, 2019, ISBN: 978-92-4-156568-4. [Online]. Available: <https://www.who.int/publications/item/9789241565684> (visited on 11/12/2020).
- [4] S. Singh, "Critical Reasons for Crashes Investigated in the National Motor Vehicle Crash Causation Survey.," National Highway Traffic Safety Administration, Washington, DC, DOT HS 812 115, Feb. 2015. [Online]. Available: <http://www-nrd.nhtsa.dot.gov/Pubs/812115.pdf> (visited on 11/08/2020).
- [5] S. Otmani, T. Pebayle, J. Roge, and A. Muzet, "Effect of driving duration and partial sleep deprivation on subsequent alertness and performance of car drivers," *Physiology & Behavior*, vol. 84, no. 5, pp. 715–724, Apr. 13, 2005, ISSN: 0031-9384. DOI: [10.1016/j.physbeh.2005.02.021](https://doi.org/10.1016/j.physbeh.2005.02.021). [Online]. Available: [http://www.sciencedirect.com/science/article/pii/S0031938405000740](https://www.sciencedirect.com/science/article/pii/S0031938405000740) (visited on 11/07/2020).
- [6] K. Spieser, K. Treleaven, R. Zhang, E. Fazzoli, D. Morton, and M. Pavone, "Toward a Systematic Approach to the Design and Evaluation of Automated Mobility-on-Demand Systems: A Case Study in Singapore," in *Road Vehicle Automation*, ser. Lecture Notes in Mobility, G. Meyer and S. Beiker, Eds., Cham: Springer International Publishing, 2014, pp. 229–245, ISBN: 978-3-319-05990-7. DOI: [10.1007/978-3-319-05990-7_20](https://doi.org/10.1007/978-3-319-05990-7_20). [Online]. Available: https://doi.org/10.1007/978-3-319-05990-7_20 (visited on 11/07/2020).
- [7] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving," *Journal of Field Robotics*, vol. 37, no. 3, pp. 362–386, 2020, ISSN: 1556-4967. DOI: [10.1002/rob.21918](https://doi.org/10.1002/rob.21918). [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21918> (visited on 11/09/2020).
- [8] S. o. A. Engineers, "J3016B: Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles - SAE International," SAE International, 2018. [Online]. Available: https://www.sae.org/standards/content/j3016_201806/ (visited on 11/07/2020).
- [9] (). "Website of the Federal Statistical Office of Germany (Statistisches Bundesamt) - Destatis," Federal Statistical Office, [Online]. Available: https://www.destatis.de/EN/Home/_node.html (visited on 11/12/2020).

- [10] S. Dixit, U. Montanaro, S. Fallah, M. Dianati, D. Oxtoby, T. Mizutani, and A. Mouzakitis, “Trajectory Planning for Autonomous High-Speed Overtaking using MPC with Terminal Set Constraints,” in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, Nov. 2018, pp. 1061–1068. DOI: [10.1109/ITSC.2018.8569529](https://doi.org/10.1109/ITSC.2018.8569529).
- [11] S. Dixit, U. Montanaro, M. Dianati, D. Oxtoby, T. Mizutani, A. Mouzakitis, and S. Fallah, “Trajectory Planning for Autonomous High-Speed Overtaking in Structured Environments Using Robust MPC,” *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–14, 2019, ISSN: 1558-0016. DOI: [10.1109/TITS.2019.2916354](https://doi.org/10.1109/TITS.2019.2916354).
- [12] (). “Tesla Autopilot,” [Online]. Available: <https://www.tesla.com/autopilot> (visited on 11/10/2020).
- [13] (). “Waypoint - The official Waymo blog: Introducing the 5th-generation Waymo Driver: Informed by experience, designed for scale, engineered to tackle more environments,” Waymo Blog, [Online]. Available: <https://blog.waymo.com/2020/03/introducing-5th-generation-waymo-driver.html> (visited on 11/10/2020).
