**IMT School for Advanced Studies, Lucca**
Lucca, Italy

# Learning-based Stochastic Model Predictive Control for Autonomous Driving

PhD Program in System Science

Track in Computer Science and Systems Engineering

XXXIII Cycle

**By**

**Surya Soman**

**2023**

**The dissertation of Surya Soman is approved.**

PhD Program Coordinator: Prof. Rocco De Nicola, IMT School for Advanced Studies Lucca

Advisor: Prof. Alberto Bemporad, IMT School for Advanced Studies Lucca

Co-Advisor: Prof. Mario Zanon, IMT School for Advanced Studies Lucca

The dissertation of Surya Soman has been reviewed by:

Reviewer1: Prof. Stefania Santini, University of Naples Federico II

Reviewer2: Dr. Rien Quirynen, Mitsubishi Electric Research Laboratories

IMT School for Advanced Studies Lucca
2023

To my Family, Friends, and Almighty

# Contents

# List of Figures

# List of Tables

# Acknowledgements

This endeavor wouldn't have been possible without Prof. Alberto Bemporad, who gave me an opportunity to pursue a Ph.D. That in turn helped me to get the career and life I always dreamed of. He has been of immense support by giving ideas and suggestions that mold this thesis. I would like to express my gratitude towards Prof.Mario Zanon, for his valuable opinions and constructive criticisms. It helped me to fine-tune the ideas as well as bring more insights to it. Moreover, his suggestions made me pay attention to my coding style. I would like to thank Prof. Stefania Santini and Rien Quirynen for taking the time to review my thesis on time and providing positive feedback that helped me improve this thesis.

I take this opportunity to thank my friend Raaja Ganapathy Subramaniam for encouraging and supporting me to pursue a PhD and introducing me to IMT and being there for me when I needed. I would also like to thank Prof Kanthalakshmi, who has been a supporting mentor.

I would like to thank my mother Vijaya Soman, who always believed in me and has been supportive throughout my life. My father G. Soman, has been a role model for me by being ambitious and thriving and who also encouraged me to pursue a doctorate. Sujin Soman, my brother have always been there by cheering me up and checking on me during difficult moments. I would like to thank my family Yesodarammal and Jayendran, whose prayers have been there for me. My late aunt Revammal, who was protective, caring towards me. I wish I was there with you when you were unwell and I'm sure your prayers will always be there.

Last but not least I take this opportunity to thank all my friends from IMT who made IMT and Lucca at home. Maria Rosaria Marulli, I am immensely thankful to have you as my friend, for being the person whom I can always count on and to help me with all the bureaucratic as well as health issues throughout these years. Zainab and her family (Mohammed and Ali), who have made me feel like family, helped and cheered even during their difficult times. I'm also thankful for baby Ali who always bring me flowers whenever he stop by to say hi. Afrodite, for being a therapist, for checking on me by screaming my name from the street and offering to take care of me while I'm done with defense, thank you it made me feel like I have someone to look after me. Elisa Bernad, being strict whenever its necessary to get me my things done and for all the dances we had during the parties. Zorica Savanovic, for being a sister, for inviting and taking care of me whenever I couldn't go back to India. Chiara Battaglini, Nicolina Bruno, Natalie Massong and Martina Berto, for all the girls trip and moments we had together in these past years. Pavan and Francis for being the caring Indian brothers in IMT. Sedric, for all the gym days and talks we shared. Victor, for being family as well as motivating by sharing bussiness ideas. Emiliano Marchese, for being a father like figure, protective and caring in the past years. Matteo Serafino, for pushing me to mingle with everyone when I was shy. Sara Belluccini, for being the best roommate. Ivan Merchanti, for all the IT support. Sam Ayele, for founding the International Student Union, during the loneliest period. Vihang Naik, for being the big brother in DYSCO and sharing the wisdom. Niraj, for all the dances, support in research as well as bureaucracy. Laura Ferrarotti, for all the help and company throughout these years. Asli, for giving a chance not only to visit Turkey but also to attend a Turkish wedding. Roberto Pizziol and Kristina for all the conversations and moments we had in this brief period. Anil, for

encouraging me to socialize again after the pandemic. Deison, for all the cheerful times we had together. Sara Landi and Nicole, for encouraging me as well attended all my Zumba lessons. Mriko Hu, for being a brother and for all the funny moments we had whenever we go somewhere together.

I would like to especially thank Sampath Kumar Mulagaleti, for being supportive not only as a flatmate and friend but as a colleague who always there to support when I was stuck in the research. Thank you for being patient and for spending time to discuss the issues.

Sara Olson, for all the help in the beginning of this PhD journey. Barbara Iacobino, for all the help throughout these years and being a go to person whether its PhD related or personal bureaucratic issue.

# Vita

| | |
|---|---|
| **August 03, 1991** | Born, Trivandrum, India |
| **2013** | Bachelor of Technology<br>Final mark: 7.34/10 gpa<br>PRS College of Engineering & Technology,<br>Kerala, India |
| **2016** | Master of Engineering<br>Final mark: 7.99/10 gpa<br>PSG College of Technology,<br>TamilNadu, India |
| **2017** | Programmer Analyst<br>Cognizant Technology Services<br>TamilNadu, India |
| **2021** | Software Engineer<br>Baker Hughes<br>Florence, Italy |
| **2023** | Control Software Engineer<br>Baker Hughes<br>Florence, Italy |

# Abstract

Autonomous driving in urban environments requires safe control policies that account for the non-determinism of moving obstacles, for instance, the intention of other vehicles while crossing an uncontrolled intersection. This thesis addresses the aforementioned problem by proposing a stochastic model predictive control (SMPC) approach. In this approach, we consider robust collision avoidance as a constraint to guarantee safety and a stochastic performance index that will increase the quality of the closed-loop tracking by ignoring the unlikely obstacle configurations that could occur. We compute the probabilities associated with different obstacle trajectories by training a classifier on a realistic dataset generated by the microscopic traffic simulator SUMO and show the benefits of the proposed stochastic MPC formulation in a simulated real intersection. This thesis is divided into two parts: first, discuss the formulation of the existing control algorithm and our proposed approach, and second, the scenario prediction of the obstacle vehicles.

# Chapter 1

# Introduction

## 1.1 Background and Motivation

Autonomous driving is one of the areas that has gone through a lot of evolution in the last few decades, still achieving fully automated reliable driving is expected to require further significant research efforts [1, 2, 3]. The timeline of autonomous vehicles started in 1926 with the introduction of radio controlled car named 'Linriccan Wonder'. In the 1980s a vision-guided driverless Mercedes Benz van achieved 63km/hr speed on a road without traffic. The actual start of autonomous driving technology started with the development of the driverless autonomous car by the Defense Advanced Research Projects Agency (DARPA). It is the arm of the USA Department of Defence that is responsible for advancing military technology and was financially supporting this research in the name of the DARPA Grand Challenge. When it comes to commercializing this technology Google was one of the first companies that took the initiative.

According to the Society of Automotive Engineers **(SAE)** International, there are six levels of automation.

- **Level 0** : No Automation
- **Level 1** : With few autonomous features such as automatic braking

**Figure 1:** Mercedes's Drive Pilot

but need human intervention at all the time

- **Level 2** : It will take safety actions, still the driver needs to be alert at the steering wheels and pedals of the car. The features include cruise control/autopilot and lane centering
- **Level 3** : In this, the car can handle dynamic driving which means it can do the steering, braking, and lane changing but still needs a driver when it signals for intervention
- **Level 4** : The level that needs less human intervention, as it drives safely without human input except in unmapped area as well as during bad weather
- **Level 5** : It doesn't have a brake pedal or steering wheel. This type of car can drive in all conditions and doesn't need a driver at all

We currently have Mercedes's Drive Pilot as the world's first certified Level 3 Automation system, which is already available in Germany.

The architecture of self-driving vehicles [4] is mainly divided into two parts 1) Perception and 2) Decision making as shown in Figure 2.

1. **Perception** This system is responsible for estimating the state of the ego as well as creating a representation of the surroundings with

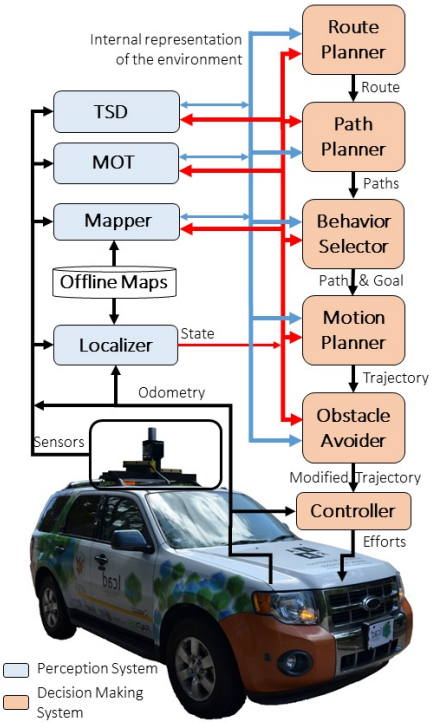


**Figure 2:** Architecture of a typical Self-driving vehicle. TSD is Traffic Signalization Detection and MOT, Moving Objects Tracking [4]

the help of data from the onboard sensors such as Light Detection and Radar (LIDAR), Radio Detection and Ranging (RADAR), camera, Global Positioning System (GPS), Inertial Measurement Unit (IMU), odometer, etc and the prior knowledge of sensor models, traffic rules, road networks, etc.

2. **Decision making** It is responsible for navigating the vehicle from starting to the target position defined by the user taking into account the input from the perception system.

In this thesis, we focus on the Obstacle Avoider and Controller part within the Decision-making system.

## 1.2 Challenges and Existing Solutions

As fascinating as it is the concept of higher-level automation (Above level 2), is as challenging as it gets. One of the main challenges is navigating through an uncertain environment that includes pedestrians, other non-automated obstacle vehicles, etc. Various types of sensors are nowadays able to provide reliable information about the *current* obstacle positions, see, e.g., [5]. However, in order to drive safely and effectively, the control algorithm needs to take *future* obstacle positions into account. This poses two challenges: on the one hand how to obtain such information; and on the other hand, how to exploit it in order to issue safe control commands.

Several approaches have been proposed for modeling pedestrians and surrounding vehicles; see, e.g., the overview in [6]. In particular, interacting multiple-model (IMM) filters that predict the intention and future states of surrounding vehicles were suggested in [7, 8].

A similar approach was taken in [9], where, in addition, hidden Markov models (HMMs) were used to recognize vehicle maneuvers and probabilistic trajectories generated with the help of variational Gaussian mixture models.

A multimodal hierarchical Inverse Reinforcement Learning (IRL) approach was instead in [10] to learn joint driving pattern-intention-motion models and use them to probabilistically predict continuous motions. A simpler but effective and computationally inexpensive approach that retains the ability to predict multimodal distributions was proposed in [11]. In [12], the authors proposed a general semantic-based intention and motion prediction based on deep neural networks; similar approaches were also proposed in [13] and [14] to predict lane change maneuvers. A human-like decision model for unsignalized intersection was suggested in [15] by an intention-aware prediction of other vehicles via convolutional neural networks with multiple object tracking combined with a Kalman filter. In [16], recurrent neural networks in long short-term mem-

ory (LSTM) form was used to predict the future driving lane of the vehicle. A hybrid approach using a neural classifier for maneuver classification and an LTSM memory for trajectory prediction was analyzed in [17]. For lane-change maneuver prediction, a combination of support vector machines and neural networks was considered in [18], while random forests and conditional random fields method were suggested in [19] for a T-intersection.

Assuming that a model estimating the future positions of the surrounding obstacles is available, this can be naturally exploited for planning the motion of the ego vehicle by using model predictive control (MPC) techniques [20, 21], as collision avoidance can then be formulated as an explicit constraint. Indeed, MPC has gained considerable attention in the last years in automotive control, especially for its ability to handle constraints on system variables, not only in academic research but also in industrial practice; see, e.g., [22] for a documented use of MPC in high-volume production. Regarding motion planning, MPC enables tracking reference paths at desired speeds while ensuring collision avoidance, thanks to the introduction of explicit constraints that keep the predicted distance between the ego vehicle and obstacles above a prescribed safety margin; see, e.g., [23], in which an MPC formulation was used for path planning in a dynamic environment by modeling the surrounding obstacle vehicles as polygons, and [24] for an MPC formulation using spatial-based models.

To handle uncertainty in the prediction, *stochastic* MPC formulations [25] can be introduced, in particular, scenario-based approaches [26]; see, e.g., [27, 28, 29] for applications of stochastic MPC in automotive control. In [30], the authors proposed the use of stochastic MPC based on Gaussian mixture models to get a multimodal prediction of the trajectories of the surrounding vehicles. MPC approaches tailored to collision avoidance with pedestrians were proposed in [31, 32], while a generic MPC framework providing rigorous safety guarantees in uncertain environments was analyzed in [33, 34].

Alternative approaches for safe path planning were proposed in [35, 36]. A stochastic scenario-based MPC approach was adopted in [37] by

relying on partially observable Markov decision process models. More recently, [38] employed what the authors called *branch MPC*, which uses ideas similar to [32], and leverages neuroscience studies to model human decision-making and predict the obstacle vehicle intentions.

## 1.3 Scope and Contribution of the thesis

In this thesis, we endeavor to handle the uncertain obstacles in autonomous driving by considering approaches that are best suitable for each intention prediction of the obstacles and planning of the respective trajectory of the ego. First, we employ i) a stochastic MPC formulation with robust collision-avoidance constraints for commanding the longitudinal acceleration and steering rate of the ego vehicle, focusing on the case of uncontrolled intersections ii) Second, a classifier that is trained on a realistic dataset generated by the microscopic traffic simulator SUMO (Simulation of Urban Mobility) [39] estimates the probabilities associated with different future scenarios, which are crucial for closed-loop performance and predicts the intention of incoming vehicles to drive straight ahead, turn left, or turn right. iii) Third, using a realistic simulation setting, we compare the proposed approach against alternative MPC formulations, namely deterministic prescient MPC, in which future obstacle positions are known exactly, and robust MPC, in which probabilities are not taken into account.

## 1.4 Structure

**Chapter 2 - Problem Definition**

This chapter elaborates on the problem we address in this thesis as well as the overview of the vehicle model. We formulate the cost function and constraints we are trying to optimize. And we briefly explain the keywords that might be useful to understand the problem formulation.

**Chapter 3 - Model Predictive control algorithms**

This chapter introduces several MPC approaches that can be used to solve the problem we defined in Chapter 2. Firstly, we introduce the framework for the ideal case where the exact obstacle waypoints are known prior and which is further called the prescient model predictive control (PMPC). The other two approaches give a control algorithm that takes into account uncertainty. The first approach is robust MPC. We elaborate on the framework and its shortcomings. The second approach is the stochastic MPC which is the base of our control approach. This chapter gives a detailed insight into our proposed scenario-based stochastic model predictive control and the corresponding problem formulation.

**Chapter 4 - Obstacle Model and Scenario Prediction**

This chapter gives a background on different learning methods and the relevant basic information that is necessary for the reader. It focuses mainly on supervised learning to have an understanding of the classifiers that we have used in this thesis. It elaborates on the features chosen for the classifier and its comparison with other classifiers as well as the characteristics of the chosen classifier.

**Chapter 5 - Simulation of Urban Mobility**

This chapter gives an overview of the SUMO Traffic simulator and a brief glimpse into Traci the interface which facilitated the connection of MatLab and SUMO. It describes in detail the type of vehicles and other relevant driving parameters which we have used for simulating several obstacle scenarios to train and test the classifier.

**Chapter 6 - Simulation Results**

This chapter gives the simulation results of several ego-obstacle interaction scenarios and compares our control algorithm output with Robust based and Prescient Model Predictive control.

**Chapter 7 - Conclusion & Future work**

This chapter elaborates conclusions we have reached in this thesis and the research and work that needs to be continued.

# Chapter 2

# Problem Definition

As we have addressed in the previous chapter, we focus on predicting the intentions of obstacle vehicles and implement a control algorithm that takes into account the predicted intentions to navigate the ego vehicle along the given reference path by avoiding possible collisions.

In this chapter, a detailed description of the ego vehicle model as well as the constraints associated with the control problem formulation are discussed.

## 2.1 Mathematical preliminaries

**4-step Runge Kutta method**

To solve an ordinary differential equation (ODE) there are several methods available. In this thesis, we considered Runge-Kutta as the method to solve an ODE.

Consider a first order ordinary differential equation

$$\frac{dy(t)}{dt} = f(y(t), t), y(t_0) = y_0, \tag{2.1}$$

where, $y$ is the unknown function of time $t$ that requires to be approximated, the function $f$ and the initial values $t_0$ and $y_0$ are given [40].

Consider a step size $h > 0$ and define:

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)h, \tag{2.2a}$$

$$t_{n+1} = t_n + h \tag{2.2b}$$

$$\text{for } n = 0, 1, \ldots \tag{2.2c}$$

$$\text{where,} \tag{2.2d}$$

$$k_1 = f(t_n, y_n), \tag{2.2e}$$

$$k_2 = f(t_n + \frac{h}{2}, y_n + h\frac{k_1}{2}), \tag{2.2f}$$

$$k_3 = f(t_n + \frac{h}{2}, y_n + h\frac{k_2}{2}), \tag{2.2g}$$

$$k_4 = f(t_n + h, y_n + hk_3). \tag{2.2h}$$

Where, $y_{n+1}$ is the RK4 approximation of $y(t_{n+1})$ and $k_1, k_2, k_3, k_4$ are the slopes computed at the beginning, midpoint(both $k_2$ and $k_3$) and end of the interval respectively.

**Optimization Problem**

An optimization problem [41] is generally formulated as

$$inf_z \quad f(z) \tag{2.3a}$$

$$subj.to \quad z \in S \subseteq Z, \tag{2.3b}$$

where the vector $z$ collects the decision variables, $Z$ is the optimization problem domain, and $S \subseteq Z$ is the set of feasible or admissible decisions. The function $f : Z \to \mathbb{R}$ assigns to each decision $z$ a cost $f(z) \in \mathbb{R}$. Throughout this section, we will consider a shorter version of equation (2.3) given by in equation (2.4).

$$\inf_{z \in S \subseteq Z} f(z). \tag{2.4}$$

Solving equation (2.4) means to compute least possible cost $f^*$ which is given by:-

$$f^* = \inf_{z \in S \subseteq Z} f(z).$$

The value $f^*$ is the optimal value of the equation (2.4), i.e.,

$$f(z) \geq f(z^*) = f^* \; \forall \; z \in S, \exists \; z^* \in S$$

If $f^* = -\infty$ we say the problem is unbounded below. If the set $S$ is empty then the problem is *infeasible* and set $f^* = +\infty$ by convention. If $S = Z$ the problem is *unconstrained*. To find the optimal solution, that is to find $z^* \in S$ with $f(z^*) = f^*$. If such $z^*$ exists, then we rewrite problem (2.4) as (2.5)

$$f^* = \min_{z \in S} f(z) \tag{2.5}$$

and $z^*$ is called an optimizer, global optimizer or optimal solution. The set of all optimal solutions are referred by

$\arg\min_{z \in S} f(z) = \{z \in S : f(z) = f^*\}$.

A problem of determining whether the set of feasible decisions is empty and, if not, to find a point that is feasible, is called a *feasibility problem*.

**Convexity**

A set $S \in \mathbb{R}^s$ is *convex* if

$\lambda z_1 + (1 - \lambda) z_2 \in S$ for all $z_1 \in S, z_2 \in S$ and $\lambda \in [0, 1]$.

A function $f : S \to \mathbb{R}$ is convex if $S$ is convex and

$f(\lambda z_1 + (1 - \lambda) z_2 \leq \lambda f(z_1) + (1 - \lambda) f(z_2)$ for all $z_1 \in S, z_2 \in S$ and $\lambda \in [0, 1]$.

A function $f : S \to \mathbb{R}$ is *strictly convex* if $S$ is convex and $f(\lambda z_1 + (1-\lambda) z_2 < \lambda f(z_1) + (1-\lambda) f(z_2)$ for all $z_1 \in S, z_2 \in S$ and $\lambda \in (0, 1)$.

A twice differentiable function $f : S \to \mathbb{R}$ is *strongly convex* if the Hessian $\nabla_2 f(z) > 0$ for all $z \in S$.

A function $f : S \to \mathbb{R}$ is concave if $S$ is convex and $-f$ is convex.