- [14] C. V. Poulton, A. Yaacobi, D. B. Cole, M. J. Byrd, M. Raval, D. Vermeulen, and M. R. Watts, “Coherent solid-state LIDAR with silicon photonic optical phased arrays,” *Optics Letters*, vol. 42, no. 20, pp. 4091–4094, Oct. 15, 2017, ISSN: 1539-4794. DOI: [10.1364/OL.42.004091](https://doi.org/10.1364/OL.42.004091). [Online]. Available: <https://www.osapublishing.org/ol/abstract.cfm?uri=ol-42-20-4091> (visited on 11/10/2020).
- [15] R. Negenborn, “Robot Localization and Kalman Filters. On finding your position in a noisy world,” M.S. thesis, Utrecht University, Copenhagen University (DIKU), Denmark, 2003, 155 pp. [Online]. Available: http://www.negenborn.net/kal_loc/thesis.pdf.
- [16] E. W. Dijkstra, “A note on two problems in connexion with graphs,” *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, Dec. 1, 1959, ISSN: 0945-3245. DOI: [10.1007/BF01386390](https://doi.org/10.1007/BF01386390). [Online]. Available: <https://doi.org/10.1007/BF01386390>.
- [17] P. Hart, N. Nilsson, and B. Raphael, “A Formal Basis for the Heuristic Determination of Minimum Cost Paths,” *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968, ISSN: 0536-1567. DOI: [10.1109/TSSC.1968.300136](https://doi.org/10.1109/TSSC.1968.300136). [Online]. Available: <http://ieeexplore.ieee.org/document/4082128/> (visited on 12/20/2020).
- [18] B. Paden, M. Cap, S. Z. Yong, D. Yershov, and E. Frazzoli. (Apr. 25, 2016). “A Survey of Motion Planning and Control Techniques for Self-driving Urban Vehicles.” arXiv: [1604.07446 \[cs\]](https://arxiv.org/abs/1604.07446), [Online]. Available: [http://arxiv.org/abs/1604.07446](https://arxiv.org/abs/1604.07446) (visited on 11/09/2020).
- [19] H. Bast, D. Delling, A. Goldberg, M. Müller-Hannemann, T. Pajor, P. Sanders, D. Wagner, and R. F. Werneck, “Route Planning in Transportation Networks,” in *Algorithm Engineering: Selected Results and Surveys*, ser. Lecture Notes in Computer Science, L. Kliemann and P. Sanders, Eds., Cham: Springer International Publishing, 2016, pp. 19–80, ISBN: 978-3-319-49487-6. DOI: [10.1007/978-3-319-49487-6_2](https://doi.org/10.1007/978-3-319-49487-6_2). [Online]. Available: https://doi.org/10.1007/978-3-319-49487-6_2 (visited on 11/10/2020).

- [20] S. Dixit, S. Fallah, U. Montanaro, M. Dianati, A. Stevens, F. Mccullough, and A. Mouzakitis, “Trajectory planning and tracking for autonomous overtaking: State-of-the-art and future prospects,” *Annual Reviews in Control*, vol. 45, pp. 76–86, Jan. 1, 2018, ISSN: 1367-5788. DOI: [10.1016/j.arcontrol.2018.02.001](https://doi.org/10.1016/j.arcontrol.2018.02.001). [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S136757881730130X> (visited on 03/31/2020).
- [21] M. Buehler, K. Iagnemma, and S. Singh, Eds., *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic*, ser. Springer Tracts in Advanced Robotics. Berlin Heidelberg: Springer-Verlag, 2009, ISBN: 978-3-642-03990-4. DOI: [10.1007/978-3-642-03991-1](https://doi.org/10.1007/978-3-642-03991-1). [Online]. Available: <https://www.springer.com/gp/book/9783642039904> (visited on 12/20/2020).