**Convex Optimization Problem**

A standard optimization is given by equation (2.6)

$$\begin{aligned} inf_z \quad & f(z) && \text{(2.6a)} \\ subj.to \quad & g_i(z) \leq 0 \quad \text{for } i = 1, \ldots, m && \text{(2.6b)} \\ & h_j(z) = 0 \quad \text{for } j = 1, \ldots, p && \text{(2.6c)} \\ & z \in Z, && \text{(2.6d)} \end{aligned}$$

where $f, g_1 \ldots, g_m, h_1, \ldots, h_p$ are real-valued functions defined over $\mathbb{R}^s$, i.e., $f : \mathbb{R}^s \to \mathbb{R}, g_i : \mathbb{R}^s \to \mathbb{R}, h_i : \mathbb{R}^s \to \mathbb{R}$. The domain $Z$ is the intersection of the domains of the cost and constraint functions:

$$Z = \{z \in \mathbb{R}^s : z \in \operatorname{dom} f, z \in \operatorname{dom} g_i, i = 1, \ldots, m, z \in \operatorname{dom} h_j, j = 1, \ldots, p\}. \tag{2.7}$$

The problem (2.6) is unconstrained if $m = p = 0$. The inequalities $g_i(z) \leq 0$ are called *inequality constraints* and the equations $h_i(z) = 0$ are called *equality constraints*. A point $\bar{z} \in \mathbb{R}^s$ is *feasible* for problem (2.6) if: (i) it belongs to $Z$, (ii) it satisfies all inequality and equality constraints, i.e., $g_i(\bar{z}) \leq 0, i = 1, \ldots, m, h_j(\bar{z}) = 0, j = 1, \ldots, p$. The set of feasible vectors is

$$S = \{z \in \mathbb{R}^s : z \in Z, g_i(z) \leq 0, i = 1, \ldots, m, h_j(z) = 0, j = 1, \ldots, p\}. \tag{2.8}$$

Let $f^*$ be the optimal value of the problem (2.6). An optimizer, if it exists, is a feasible vector $z^*$ with $f(z^*) = f^*$.

A feasible point $\bar{z}$ is locally optimal for problem (2.6) if there exists an $R > 0$ such that

$$
\begin{aligned}
f(\bar{z}) = inf_z \quad & f(z) && \text{(2.9a)}\\
subj.to \quad & g_i(z) \leq 0 \quad \text{for } i = 1, \ldots, m && \text{(2.9b)}\\
& h_j(z) = 0 \quad \text{for } j = 1, \ldots, p && \text{(2.9c)}\\
& \|z - \bar{z}\| \leq R && \text{(2.9d)}\\
& z \in Z, && \text{(2.9e)}
\end{aligned}
$$

Where, $\bar{z}$ is the minimizer of $f(z)$ in a feasible neighborhood of $\bar{z}$ defined by $\|z - \bar{z}\| \leq R$. The point $\bar{z}$ is called a local optimizer or local minimizer.

The standard optimization problem (2.6) is said to be convex if the cost function $f$ is convex on $Z$ and $S$ is a convex set. A fundamental property of convex optimization problems is that local optimizers are also global optimizers.

It is difficult to determine whether the feasible set $S$ of the optimization problem (2.6) is convex or not except in special cases. For instance, if the functions $g_1(z), \ldots, g_m(z)$ are convex and all the $h_i(z)$ (if any) are affine in $z$, then the feasible set $S$ in (2.8) is an intersection of convex sets and is therefore convex. Moreover, there are nonconvex problems that can be transformed into convex problems through a change of variables and manipulations of cost and constraints.

**Polytope and Polyhedron**

A Polyhedron $P$ in $\mathbb{R}^n$ denotes the intersection of a finite set of closed half spaces in $\mathbb{R}^n$:

$$P = \{x \in \mathbb{R}^n : Ax \leq b\},$$

where $Ax \leq b$ is a system of inequalities, namely $a_i^{'}x \leq b_i, i = 1, \ldots, m$, where $a_1^{'}, \ldots, a_m^{'}$ are the rows of $A$, and $b_1, \ldots, b_m$ are the components of $b$.

A Polytope is a bounded polyhedron as shown in Figure3.

A linear inequality $c^{'}z \leq c_0$ is said to be valid for $P$ if it is satisfied for all points $z \in P$. A face of $P$ is any nonempty set of the form

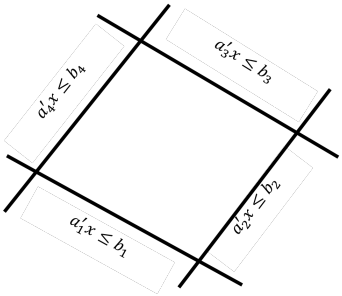


**Figure 3:** Example of a Polytope

$$F = P \cap \{z \in \mathbb{R}^s : c^{'} z \leq c_0\},$$

where $c^{'} z \leq c_0$ is a valid inequality for P i.e., this inequality holds for all points in $P$. All faces of $P$ satisfying $F \cap P$ are called proper faces and have dimension less than $\dim(P)$. The faces of dimension $0, 1, \dim(P)-2$ and $\dim(P) - 1$ are vertices, edges, ridges and facets, respectively.

## 2.2 Kinematic Bicycle model

To model the lateral motion of the ego vehicle we consider the kinematic bicycle model as shown in figure 4 with the front axle of the vehicle as the desired point which assumes to have a planar motion [42]. In a bicycle model, the two left and right front wheels are represented by A and similarly, B represents both the rear wheels as one. The corresponding model is given by the equation (2.10).

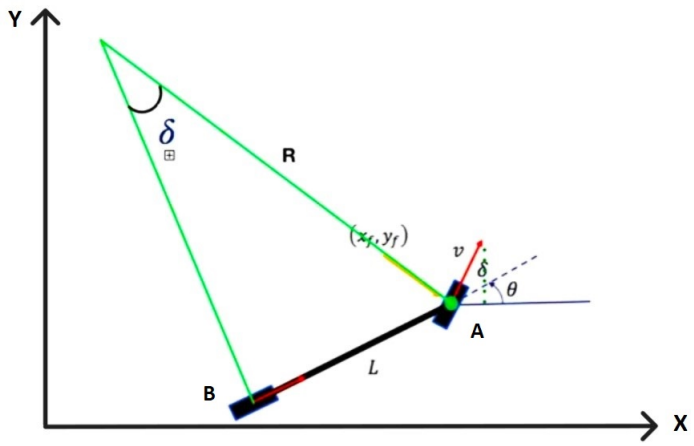


**Figure 4:** Kinematic Bicycle model (2.10)

$$\dot{x} = v\cos(\theta + \delta) \tag{2.10a}$$

$$\dot{y} = v\sin(\theta + \delta) \tag{2.10b}$$

$$\dot{\theta} = \frac{v}{L}\sin(\delta) \tag{2.10c}$$

$$\dot{v} = a \tag{2.10d}$$

$$\dot{\delta} = \omega \tag{2.10e}$$

where $(x, y)$ are the coordinates of the location of the center of the front axle of the ego vehicle in the fixed absolute frame and $\theta$ its orientation with respect to global $x$ axis, $v$ the longitudinal speed, and $\delta$ the steering angle. Throughout this thesis, we denote the state vector of the model as $X = [x\ y\ \theta\ v\ \delta]'$ and $U = [a\ \omega]'$ as the input vector, where $a$ is the longitudinal acceleration and $\omega$ the steering rate.

The prediction model for MPC is obtained by discretizing (2.10) using an explicit Runge-Kutta 4 method with time-discretization $T_s$, ob-

taining the following discrete-time nonlinear model

$$X_{t+1} = f(X_t, U_t) \tag{2.11}$$

where $t$ denotes the sample step.

## 2.3 Problem Definition

Our challenge is to decide which mode of input can be obtained from the obstacle intention prediction model and how to incorporate it into a control algorithm. In this section, we describe a high-level problem formulation.

To formulate a path-following problem with dynamic collision avoidance the crucial component is the position $(x^o, y^o)$ of the dynamic obstacle. Hence we need a formulation that integrates this uncertain element. The collision avoidance can be considered as a constraint to our optimization problem where the Euclidean distance between the position $(x, y)$ of the ego vehicle and the obstacle vehicle should be greater than the minimum collision distance $d_{\min}$ as described in equation (2.12). As for the cost function, we need to penalize the state $X$ and the input $U$ of the ego vehicle with respect to the given reference $X_r$ and $U_r$ as in equation (2.13). Thus we obtain a nonlinear non-convex optimization problem considering also the vehicle model.

$$(x - x^o)^2 + (y - y^o)^2 \geq d_{\min}^2 \tag{2.12}$$

$$\|X - X^r\|^2 + \|U - U^r\|^2 \tag{2.13}$$

To obtain the obstacle position $(x^o(t), y^o(t))$ at each time instant $t$ we need to assign a model or trajectory for each possible scenario it might take. And a prediction model to acquire the possible intentions of the same. For the implementation of the intention prediction model of the obstacle vehicles, we rely on supervised learning as a specific machine learning technique. A detailed description of the method is explained in Chapter 4. There are several approaches to include the uncertainty in the

obstacle position $(x^o(t), y^o(t))$ at each time instant $t$ in the problem and it is described in detail in the upcoming Chapter 3.

Apart from the vehicle model we need to define the road bounds to find a realistic control input for our problem. Here we consider a polytope with reference waypoints of ego vehicle $(x^r(t), y^r(t), \theta^r(t))$ as the center at each time instant $t$. And with a width, $W_r$ and length $L_r$ that depends on the road profile and are chosen small enough to guarantee that the ego vehicle remains within the road. Thus we get the equation (2.14).

$$A_t(X_t - X_t^r) \leq B_t \tag{2.14}$$

where,

$$A_t = [I_2 \ -I_2]' \begin{bmatrix} \cos(\theta_t^r) & \sin(\theta_t^r) \\ -\sin(\theta_t^r) & \cos(\theta_t^r) \end{bmatrix},$$

where $\theta_t^r$ is the given reference orientation of ego vehicle wrt global axis for each time instant $t$ and $I_2$ is the identity matrix of order 2, and $B_t = \left[\frac{L_r}{2} \ \frac{W_r}{2} \ \frac{L_r}{2} \ \frac{W_r}{2}\right]'$.

## 2.4 Conclusions

In this chapter, we have addressed the control problem we are trying to solve in this thesis. In addition, we described the need for obstacle intention prediction and the association of obstacle parameters in the control problem of the ego vehicle.

# Chapter 3

# Model Predictive Control Algorithms

In this chapter, we emphasize the controller design for our problem. We define an ideal MPC framework to act as a baseline and we further introduce the robust and stochastic MPC formulations that are capable of directly dealing with the uncertainty of obstacle predictions.

## 3.1 Prescient Model Predictive Control

In the ideal case, the obstacle vehicle position at each time instant $t$ is known prior, and we further refer to the resulting MPC framework as Prescient Model predictive control (PMPC). At time $t$ the corresponding PMPC formulation is given as follows.

$$\min_{\substack{\{X_{k|t},U_{k|t}\}\\ k=0,\dots,N}} \quad \sum_{k=0}^{N} \|X_{k|t} - X^r_{k|t}\|^2_Q + \|U_{k|t} - U^r_{k|t}\|^2_R \tag{3.1a}$$

$$\text{s.t.} \quad X_{k+1|t} = f(X_{k|t}, U_{k|t}) \tag{3.1b}$$

$$X_{0|t} = X_t \tag{3.1c}$$

$$X_{k|t} \in \mathcal{X} \tag{3.1d}$$

$$U_{k|t} \in \mathcal{U} \tag{3.1e}$$

$$A_{k|t}(X_{k|t} - X^{\mathrm{r}}_{k|t}) \leq B_{k|t} \tag{3.1f}$$

$$(x_{k|t} - x^{\mathrm{o}}_{k|t})^2 + (y_{k|t} - y^{\mathrm{o}}_{k|t})^2 \geq d^2_{\min}. \tag{3.1g}$$

In (3.1), we assume that at the current time $t$ a measurement or an estimate of the state $X_t$ is available, $(x^{\mathrm{o}}_{k|t}, y^{\mathrm{o}}_{k|t})$ denotes the known obstacle trajectory (we will relax this assumption later). The given reference samples are $(X^{\mathrm{r}}_{k|t}, U^{\mathrm{r}}_{k|t})$, $k = 0, 1, \dots, N$, and $\|x\|^2_Q = x'Qx$, where matrix $Q \in \mathbb{R}^{5\times5}$ is symmetric and positive semi-definite, while matrix $R \in \mathbb{R}^{2\times2}$ is symmetric and positive definite. The sets $\mathcal{X}, \mathcal{U}$ are the sets of feasible state and input vectors, respectively.

The equation (3.1f) and (3.1g) are the road and obstacle avoidance constraints introduced in chapter 2 and are given by (2.14) and (2.12) respectively.

Problem (3.1) is solved at each time step $t$. The optimal solution's first component $U^{\mathrm{PMPC}}_{0|t}$ is commanded as the input $U_t$ to the vehicle and the remaining components $U^{\mathrm{PMPC}}_{1|t}, \dots, U^{\mathrm{PMPC}}_{N|t}$ are discarded using receding horizon strategy, and the problem is solved again at time $t+1$, and so on.

### 3.1.1 Conclusion

The ideal control framework (3.1) cannot be applied in practice, as it requires the knowledge of the future obstacle positions $(x^o_{k|t}, y^o_{k|t})$, which are not exactly known a priori. Nonetheless, we will consider (3.1) as a baseline policy for comparison with other control strategies.

## 3.2 Robust Model Predictive Control

While the previous problem formulation (3.1) assumes the exact knowledge of obstacle vehicle trajectory $(x^{\mathrm{o}}_{k|t}, y^{\mathrm{o}}_{k|t})$ in order to model the obstacle avoidance constraint, verifying this assumption might be unrealistic in practice. A sensible way to tackle this issue is to enumerate all possible scenarios that can be encountered by explicitly modeling all the possible future trajectories of the obstacle. These trajectories are obtained by simulating the obstacle dynamics and represent the set of all the positions that are reachable by the obstacle. Then, Problem (3.1) must 'tighten' the feasible region by all possible obstacle reachable positions. This mechanism of tightening the constraint set is a common approach in robust MPC [43, 44, 45, 46]. Unlike the standard setting of robust MPC, we encounter uncertainty in the constraints rather than in the dynamics. A similar setting has been studied previously in [31, 33, 34], in which guarantees for the recursive feasibility of MPC in the presence of unknown constraints were provided.

### 3.2.1 Problem Formulation

We consider the following formulation to take into account the uncertainty associated with future obstacle positions,

$$\min_{\substack{\{X_{k|t},U_{k|t}\}\\ k=0,\dots,N}} \quad \sum_{k=0}^{N} \|X_{k|t} - X^r_{k|t}\|^2_Q + \|U_{k|t} - U^r_{k|t}\|^2_R \tag{3.2a}$$

$$\text{s.t.} \quad X_{k+1|t} = f(X_{k|t}, U_{k|t}) \tag{3.2b}$$

$$X_{0|t} = X_t \tag{3.2c}$$

$$X_{k|t} \in \mathcal{X} \tag{3.2d}$$

$$U_{k|t} \in \mathcal{U} \tag{3.2e}$$

$$A_{k|t}(X_{k|t} - X^r_{k|t}) \leq B_{k|t} \tag{3.2f}$$

$$(x_{k|t} - x^{o_i}_{k|t})^2 + (y_{k|t} - y^{o_i}_{k|t})^2 \geq d^2_{\min}$$
$$\forall i = 1, \dots, m_t \tag{3.2g}$$

where $(x^{o_i}_{k|t}, y^{o_i}_{k|t})$, $i = 1, \ldots, m_t$, are meaningful corner-case scenarios used to account for the set of all possible trajectories taken by an obstacle traveling over the future horizon. The formulation (3.2) is robust in that collisions are avoided in all possible scenarios; in fact, the only difference between (3.1) and (3.2) is in the obstacle avoidance constraint (3.2g). Note that $m_t$ is the number of corner cases considered, which generally depends on time $t$, i.e., at time $t$ if the ego vehicle is performing a lane change in the highway the corresponding corner cases $m_t$ won't be the same as the corner cases it would be while crossing an intersection.

### 3.2.2 Conclusions

In this section, we have introduced a robust approach for overcoming the drawbacks in PMPC and elaborated the formulation for a single obstacle problem. The shortcoming of this approach is that it can lead to a conservative output and has a chance to obtain an infeasible solution since we are considering all possible obstacle scenarios simultaneously.

## 3.3 Stochastic Model Predictive Control

The main drawback of (3.2) is the possibility of conservativeness, due to the fact that all the corner cases are considered equally likely. In this section, we will elaborate on an approach that mitigates this drawback of the previously discussed control approach.

For real-world systems, uncertainties in obstacle vehicle $(x^{o}_{k|t}, y^{o}_{k|t})$ can be modeled effectively in a probabilistic setting. For instance, the event of an obstacle vehicle slowing down or stopping on a high-speed lane is of low probability. On the hand, the event might occur with high probability on a low-speed lane. Hence we move to a stochastic approach [47, 48] by taking into account the probabilities of all the possible intentions of the obstacle vehicle. We obtain these probabilities from the classifier which we will discuss in the upcoming chapter 4. There are several ways of formulating a stochastic MPC which is discussed in [49]. In the chance-constrained method [50, 51], the cost function is considered

as a random variable, the expected value of which is minimized, and probabilistic guarantees are provided with respect to constraint satisfaction. Alternatively in a scenario-based [52, 53] approach, the expectation of the cost function considering all scenarios is computed. The latter can become computationally challenging with the increase in the number of scenarios. Hence, to eliminate the irrelevant scenarios, a scenario tree is introduced which is obtained from the historical data on the uncertainty. In this thesis, we adopt the scenario-based stochastic approach to reduce the conservativeness in problem (3.2).

### 3.3.1 Scenario-based Stochastic MPC

In order to tackle the issue in the robust approach, let us associate a probability $\omega_t^i$ to each scenario $i$, $\sum_{i=0}^{m_t} \omega_t^i = 1, \omega_t^i \geq 0, \forall t \geq 0$.

#### Scenario-tree generation

We assume that the scenarios are organized as a scenario tree, see the example depicted in Figure 5. Each different branch of the tree is parameterized with a corresponding command input to be optimized and the number of leaves of the tree is equal to the number $m_t$ of considered corner cases. The time instants over the prediction horizon at which a split in sub-scenarios occurs are those at which one will be able to recognize different maneuvers taken by the obstacle, e.g., go straight or go left. We will explain the construction of the scenario tree in detail in the next chapter 4.

As multiple control moves can occur at the same prediction instant, the state prediction is no longer unique, as in the prescient and robust MPC cases. In fact, we associate the state trajectory $\{X_{k|t}^i\}$ to scenario $i$ and let $\{U_{k|t}^i\}$ denote the corresponding sequence of optimal control inputs. To take into account the tree structure, i.e., that scenarios $i$ and $j$ have a common subpath

$$(x_{k|t}^{o_i}, y_{k|t}^{o_i}) = (x_{k|t}^{o_j}, y_{k|t}^{o_j}),\ k = 0, \ldots, k_{ij}$$

we impose the equality constraints (a.k.a. causality constraints) $U_{k|t}^i =$

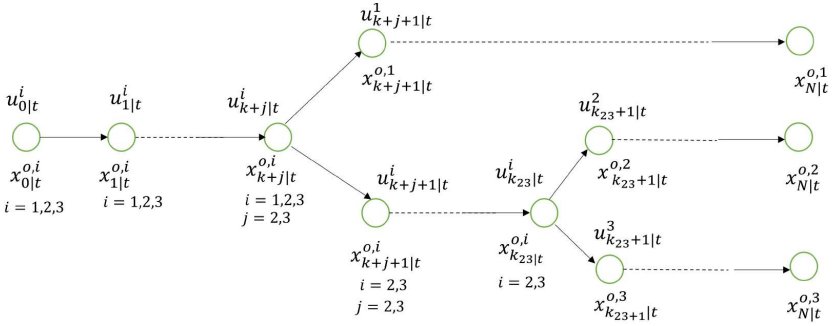


**Figure 5:** Scenario tree used in the SMPC single obstacle case formulation (3.3)

$U^j_{k|t}$ for all $k = 0, \ldots, k_{ij}\ \forall\ i, j = 1, \ldots, m_t$ and $i \neq j$. Note that all trajectories originate from the current state, i.e., $X^i_{0|t} \equiv X_t$.