- [22] M. Ardelt, C. Coester, and N. Kaempchen, “Highly Automated Driving on Freeways in Real Traffic Using a Probabilistic Framework,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 4, pp. 1576–1585, Dec. 2012, ISSN: 1558-0016. DOI: [10.1109/TITS.2012.2196273](https://doi.org/10.1109/TITS.2012.2196273).
- [23] S. Brechtel, T. Gindele, and R. Dillmann, “Probabilistic MDP-behavior planning for cars,” in *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, Oct. 2011, pp. 1537–1542. DOI: [10.1109/ITSC.2011.6082928](https://doi.org/10.1109/ITSC.2011.6082928).
- [24] A. Safiotti, “Fuzzy logic in autonomous robotics: Behavior coordination,” in *Proceedings of 6th International Fuzzy Systems Conference*, vol. 1, Jul. 1997, 573–578 vol.1. DOI: [10.1109/FUZZY.1997.616430](https://doi.org/10.1109/FUZZY.1997.616430).
- [25] H. A. Hagras, “A hierarchical type-2 fuzzy logic control architecture for autonomous mobile robots,” *IEEE Transactions on Fuzzy Systems*, vol. 12, no. 4, pp. 524–539, Aug. 2004, ISSN: 1941-0034. DOI: [10.1109/TFUZZ.2004.832538](https://doi.org/10.1109/TFUZZ.2004.832538).
- [26] J. Jaafar, E. McKenzie, and A. Smaill, “A Fuzzy Action Selection Method for Virtual Agent Navigation in Unknown Virtual Environments,” in *2007 IEEE International Fuzzy Systems Conference*, Jul. 2007, pp. 1–6. DOI: [10.1109/FUZZY.2007.4295541](https://doi.org/10.1109/FUZZY.2007.4295541).
- [27] C. You, J. Lu, D. Filev, and P. Tsotras, “Highway Traffic Modeling and Decision Making for Autonomous Vehicle Using Reinforcement Learning,” in *2018 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2018, pp. 1227–1232. DOI: [10.1109/IVS.2018.8500675](https://doi.org/10.1109/IVS.2018.8500675).
- [28] M. T. J. Spaan, “Partially Observable Markov Decision Processes,” in *Reinforcement Learning: State-of-the-Art*, ser. Adaptation, Learning, and Optimization, M. Wiering and M. van Otterlo, Eds., Berlin, Heidelberg: Springer, 2012, pp. 387–414, ISBN: 978-3-642-27645-3. DOI: [10.1007/978-3-642-27645-3_12](https://doi.org/10.1007/978-3-642-27645-3_12). [Online]. Available: https://doi.org/10.1007/978-3-642-27645-3_12 (visited on 11/11/2020).
- [29] S. Brechtel, T. Gindele, and R. Dillmann, “Probabilistic decision-making under uncertainty for autonomous driving using continuous POMDPs,” in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, Oct. 2014, pp. 392–399. DOI: [10.1109/ITSC.2014.6957722](https://doi.org/10.1109/ITSC.2014.6957722).
- [30] S. Ulbrich and M. Maurer, “Towards Tactical Lane Change Behavior Planning for Automated Vehicles,” in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, Sep. 2015, pp. 989–995. DOI: [10.1109/ITSC.2015.165](https://doi.org/10.1109/ITSC.2015.165).

- [31] G. Shani, J. Pineau, and R. Kaplow, “A survey of point-based POMDP solvers,” *Autonomous Agents and Multi-Agent Systems*, vol. 27, no. 1, pp. 1–51, Jul. 2013, ISSN: 1387-2532, 1573-7454. DOI: [10.1007/s10458-012-9200-2](https://doi.org/10.1007/s10458-012-9200-2). [Online]. Available: <http://link.springer.com/10.1007/s10458-012-9200-2> (visited on 11/12/2020).
- [32] Z. Qiao, Z. Tyree, P. Mudalige, J. Schneider, and J. M. Dolan. (Nov. 9, 2019). “Hierarchical Reinforcement Learning Method for Autonomous Vehicle Behavior Planning.” arXiv: [1911.03799 \[cs\]](https://arxiv.org/abs/1911.03799), [Online]. Available: <http://arxiv.org/abs/1911.03799> (visited on 11/06/2020).
- [33] C. You, J. Lu, D. Filev, and P. Tsiotras, “Advanced planning for autonomous vehicles using reinforcement learning and deep inverse reinforcement learning,” *Robotics and Autonomous Systems*, vol. 114, pp. 1–18, Apr. 1, 2019, ISSN: 0921-8890. DOI: [10.1016/j.robot.2019.01.003](https://doi.org/10.1016/j.robot.2019.01.003). [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0921889018302021> (visited on 11/12/2020).
- [34] A. Defazio and T. Graepel. (Oct. 30, 2014). “A Comparison of learning algorithms on the Arcade Learning Environment.” arXiv: [1410.8620 \[cs\]](https://arxiv.org/abs/1410.8620), [Online]. Available: <http://arxiv.org/abs/1410.8620> (visited on 11/12/2020).
- [35] M. A. Mousavi, Z. Heshmati, and B. Moshiri, “LTV-MPC based path planning of an autonomous vehicle via convex optimization,” in *2013 21st Iranian Conference on Electrical Engineering (ICEE)*, May 2013, pp. 1–7. DOI: [10.1109/IranianCEE.2013.6599610](https://doi.org/10.1109/IranianCEE.2013.6599610).
- [36] F. Havlak and M. Campbell, “Discrete and Continuous, Probabilistic Anticipation for Autonomous Robots in Urban Environments,” *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 461–474, Apr. 2014, ISSN: 1941-0468. DOI: [10.1109/TRO.2013.2291620](https://doi.org/10.1109/TRO.2013.2291620).
- [37] Q. Tran and J. Firl, “Modelling of traffic situations at urban intersections with probabilistic non-parametric regression,” in *2013 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2013, pp. 334–339. DOI: [10.1109/IVS.2013.6629491](https://doi.org/10.1109/IVS.2013.6629491).
- [38] L. Ma, J. Xue, K. Kawabata, J. Zhu, C. Ma, and N. Zheng, “A fast RRT algorithm for motion planning of autonomous road vehicles,” in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, Oct. 2014, pp. 1033–1038. DOI: [10.1109/ITSC.2014.6957824](https://doi.org/10.1109/ITSC.2014.6957824).
- [39] W. Khaksar, K. S. M. Sahari, and T. S. Hong, “Application of Sampling-Based Motion Planning Algorithms in Autonomous Vehicle Navigation,” *Autonomous Vehicle*, Sep. 7, 2016. DOI: [10.5772/64730](https://doi.org/10.5772/64730). [Online]. Available: <https://www.intechopen.com/books/autonomous-vehicle/application-of-sampling-based-motion-planning-algorithms-in-autonomous-vehicle-navigation> (visited on 11/14/2020).
- [40] C. Katrakazas, M. Quddus, W.-H. Chen, and L. Deka, “Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions,” *Transportation Research Part C: Emerging Technologies*, vol. 60, pp. 416–442, Nov. 1, 2015, ISSN: 0968-090X. DOI: [10.1016/j.trc.2015.09.011](https://doi.org/10.1016/j.trc.2015.09.011). [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0968090X15003447> (visited on 11/13/2020).

- [41] Y. K. Hwang and N. Ahuja, "Path planning using a potential field representation," in *1988 IEEE International Conference on Robotics and Automation Proceedings*, Apr. 1988, 648–649 vol.1. DOI: [10.1109/ROBOT.1988.12131](https://doi.org/10.1109/ROBOT.1988.12131).