### 3.3.2 Problem Formulation

Following the probabilistic scenario tree mentioned above, we formulate the following **Stochastic MPC** problem

$$\min_{\substack{\{X^i_{k|t}, U^i_{k|t}\} \\ k=0,\ldots,N \\ i=1,\ldots,m_t}} \quad \sum_{i=1}^{m_t} \omega^i_t \sum_{k=0}^{N} \|X^i_{k|t} - X^r_{k|t}\|^2_Q + \|U^i_{k|t} - U^r_{k|t}\|^2_R \tag{3.3a}$$

$$\text{s.t.} \quad X^i_{k+1|t} = f(X^i_{k|t}, U^i_{k|t}) \tag{3.3b}$$

$$X^i_{0|t} = X_t \tag{3.3c}$$

$$X^i_{k|t} \in \mathcal{X} \tag{3.3d}$$

$$U^i_{k|t} \in \mathcal{U} \tag{3.3e}$$

$$A^i_{k|t}(X^i_{k|t} - X^r_{k|t}) \leq B^i_{k|t} \tag{3.3f}$$

$$(x^i_{k|t} - x^{o_i}_{k|t})^2 + (y^i_{k|t} - y^{o_i}_{k|t})^2 \geq d^2_{\min}$$
$$\forall i = 1, \ldots, m_t \tag{3.3g}$$

$$U^i_{k|t} = U^j_{k|t},\ k = 0, \ldots, k_{ij} \tag{3.3h}$$

$$\forall\ i, j = 1, \ldots, m_t \text{ and } i \neq j \tag{3.3i}$$

Where $\omega_t^i$ will be obtained at each time $t$ from the classifier and the scenario tree will lift the obstacle avoidance constraints associated with unnecessary scenarios in the equation (3.3) once they are distinguishable.

### 3.3.3 Conclusions

An obvious reason why the SMPC formulation (3.3) is less conservative than the RMPC formulation (3.2) is that it has more degrees of freedom, as it can employ different input values in different scenarios. This implicitly defines a closed-loop optimal policy over the prediction horizon rather than a single open-loop optimal trajectory.

# Chapter 4

# Obstacle Model and Scenario Prediction

Our goal is to obtain a control law that can drive the ego vehicle autonomously through uncontrolled intersections. To use the MPC approaches described in the previous chapters, we need to define uncertain scenarios (3.3) and (3.2) and also the corresponding probabilities in the case of SMPC.

In this thesis, we have considered three possible scenarios that the obstacle vehicle might take which are straight, right, and left turn. And predict those scenarios with the help of a supervised learning classifier approach.

## 4.1 Machine Learning

In [54] machine learning is defined as solving a practical problem by 1) gathering a dataset, and 2) algorithmically building a statistical model based on that dataset. That statistical model is assumed to be used somehow to solve the practical problem.

Machine learning is mainly classified as supervised, unsupervised, and reinforcement learning.

### Supervised Learning

In supervised learning, the dataset is a collection of labeled examples $\{(x_i, y_i)\}_{i=1}^{N}$. Each $x_i$ within the vector of length $N$ is called a feature vector. A feature vector is a vector of dimension $D$,i.e., $j = 1, \ldots, D$, each $x_i^j$ is a value that describes the example, and this value $x_i^j$ is called a feature. The label $y_i$ indicates the class to which each feature vector $x_i$ belongs. A label can be a set of finite real numbers or an index selecting one of the multiple classes $1, \ldots, C$.

The goal of a supervised learning algorithm is to use the dataset to produce a model that takes a feature vector $x$ as input and outputs information that allows deducing the label for this feature vector.

Some of the supervised learning algorithms are decision trees [55], support vector machine (SVM) [56], and artificial neural network [57].

### Unsupervised Learning

In unsupervised learning, the dataset is a collection of unlabeled examples $\{x_i\}_{i=1}^{N}$. $x_i$ is a feature vector, and the goal of an unsupervised learning algorithm is to create a model that takes a feature vector $x$ as input and either transforms it into another vector or into a value that can be used to solve a practical problem. For example, in clustering [58], the model returns the index of the cluster for each feature vector in the dataset.

### Reinforcement Learning

Reinforcement learning [59] is a subfield of machine learning where the algorithm trains a policy that interacts with the environment and it can perceive the state of that environment as a vector of features. The machine can execute actions in every state. Different activities bring different rewards and could also move the machine to another state of the environment. The goal of a reinforcement learning algorithm is to learn a policy. A policy is a function $f$ similar to supervised learning that takes the feature vector of a state as input and outputs an action. The action is optimal if it maximizes the expected average reward.

Reinforcement learning solves a particular kind of problem where decision-making is sequential, and the goal is long-term, such as game playing, robotics, resource management, or logistics.

In this thesis, we utilize an ensemble-based [60] decision tree classifier for predicting the obstacle vehicle intentions.

## 4.2 Problem Formulation

As depicted in Figure 6, we consider the simplest case of a four-way intersection in which collisions with a single obstacle vehicle must be avoided. Hence, we have $m_t \leq 3$ possible obstacle maneuvers

$$M \in \{\text{straight}, \text{left}, \text{right}\}$$

i.e., the obstacle can only pass the intersection by driving straight, turning left, or right.

In addition to each corner case, we need to also define the following probabilities: $\omega_t^1$ associated with $M = \text{straight}$, $\omega_t^2$ with $M = \text{left}$, and $\omega_t^3$ with $M = \text{right}$. To this end, we collect trajectories (see Chapter 5)of multiple obstacles at intersections and use them to train a classifier, with categorical output $M$ and a suitably defined input feature vector $F$ that is sufficiently informative about the current state of the obstacle. As our ultimate goal is to get the time-varying discrete probability distribution $\{\omega_t^1, \omega_t^2, \omega_t^3\}$, we will employ classification methods that also return probabilities.

Each of the three possible corner-case scenarios is associated with an obstacle trajectory

$$(x_{k|t}^{o_i}, y_{k|t}^{o_i}),\ k = 0, \ldots, N,\ i = 1, 2, 3 \tag{4.1}$$

that will be used in (3.2g) and (3.3g). The obstacle coordinates in (4.1) are $(x_{k|t}^{o_i}, y_{k|t}^{o_i}) = f^{o_i}(d_{k|t}^{\text{o}})$. In order to obtain the continuous function $f^{o_i}(d_{k|t}^{\text{o}})$, we use B-spline interpolation on data collected from the SUMO simulator and parameterized by the distance $d^{\text{o}}$ traveled by the obstacle vehicle along the center of the lane.

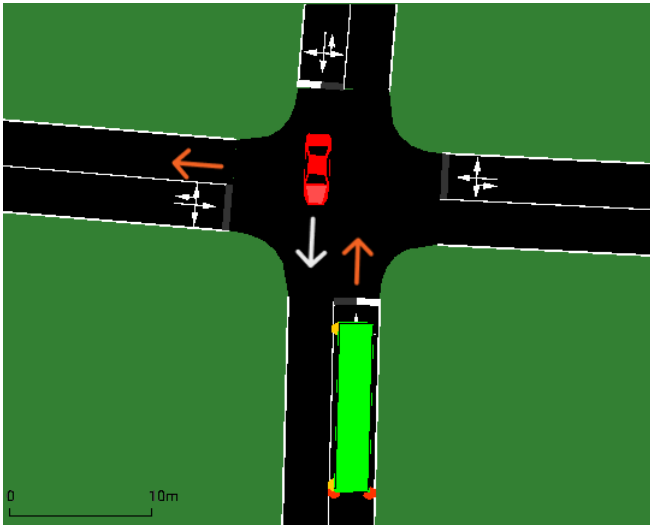


**Figure 6:** Uncontrolled intersection example: the ego vehicle (red) drives straight and the obstacle vehicle (green) takes a left turn

### 4.2.1 Features of Classifier

The feature vector consumed by the classifier is defined as

$$F_t = [v^{\mathrm{o}}\ a^{\mathrm{o}}\ \theta^{\mathrm{o}}_{diff}\ d^{\mathrm{o}}_{ln}\ d^{\mathrm{o}}_{lt}\ d^{\mathrm{o}}_{t}]'$$

where, assuming the center of the lane for a straight scenario as the reference lane, $v^{\mathrm{o}}$ and $a^{\mathrm{o}}$ are, respectively, the speed and acceleration of the obstacle, $\theta^{\mathrm{o}}_{\mathrm{diff}}$ denotes the obstacle's orientation with respect to the reference, $d^{\mathrm{o}}_{ln}$, $d^{\mathrm{o}}_{lt}$ are, respectively, its longitudinal and lateral distances from the reference, and $d^{\mathrm{o}}_{t}$ is the distance traveled by the obstacle.

We collected 605,016 training data and 151,254 test data. The bagged-tree classifier, based on 25 learners, is trained in approximately 213 s on an Intel(R) Corei7-8550U CPU @ 1.80GHz machine in MATLAB R2019a using the fitcensemble and templateTree functions available in the statistics and machine learning toolbox for MATLAB. The resulting number of splits in the decision tree is 90,323 (14.93% of the training data set). Further details regarding the choice of learners and no of splits in the

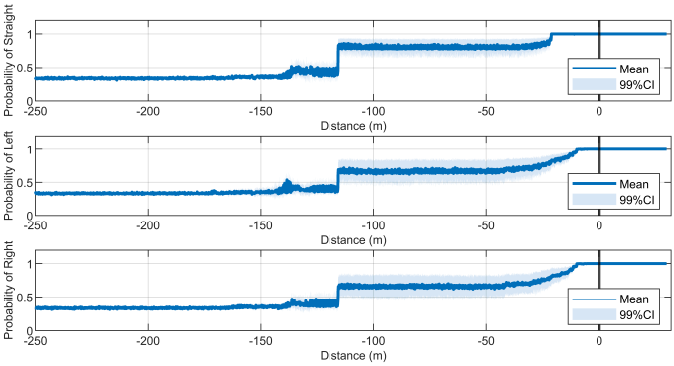


**Figure 7:** Predicted probability $\omega_t^i$ associated with the actual outcome $M_i$ of each training trajectory as a function of the distance traveled by obstacle $d_t^o$ with respect to the intersection, averaged on all training trajectories with outcome $M_i$, $i = 1, 2, 3$. The black vertical line represents the starting point of the intersection.

decision tree can be found in Appendix A

The predicted probability $\omega_t^i$ associated with the actual outcome $M_i$, $i = 1, 2, 3$, averaged on all trajectories, is displayed in Figure 7 (training data) and Figure 8 (test data): it is evident that when the obstacle is almost 150 m away from the intersection, all the scenarios are equally probable ($\omega_t^i \approx \frac{1}{3}$, $\forall i = 1, 2, 3$); they become more distinguishable as the obstacle moves towards the intersection. In particular, the obstacle is correctly predicted with probability 1 to drive straight about 21 m before the intersection, and about 5.5 m for left and right turns. Refer to Figures-Figures 23, 24, 25 in Appendix A for the detailed plot of probabilities of all trajectories in the training dataset for each scenario.