- [42] S. Glaser, B. Vanholme, S. Mammar, D. Gruyer, and L. Nouvelière, "Maneuver-Based Trajectory Planning for Highly Autonomous Vehicles on Real Road With Traffic and Driver Interaction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 3, pp. 589–606, Sep. 2010, ISSN: 1558-0016. DOI: [10.1109/TITS.2010.2046037](https://doi.org/10.1109/TITS.2010.2046037).
- [43] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *1985 IEEE International Conference on Robotics and Automation Proceedings*, vol. 2, Mar. 1985, pp. 500–505. DOI: [10.1109/ROBOT.1985.1087247](https://doi.org/10.1109/ROBOT.1985.1087247).
- [44] J. Borenstein and Y. Koren, "The vector field histogram-fast obstacle avoidance for mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, pp. 278–288, Jun. 1991, ISSN: 1042296X. DOI: [10.1109/70.88137](https://doi.org/10.1109/70.88137). [Online]. Available: <http://ieeexplore.ieee.org/document/88137/> (visited on 11/13/2020).
- [45] S. Quinlan and O. Khatib, "Elastic bands: Connecting path planning and control," in *[1993] Proceedings IEEE International Conference on Robotics and Automation*, May 1993, 802–807 vol.2. DOI: [10.1109/ROBOT.1993.291936](https://doi.org/10.1109/ROBOT.1993.291936).
- [46] M. T. Wolf and J. W. Burdick, "Artificial potential functions for highway driving with collision avoidance," in *2008 IEEE International Conference on Robotics and Automation*, May 2008, pp. 3731–3736. DOI: [10.1109/ROBOT.2008.4543783](https://doi.org/10.1109/ROBOT.2008.4543783).
- [47] V. S. Chipade, Q. Shen, L. Huang, N. Ozay, S. Z. Yong, and D. Panagou, "Safe Autonomous Overtaking with Intention Estimation," in *2019 18th European Control Conference (ECC)*, Jun. 2019, pp. 2050–2057. DOI: [10.23919/ECC.2019.8795715](https://doi.org/10.23919/ECC.2019.8795715).
- [48] "Constrained Finite Time Optimal Control," in *Constrained Optimal Control of Linear and Hybrid Systems*, ser. Lecture Notes in Control and Information Sciences, F. Borrelli, Ed., Berlin, Heidelberg: Springer, 2003, pp. 51–69, ISBN: 978-3-540-36225-8. DOI: [10.1007/3-540-36225-8_2](https://doi.org/10.1007/3-540-36225-8_2). [Online]. Available: https://doi.org/10.1007/3-540-36225-8_2 (visited on 11/14/2020).
- [49] W. Li and E. Todorov, "ITERATIVE LINEAR QUADRATIC REGULATOR DESIGN FOR NONLINEAR BIOLOGICAL MOVEMENT SYSTEMS," in *Proceedings of the First International Conference on Informatics in Control, Automation and Robotics - Volume 1: ICINCO*, INSTICC, SciTePress, 2004, pp. 222–229, ISBN: 972-8865-12-0. DOI: [10.5220/0001143902220229](https://doi.org/10.5220/0001143902220229).
- [50] J. van den Berg, "Extended LQR: Locally-Optimal Feedback Control for Systems with Non-Linear Dynamics and Non-Quadratic Cost," in *Robotics Research: The 16th International Symposium ISRR*, ser. Springer Tracts in Advanced Robotics, M. Inaba and P. Corke, Eds., Cham: Springer International Publishing, 2016, pp. 39–56, ISBN: 978-3-319-28872-7. DOI: [10.1007/978-3-319-28872-7_3](https://doi.org/10.1007/978-3-319-28872-7_3). [Online]. Available: https://doi.org/10.1007/978-3-319-28872-7_3 (visited on 11/14/2020).
- [51] J. Chen, W. Zhan, and M. Tomizuka, "Autonomous Driving Motion Planning With Constrained Iterative LQR," *IEEE Transactions on Intelligent Vehicles*, vol. 4, no. 2, pp. 244–254, Jun. 2019, ISSN: 2379-8904. DOI: [10.1109/TIV.2019.2904385](https://doi.org/10.1109/TIV.2019.2904385).

- [52] K. Chu, M. Lee, and M. Sunwoo, “Local Path Planning for Off-Road Autonomous Driving With Avoidance of Static Obstacles,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 4, pp. 1599–1616, Dec. 2012, ISSN: 1558-0016. DOI: [10.1109/TITS.2012.2198214](https://doi.org/10.1109/TITS.2012.2198214).
- [53] M. H. Moradi, “Predictive control with constraints, J.M. maciejowski; pearson education limited, prentice hall, london, 2002, pp. IX+331, price £35.99, ISBN 0-201-39823-0,” *International Journal of Adaptive Control and Signal Processing*, vol. 17, no. 3, pp. 261–262, 2003. DOI: [10.1002/acs.736](https://doi.org/10.1002/acs.736). eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/acs.736>. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/acs.736>.
- [54] M. L. Darby, “Industrial MPC of Continuous Processes,” in *Encyclopedia of Systems and Control*, J. Baillieul and T. Samad, Eds., London: Springer, 2013, pp. 1–10, ISBN: 978-1-4471-5102-9. DOI: [10.1007/978-1-4471-5102-9_242-1](https://doi.org/10.1007/978-1-4471-5102-9_242-1). [Online]. Available: https://doi.org/10.1007/978-1-4471-5102-9_242-1 (visited on 11/14/2020).
- [55] C. Shen, Y. Shi, and B. Buckham, “Integrated Path Planning and Tracking Control of an AUV: A Unified Receding Horizon Optimization Approach,” *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 3, pp. 1163–1173, Jun. 2017, ISSN: 1941-014X. DOI: [10.1109/TMECH.2016.2612689](https://doi.org/10.1109/TMECH.2016.2612689).
- [56] N. Murgovski and J. Sjöberg, “Predictive cruise control with autonomous overtaking,” in *2015 54th IEEE Conference on Decision and Control (CDC)*, Dec. 2015, pp. 644–649. DOI: [10.1109/CDC.2015.7402302](https://doi.org/10.1109/CDC.2015.7402302).
- [57] F. Molinari, N. N. Anh, and L. Del Re, “Efficient mixed integer programming for autonomous overtaking,” in *2017 American Control Conference (ACC)*, May 2017, pp. 2303–2308. DOI: [10.23919/ACC.2017.7963296](https://doi.org/10.23919/ACC.2017.7963296).
- [58] A. Carvalho, S. Lefévre, G. Schildbach, J. Kong, and F. Borrelli, “Automated driving: The role of forecasts and uncertainty—A control perspective,” *European Journal of Control*, SI: ECC15, vol. 24, pp. 14–32, Jul. 1, 2015, ISSN: 0947-3580. DOI: [10.1016/j.ejcon.2015.04.007](https://doi.org/10.1016/j.ejcon.2015.04.007). [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S094735801500062X> (visited on 04/02/2020).
- [59] S. Li, Z. Li, Z. Yu, B. Zhang, and N. Zhang, “Dynamic Trajectory Planning and Tracking for Autonomous Vehicle With Obstacle Avoidance Based on Model Predictive Control,” *IEEE Access*, vol. 7, pp. 132 074–132 086, 2019, ISSN: 2169-3536. DOI: [10.1109/ACCESS.2019.2940758](https://doi.org/10.1109/ACCESS.2019.2940758). [Online]. Available: <https://ieeexplore.ieee.org/document/8835033/> (visited on 11/13/2020).
- [60] Y. Pan, Q. Lin, H. Shah, and J. M. Dolan. (Mar. 5, 2020). “Safe Planning for Self-Driving Via Adaptive Constrained ILQR.” arXiv: [2003.02757 \[cs, eess\]](https://arxiv.org/abs/2003.02757), [Online]. Available: [http://arxiv.org/abs/2003.02757](https://arxiv.org/abs/2003.02757) (visited on 10/25/2020).