### 4.2.2 Bagged Decision tree

To classify the intent of the obstacle vehicle to go straight, turn left, or turn right we use a bagged decision tree [61], due to its ability to return

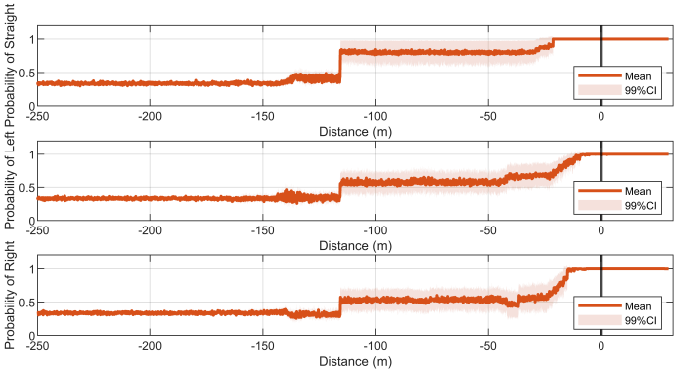


**Figure 8:** Predicted probability $\omega_t^i$ associated with the actual outcome $M_i$ of each testing trajectory as a function of the distance traveled by obstacle $d_t^o$ with respect to the intersection, averaged on all testing trajectories with outcome $M_i$, $i = 1, 2, 3$. The black vertical line represents the starting point of the intersection.

the discrete probability distribution $\{\omega_t^1, \omega_t^2, \omega_t^3\}$ associated with the predicted categorical target $M$. The motivation to choose this specific classifier is that, as we will discuss next, we observed experimentally that it produces better probability estimates compared to other methods, such as, e.g., naïve-bayes (NB) classifiers [62] and SVM [63], without using additional calibration methods such as Platt scaling or Isotonic Regression [64].

The Figures 9 and 10 compares the receiver operating characteristic (ROC) curve, relating the true positive rate (TPR) and false positive rate (FPR), obtained by the trained bagged decision tree with those obtained by an NB and SVM classifier for the three different scenarios, i.e., {Straight, Left, Right} on test data. More specifically, Figure 9 shows the ROC curve by considering data points whose distance from the intersection is in the interval $[100, 25]$ m (i.e., far away from the intersection), while Figure 10 for data in the interval $[25, 5]$ m (i.e., close to the intersection). It can be observed that, close to the intersection, the TPR is much

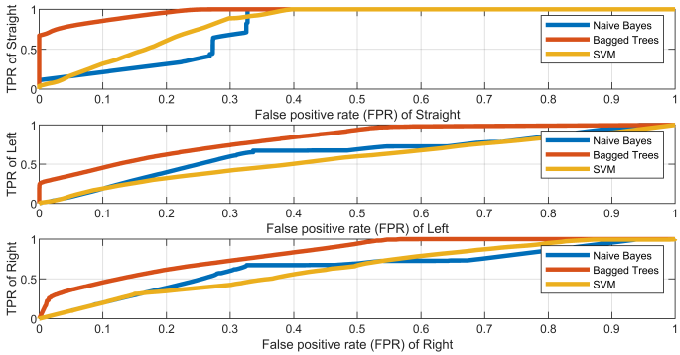


**Figure 9:** ROC curve of Bagged decision tree, Naïve-Bayes, and SVM for each scenario with the data points on testing dataset within a distance 100m to 25m from the intersection.

better for all the classifiers and the bagged decision tree has a significantly higher TPR than NB and SVM. As expected, it is nearly impossible to distinguish the three scenarios far away from the intersection. Finally, when the obstacle is closer than $5$ m to the intersection the TPR for the bagged decision tree is $1$ for all scenarios, indicating that the classifier correctly identifies the obstacle's intention.

### 4.2.3 Scenario-tree

For setting up the SMPC controller, we consider the scenario tree depicted in Figure 5, in which we set $k_{12} = k_{13} < k_{23}$, where $t+k_{12} = t+k_{13}$ is the time at which we assume we can distinguish between scenario 1 (straight) and the remaining scenarios, while $t+k_{23}$ is the time one will be able to distinguish between the left and right scenarios. The trained classifier generates the associated probabilities. The times at which we can distinguish the scenarios are learned offline from the probability distribution of training trajectories for each scenario. We considered the time of trajectory that distinguishes each scenario at the end. Refer to Figures-

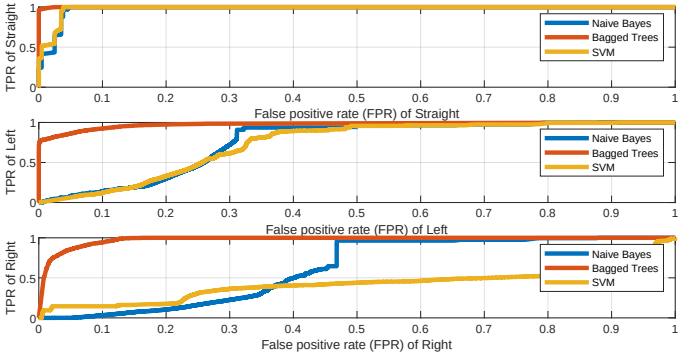


**Figure 10:** ROC curve of Bagged decision tree, Naïve-Bayes, and SVM for each scenario with the data points on testing dataset within a distance 25m to 5m from the intersection.

Figures 23, 24, 25 in Appendix A for the corresponding probability plots of each scenario.

## 4.3 Conclusions

In this chapter, we have given a brief introduction to the machine learning approaches and elaborated on the supervised learning technique we have used to predict the scenarios taken by the obstacle vehicle. Particularly, the features we have chosen for designing the classifier that provides probabilities to the SMPC control algorithm (3.3). As well as the performance comparison of the bagged decision tree with NB and SVM classifiers.

# Chapter 5

# Simulation of Urban Mobility

Simulation of Urban Mobility (SUMO) [39] is an open-source microscopic traffic simulator that has a set of tools for designing several traffic scenarios. SUMO has been developed by the employees of the Institute of Transportation Systems at the German Aerospace Center. Microscopic simulator means each vehicle is modeled explicitly, has its route, and moves independently.

These features make SUMO suitable to model obstacles with different types and parameters in order to generate realistic data of obstacle trajectories as well as to create test scenarios for our proposed control algorithm.

**Preliminaries**

**Edges**: A ″normal″ edge is a connection between two nodes (junctions) and it is unidirectional. The attributes of an edge are:

*id*: The id of the edge.

*from*: The id of the node it starts.

*to*: The id of the node it ends at.

*priority*: Indicate the importance of the road (optional)

*function*: Describes the edge purpose. The values are "normal", "internal", "connector", "crossing", and "walking area". The default is "normal".

**Netconvert**: It is a command line application that imports digital road networks from different sources and generates road networks suitable for SUMO environment.

**Route**: Defines the edges that the object should pass by.

## 5.1 Scenario Creation using SUMO

There are three main components required to build a scenario in SUMO, which is Network, Demand, and Simulation. Each of these components is explained in detail in the following sections.

### 5.1.1 Network Modeling

The network gives information on the type of roadway, the no. of lanes, the right of way, and the surroundings. It is in the form of .net.xml file that consists of edges and nodes, its connection, access to roads, traffic lights program definition, walking areas, speed limits, etc. The network can be defined either by using netedit tool within SUMO or netconvert command to import from other platforms such as Open Street Map (OSM), VISUM, Vissim, OPENDRIVE, and a few more.

#### Open Street Map

Open Street Map is a free editable map of the world. It can be imported to SUMO with the help of netconvert command. Here below is an example where an OSM file lucca.osm.xml is imported to SUMO network file format lucca.net.xml using netconvert command.

$$netconvert - -osm - files \;\; lucca.osm.xml - o \;\; lucca.net.xml$$

In this thesis, we have used OSM to extract a real road network for learning the obstacle vehicles as well as for testing our proposed control algorithms.

### 5.1.2 Demand Modeling

Demand modeling describes the vehicles that need to be running within the network. There are several approaches within SUMO to model demand that are by using trip definitions, flow definitions, randomization, origin-destination (OD) matrices, flow definitions and turn ratios, detector data, by hand, population statistics, data from other sources such as routing by turn probabilities, activity-based demand generation, random trips, importing OD matrices from VISUM or Vissim.

In our thesis, we created demand by hand that is we defined our .rou.xml file manually. There are 3 parts to define demand in SUMO. 1) a vehicle type that describes the vehicle's physical properties 2) a route of the vehicle 3) and the vehicle itself. An example .rou.xml file is given below where a vehicle with id $ego$ of type $car1$ that starts from edge $beg$ passes through $middle$ and stops at $end$.

$< routes >$

$< vType \;\; id = "car1" \;\; accel = "0.8" \;\; decel = "4.5"$
$length = "5" \;\; maxSpeed = "70" / >$

$< vehicle \;\; id = "ego" \;\; type = "car1" \;\; depart = "0"$
$color = "1, 0, 0" >$

$< route \;\; edges = "beg \;\; middle \;\; end" / >$

$< /vehicle >$

$< /routes >$

Here we emphasize important vehicle and vehicle type attributesthat we considered in our thesis.

The following are the vehicle attributes that we have used:-

- Vehicle id: Defines the name of the vehicle in the network. It is an id(string) value type.
- Vehicle type: Defines the type of vehicle. It is defined separately and has its attributes.
- depart: Time step at which the vehicle should enter the network. It is a float value type.

- departPos: The position at which the vehicle should enter the network. It's a float(m)/string (”random”, ”free”, ”random-free”, ”base”, ”last”, ”stop”). Default value is ”base”.
- color: Defines vehicle's color.
- route: Defines the ids of the edges that the vehicle should drive.

The following attributes are used within vType:-

- vClass: Defines an abstract vehicle class that a vehicle can take. The vClass available in SUMO is given in Table 1.
- length: It is the vehicle's netto-length (length) in meters. The default value is 5m.
- maxSpeed: The vehicle's (technical) maximum velocity in m/s. Its default value is 200km/hr.
- speedFactor: Individual Speed factor or Speed factor is the SUMO vehicle modeling parameter that determines the desired driving speed for a vehicle. The default value is 1, if the value is higher it can exceed the road limit speed or maximum speed.
- car following models: It describes the way a vehicle will follow the preceding vehicle in traffic. The Table 2 gives the available car following models in SUMO.