- [61] T. Liu, Z. Deng, X. Tang, H. Wang, and D. Cao, “Predictive Freeway Overtaking Strategy for Automated Vehicles Using Deep Reinforcement Learning,” in *2019 3rd Conference on Vehicle Control and Intelligence (CVCI)*, Sep. 2019, pp. 1–6. DOI: [10.1109/CVCI47823.2019.8951536](https://doi.org/10.1109/CVCI47823.2019.8951536).

- [62] P. Corke, *Robotics, Vision and Control: Fundamental Algorithms In MATLAB® Second, Completely Revised, Extended And Updated Edition*, 2nd ed., ser. Springer Tracts in Advanced Robotics. Springer International Publishing, 2017, ISBN: 978-3-319-54412-0. DOI: [10.1007/978-3-319-54413-7](https://doi.org/10.1007/978-3-319-54413-7). [Online]. Available: <https://www.springer.com/gp/book/9783319544120> (visited on 12/20/2020).
- [63] N. H. Amer, H. Zamzuri, K. Hudha, and Z. A. Kadir, “Modelling and Control Strategies in Path Tracking Control for Autonomous Ground Vehicles: A Review of State of the Art and Challenges,” *Journal of Intelligent & Robotic Systems*, vol. 86, no. 2, pp. 225–254, May 2017, ISSN: 0921-0296, 1573-0409. DOI: [10.1007/s10846-016-0442-0](https://doi.org/10.1007/s10846-016-0442-0). [Online]. Available: <http://link.springer.com/10.1007/s10846-016-0442-0> (visited on 11/18/2020).
- [64] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, “Kinematic and dynamic vehicle models for autonomous driving control design,” in *2015 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2015, pp. 1094–1099. DOI: [10.1109/IVS.2015.7225830](https://doi.org/10.1109/IVS.2015.7225830).
- [65] F. Rosique, P. J. Navarro, C. Fernández, and A. Padilla, “A Systematic Review of Perception System and Simulators for Autonomous Vehicles Research,” *Sensors*, vol. 19, no. 3, p. 648, 3 Jan. 2019. DOI: [10.3390/s19030648](https://doi.org/10.3390/s19030648). [Online]. Available: <https://www.mdpi.com/1424-8220/19/3/648> (visited on 11/19/2020).
- [66] M. Althoff, “An Introduction to CORA 2015 (Tool Presentation),” p. 32,
- [67] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler, “SpaceEx: Scalable Verification of Hybrid Systems,” in *Computer Aided Verification*, G. Gopalakrishnan and S. Qadeer, Eds., ser. Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, 2011, pp. 379–395, ISBN: 978-3-642-22110-1. DOI: [10.1007/978-3-642-22110-1_30](https://doi.org/10.1007/978-3-642-22110-1_30).
- [68] X. Chen, E. Ábrahám, and S. Sankaranarayanan, “Flow*: An Analyzer for Non-linear Hybrid Systems,” in *Computer Aided Verification*, N. Sharygina and H. Veith, Eds., ser. Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, 2013, pp. 258–263, ISBN: 978-3-642-39799-8. DOI: [10.1007/978-3-642-39799-8_18](https://doi.org/10.1007/978-3-642-39799-8_18).
- [69] J. S. Rowlinson, “The Yukawa potential,” *Physica A: Statistical Mechanics and its Applications*, vol. 156, no. 1, pp. 15–34, Mar. 15, 1989, ISSN: 0378-4371. DOI: [10.1016/0378-4371\(89\)90108-8](https://doi.org/10.1016/0378-4371(89)90108-8). [Online]. Available: <http://www.sciencedirect.com/science/article/pii/0378437189901088> (visited on 11/27/2020).