### 5.1.3 Simulation

To run a scenario we have to provide some input files to the configuration file. The input files required are given below:-

- **Road network:** In the simulation file we require to provide a network file using the option $--net-file$.
- **Traffic demand(routes):** The demand file that contains the information about vehicles, their parameters and corresponding routes to be simulated should be provided using the option $--route-files$.

**Table 1:** Vehicle class available in SUMO

| vClass | Comment |
|---|---|
| ignoring | drive on all lanes regardless of permissions |
| private | |
| emergency | |
| authority | |
| army | |
| vip | |
| pedestrian | Allowed on the lanes called 'sidewalks' in netconvert |
| Passenger | Default vehicle class and denotes regular passenger traffic |
| hov | High-occupancy vehicle |
| taxi | |
| bus | Urban line traffic |
| coach | Overland transport |
| delivery | Allowed on service roads that are not meant for public traffic |
| truck | |
| trailer | truck with trailer |
| motorcycle | |
| moped | Motorized 2-wheeler which may not drive on motorways |
| bicycle | |
| evehicle | Reserved for future mobility concepts such as electric vehicles which may get special access rights |
| tram | |
| rail_urban | Heavier than 'tram' but distinct from 'rail'. Encompasses Light Rail and S-Bahn |
| rail | heavy rail |
| rail_electric | heavy rail vehicle that may only drive on electrified tracks |
| rail_fast | High Speed rail |
| ship | basic class for navigating waterways |
| custom1 | reserved for user-defined semantics |
| custom2 | reserved for user-defined semantics |

**Table 2:** Car following models available in SUMO

| Attribute name | Description |
|---|---|
| Krauss | The Krauß-model with some modifications which is the default model used in SUMO |
| KraussOrig1 | The original Krauß-model |
| PWagner2009 | A model by Peter Wagner, using Todosiev's action points |
| BKerner | A model by Boris Kerner |
| IDM | The Intelligent Driver Model by Martin Treiber |
| IDMM | Variant of IDM |
| EIDM | Extended Intelligent Driver Model for subsecond simulation by Dominik Salles |
| KraussPS | default Krauss model with consideration of road slope |
| KraussAB | default Krauss model with bounded acceleration |
| SmartSK | Variant of the default Krauss model |
| Wiedemann | Car following model by Wiedemann (2-Parameters) |
| W99 | Car following model by Wiedemann, 10-Parameter version |
| Daniel1 | Car following model by Daniel Krajzewicz |
| ACC | Car following model by Milanés V. and Shladover S.E. |
| CACC | Car following model by Milanés V. and Shladover S.E. |
| Rail | Model for various train types |

- **Additional files:** More than one file can be added as an additional file for the simulation. Usually, it consists of the following type of files such as:-
  - infrastructure related: traffic light programs, induction loops, and bus stops
  - additional visualization: POIs and polygons (i.e. rivers and houses)
  - dynamic simulation control structures: variable speed signs and rerouters
  - demand related entities: vehicle types and routes

  It is added to the configuration file using $--additional-files$
- **Parsing order:** All the inputs for the simulation should be added in a certain order and are given as:-
  - Read the network
  - Read the additional file in the same order as they are
  - The route files are opened and the first n steps are read
  - each n time steps, the routes for the next n time steps are read

**Traffic Control Interface (Traci)**

It is an interface that gives access to a running traffic simulation, also allows retrieval of values, and manipulates the behavior of simulated objects "on-line". In this thesis, we use TraCI4Matlab [65] to interface SUMO with MatLab to retrieve the parameter values of ego and obstacle vehicles.

## 5.2 Data collection of Obstacle vehicle

We have simulated several examples of obstacle vehicles in SUMO to collect the parameters of the same, and to build a scenario prediction model

which we will discuss in the next chapter. For the road network, we considered a real uncontrolled intersection extracted from Open Street Map and use different types of vehicles.

In order to define the driving style, we rely on the Intelligent Driver Model (IDM) [66], included in SUMO and it describes the dynamic position and velocity of a vehicle as follows.

$$\dot{x}_\delta = v_\delta \tag{5.1a}$$

$$\dot{v}_\delta = a_{\max}\left(1 - \left(\frac{v_\delta(t)}{v_{\text{des}}(t)}\right)^{\epsilon} - \left(\frac{\sigma^*\left(v_\delta(t), \Delta v_\delta(t)\right)}{\sigma_\delta(t)}\right)^2\right) \tag{5.1b}$$

where $a_{\max}$ and $d_{\max}$ are the vehicle's maximum desired acceleration and deceleration, respectively; $v_\delta$, $v_{\text{des}}$, and $\epsilon$ are the current velocity, the desired velocity, and a tuning parameter; $\Delta v_\delta$ is the relative velocity with respect to the vehicle in front. Moreover, in (5.1)

$$\sigma^*\left(v_\delta(t), \Delta v_\delta(t)\right) = \sigma_0 + v_\delta(t)T + \frac{v_\delta(t)\Delta v_\delta(t)}{2\sqrt{a_{\max} d_{\max}}} \tag{5.2}$$

where $\sigma_0$ is the minimum desired distance from the vehicle in front and $T$ the desired time headway, while the net distance between the two vehicles is

$$\sigma_\delta(t) = d_{\text{f}}(t) - d_\delta(t) - l_{\text{f}} = \Delta d_\delta(t) - l_{\text{f}} \tag{5.3}$$

where label f refers to the front vehicle, $d$ is the position of a vehicle along the centerline of the road, and $l_{\text{f}}$ is the length of the vehicle in front.

In this thesis, as we consider free-road behaviors, Equation (5.1) simplifies to $\dot{v}_\delta = a_{\max}\left(1 - \left(\frac{v_\delta(t)}{v_{\text{des}}(t)}\right)^{\epsilon}\right)$.

In SUMO, in a free-driving scenario the term $v_{\text{des}}(t)$ is given by

$$v_{\text{des}}(t) = \min(v_{\max}, s v_{\max}^{\text{des}}, s v_{\text{lim}})$$

where $v_{\max}$, $v_{\max}^{\text{des}}$, and $v_{\text{lim}}$ are the maximum speed, desired maximum speed, and speed limit respectively, and $s$ is a speed factor.

More specifically to collect obstacle vehicle data, we consider 3 types of vehicles, namely a passenger car, a motorcycle, and a bus, each with 5 different speed factor values ranging from 0.6 to 1.4.

Moreover, for each type of vehicle, we select 6 different vehicle speeds within the range of 40 to 60 km/h. As a result, we collected 90 obstacle vehicle variations for each maneuver thus a total of 270 scenarios, of which 216 trajectories are used for training and the remaining 54 for testing the classifier. The obstacle starts 250 m away from the beginning of the intersection and we considered the data from there up to 30m after the starting point of the intersection which gives a total distance of 280.1 m by using a resolution of 0.1 m. We collected 605,016 training data and 151,254 test data as we discussed in Chapter 4.

## 5.3 Conclusion

In this chapter, we have introduced the SUMO traffic simulator which we have used to collect data required for learning the prediction model of the obstacle vehicle. We were able to simulate quite some examples of the same in a real road network.

# Chapter 6

# Simulation Results

To analyze the performance of our proposed SMPC method, we compare it with the PMPC and RMPC approaches introduced in the previous chapters. We considered an uncontrolled intersection in Lucca, Italy, as shown in Figure 11 that is imported from Open Street Map and several realistic examples were simulated with the help of SUMO. We model the ego vehicle as a passenger car with a length of 5 m with a given maximum speed $v_{max}$ and $s$ = 1.

**Performance Analysis**

Closed-loop performance is assessed by the following measure

$$J_{\mathrm{cl}} = \sum_{t=0}^{t_{\mathrm{s}}} \|X_t - X^r_{0|t}\|^2_Q + \|U_t - U^r_{0|t}\|^2_R \tag{6.1}$$

consistently with the MPC cost function, where $t_{\mathrm{s}}$ is the total simulation time. SUMO's IDM model is used offline to generate a discrete set of reference samples for $X^r$ and $U^r$, given a desired maximum speed $v_{\max}$ of the ego vehicle. Such samples are used to define B-spline interpolation functions $f_{X^{\mathrm{r}}} : \mathbb{R} \mapsto \mathbb{R}^5$, $f_{U^{\mathrm{r}}} : \mathbb{R} \mapsto \mathbb{R}^2$, such that, for a generic distance $d$ along the central line of the road, $f_{X^{\mathrm{r}}}(d) = [x_{\mathrm{r}}(d)\ y_{\mathrm{r}}(d)\ \theta_{\mathrm{r}}(d)\ v_{\mathrm{r}}(d)\ \delta_{\mathrm{r}}(d)]'$ and $f_{U^{\mathrm{r}}}(d) = [a_{\mathrm{r}}(d)\ \omega_{\mathrm{r}}(d)]'$ provide the corresponding state and input


**Figure 11:** The Google image of the uncontrolled intersection in Lucca, Italy

references, respectively. Then, at each controller execution step $t$, given the piecewise-constant velocity and orientation profiles

$$\begin{aligned} v(\tau) &= v^{\star}_{k+1|t-1} \\ \theta(\tau) &= \theta^{\star}_{k+1|t-1}, \quad \forall \tau \in [(t+k)T_{\mathrm{s}}, (t+k+1)T_{\mathrm{s}}] \end{aligned}$$

obtained from the previous MPC solution $\{v^{\star}_{k|t-1}, \theta^{\star}_{k|t-1}\}$, $k = 0, \ldots, N$, we use explicit Runge-Kutta 4 to integrate the differential equation

$$\dot{d}(\tau) = v(\tau)\cos(\theta(\tau) - \theta_{\mathrm{r}}(d(\tau)))$$

over the prediction horizon $[tT_{\mathrm{s}}, (t+N)T_{\mathrm{s}}]$ to get samples $d_{k|t}$, $k = 0, \ldots, N$, of the distance $d$ traveled by the ego vehicle, starting from the initial condition $d(tT_{\mathrm{s}})$ (i.e., the current longitudinal distance traveled by the ego vehicle). Finally, the corresponding MPC references $X^{\mathrm{r}}_{k|t} = f_{X^{\mathrm{r}}}(d_{k|t})$ and $U^{\mathrm{r}}_{k|t} = f_{U^{\mathrm{r}}}(d_{k|t})$.