- [70] J. Błaszczyk, A. Karbowski, and K. Malinowski, “Object Library of Algorithms for Dynamic Optimization Problems: Benchmarking SQP and Nonlinear Interior Point Methods,” *International Journal of Applied Mathematics and Computer Science*, vol. 17, no. 4, pp. 515–537, Dec. 1, 2007. DOI: [10.2478/v10006-007-0043-y](https://doi.org/10.2478/v10006-007-0043-y). [Online]. Available: <https://content.sciendo.com/view/journals/amcs/17/4/article-p515.xml> (visited on 11/21/2020).
- [71] G. Welch and G. Bishop, “An introduction to the kalman filter,” University of North Carolina at Chapel Hill, USA, 1995.
- [72] I. The MathWorks, *Model Predictive Control Toolbox*, manual, Natick, Massachusetts, United State, 2019. [Online]. Available: <https://www.mathworks.com/help/mpc/>.

- [73] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, “SciPy 1.0: Fundamental algorithms for scientific computing in python,” *Nature Methods*, vol. 17, pp. 261–272, 2020. doi: [10.1038/s41592-019-0686-2](https://doi.org/10.1038/s41592-019-0686-2).
- [74] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, “ROS: An open-source Robot Operating System,” p. 6,
- [75] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, “CARLA: An open urban driving simulator,” in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.
- [76] S. Gillies *et al.*, “Shapely: Manipulation and analysis of geometric objects,” 2007–. [Online]. Available: <https://github.com/Toblerity/Shapely>.

A Parameters Used

Simulation Parameters	Value	Unit	Comment
Vehicle model Parameters			
l_r	1.2525	m	
l_f	2.4545	m	
δ_{fmin}	-70	deg	
δ_{fmax}	70	deg	
a_{min}	-10	ms^{-2}	
a_{max}	5	ms^{-2}	
$jerk_{min}$	-0.9	ms^{-3}	
$jerk_{max}$	0.9	ms^{-3}	
$\delta_{f\dot{min}}$	-0.5	$rads^{-1}$	
$\delta_{f\dot{max}}$	0.5	$rads^{-1}$	
Risk map parameters			
D_{max}	2.0	m	Velocity Inflation
γ_v	0.05	—	Velocity Inflation
C_v	10.0	ms^{-1}	Velocity Inflation
$Yukawa_A_{car}$	10.0	—	Yukawa amplitude
$Yukawa_Alpha$	0.5	—	Yukawa scaling
Δ_x	0.5	m	Grid resolution
Grid Range x	[−20, 20]	m	Grid Size
Grid Range y	[−10, 10]	m	Grid Size
η_{road}	3	—	Road potential
$U_{threshold}$	8	—	
Reachability parameters			
Δt	0.1	s	time step
$T_{horizon}$	1.0	s	Reachability Horizon
Behavioural selection heuristics parameters			
d_{cruise}	5.0	m	
$d_{overtake}$	12.0	m	
$d_{overtake_over}$	2.0	m	
Obstacle Avoidance Parameters			
Obstacle Ellipse Order	6	—	Order of ellipse
Inflation factor	1.5	—	
NMPC parameters			
$w_x(1)$	10	—	weights(Penalty for x)
$w_x(2)$	10	—	weights(Penalty for y)
$w_x(3)$	5	—	weights(Penalty for ψ)
$w_x(4)$	5	—	weights(Penalty for v)
T_h	1.0	s	Planning horizon
NMPC Δt	0.1	s	
Prediction Horizon	10	—	$PredictionHorizon = T_h/\Delta t$
Δ_t	0.1	s	time step
PID gains			
$Throttlek_p$	0.15	—	
$Throttlek_i$	0.1	—	
$Throttlek_d$	0.0	—	
$Brakek_p$	1.0	—	
$Brakek_i$	0.0	—	
$Brakek_d$	0.0	—	

Table A1: Parameters used for simulation.