## 6.1 Simulation examples

We use sample time $T_{\mathrm{s}} = 0.1$ s for both discretizing the continuous-time ego-vehicle model and executing the MPC controller, which has a prediction horizon $N = 40$. The CPU time for solving a single Prescient, Robust, and Stochastic MPC problem is on average, respectively, 0.7 s, 0.9 s, and 2 s using CasADi [67] and the interior-point nonlinear programming solver IPOPT [68]. Note that no particular care was taken in solving the MPC problems efficiently, and dedicated software will produce significantly lower computational times.

### 6.1.1 Collision scenario - Obstacle bus taking a left turn

**Example 1:** The first example refers to the case in which the ego vehicle aims at passing the intersection without turning at a maximum speed $v_{\max}$ = 50 km/h, while the obstacle is a bus coming from the opposite direction at $v^o_{\max}$ = 54 km/h, with speed factor $s = 1.3$, taking a left turn. The results obtained using the different MPC formulations in this scenario where a collision is possible are described in Figure 14 and Table 3, from which we can infer that all the control algorithms result in the same closed-loop costs since the ego vehicle sway from the reference path to avoid the collision with the bus. Figure 13 shows the probabilities $\omega_t^1$, $\omega_t^2$, $\omega_t^3$ estimated by the classifier with respect to the distance traveled by ego $d(0|t)$ from the intersection and the black vertical line at $0$ m represent the starting point of the intersection.

**Example 5:** This is a similar scenario as Example 1 where the ego vehicle crosses the intersection without turning at $v_{\max} = 45$ km/h and the obstacle is a bus taking a left turn at a speed $v^o_{\max}$ = 50 km/h with speed factor $s = 1$. This is also a collision scenario where all the control algorithms have the same closed-loop cost.

**Table 3:** Closed-loop cost $J_{\text{cl}}$ (6.1) of prescient, robust, and stochastic MPC obtained in the five test examples.

| Example | prescient MPC | robust MPC | stochastic MPC |
|---|---|---|---|
| Example 1 | 14.8514 | 14.8514 | 14.8514 |
| Example 2 | 6.9371 | 33.1935 | 7.4447 |
| Example 3 | 20.4692 | 40.6931 | 20.4692 |
| Example 4 | 6.9371 | 117.291 | 10.3261 |
| Example 5 | 10.5519 | 10.5519 | 10.5519 |

### 6.1.2 Non-collision scenario - Obstacle motorcycle turning right

**Example 2:** The ego vehicle travels at $v_{\max}$ = 43 km/h and passes the crossroad without turning, the obstacle vehicle is a motorcycle taking a right turn at $v^o_{\max}$ = 44 km/h with speed factor $s = 1.1$. Figure 17 shows the positions of ego and obstacle vehicle near the intersection, and the probabilities $\omega_t^1$, $\omega_t^2$, $\omega_t^3$ estimated by the classifier with respect to the distance traveled by ego $d_{0|t}$ from the intersection. It is apparent that, initially, RMPC and SMPC behave similarly until the scenarios start becoming distinguishable, i.e., the probability distribution starts getting non-uniform. In Figure 16 the black vertical line represents the point at which the ego vehicle enters the intersection. Note that, as remarked earlier, a right turn can only be predicted with high probability by the classifier when the obstacle is close to the intersection. Thanks to the use of the scenario tree, SMPC tends to behave more similarly to PMPC than RMPC, which instead remains conservative. This is a crucial benefit of using SMPC compared to other schemes: even without an *exact* information about the future positions of the obstacle, past information is used in a more clever way to infer *possible* obstacle positions and the related chances to realize. Thanks to the probabilistic model and the use of additional degrees of freedom associated with it, SMPC takes control decisions that, by construction, are best in expectation. This behavior is reflected in Table 3, which shows that the cost associated with SMPC is

significantly smaller than that of RMPC, which instead gives the same importance to all possible scenarios.

**Example 4**. The ego vehicle travels straight across the intersection at $v\max = 43$ km/h. A motorcycle with a speed factor $s = 1.3$ traveling at a maximum speed of 44 km/h takes a right turn. This is a non-collision scenario, and in Table 3 we observe that the RMPC has the highest closed-loop cost as it takes into account the scenario (left turn) that leads to a collision as well, while SMPC will consider such a scenario very unlikely, resulting in an overall control action that has a cost closer to that of PMPC.

### 6.1.3 Non-collision scenario - Obstacle passenger car going straight

**Example 3:** The ego travels at $v_{\max} = 45$ km/h takes a left turn, and the obstacle is a car with speed factor $s = 1$ driving straight across the intersection in the opposite direction at a maximum speed of $v^o_{\max}$ = 40 km/h as shown in Figure 18. There is no potential collision between the ego vehicle and the obstacle in this example. From Table 3 it is evident that SMPC works closer to PMPC than RMPC thanks to the exploitation of the scenario tree. As shown in Figure 19, the straight scenario is detected early by the classifier, thus SMPC lifts the collision avoidance constraints from the other two scenarios. Instead, the actions commanded by RMPC suffer from the presence of the irrelevant scenario (right turn), as it causes the ego to sway to avoid the collision at the turn with the obstacle that takes the right turn and allowing the obstacle to go ahead.

## 6.2 Conclusions

In this chapter, we have illustrated several examples that allowed us to compare our control approach ie. Scenario-based SMPC to the Prescient as well as Robust MPC approach. From the simulation results and by referring to Table 3 our approach has cost closer to Prescient MPC for all these examples. And the cost is the same for all the control approaches

only in case of collision.

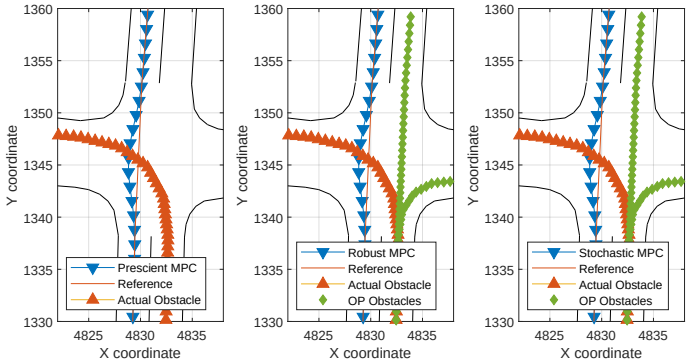


**Figure 12:** Ego and obstacle vehicle positions near the intersection. The red line represents the actual obstacle maneuver and the green line represents the other two possible maneuvers of the obstacle.

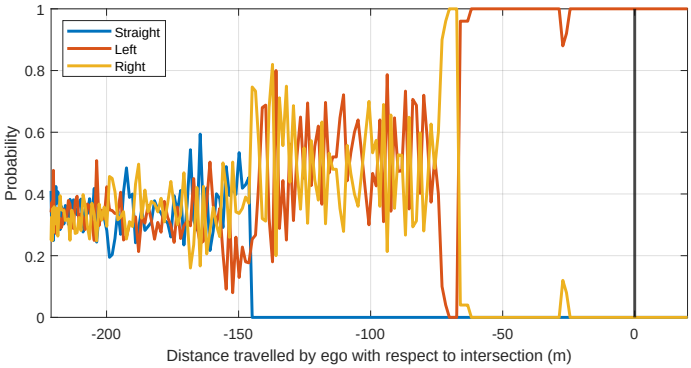


**Figure 13:** Estimated scenario probability of the obstacle vehicle as a function of distance traveled by ego $d(0|t)$ from the intersection. The black vertical line represents the starting of the intersection.

**Figure 14:** Example 1: Simulation results

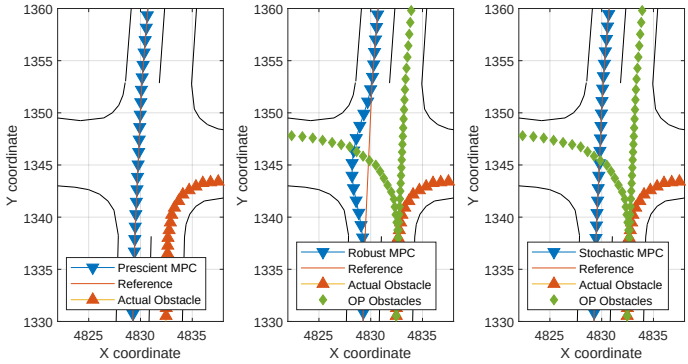


**Figure 15:** Ego and obstacle vehicle positions near the intersection. The red line represents the actual obstacle maneuver and the green line represents the other two possible maneuvers of the obstacle.

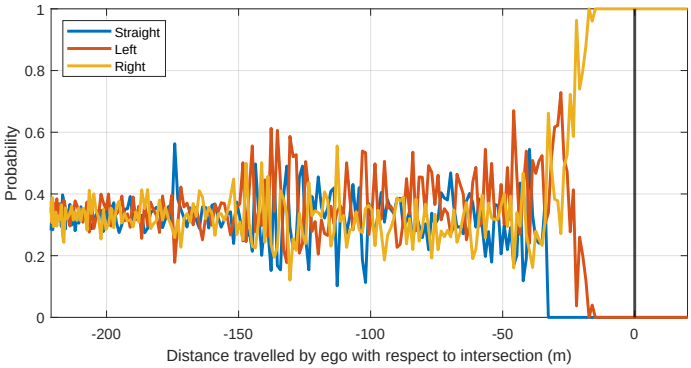


**Figure 16:** Estimated scenario probability of the obstacle vehicle as a function of distance traveled by ego $d(0|t)$ from the intersection. The black vertical line represents the starting of the intersection.

**Figure 17:** Example 2: Simulation results

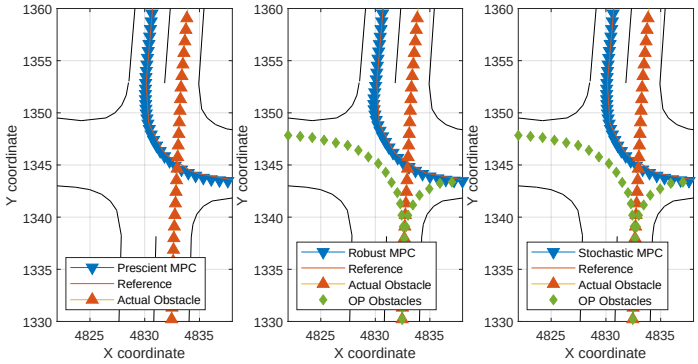


**Figure 18:** Ego and obstacle vehicle positions near the intersection. The red line represents the actual obstacle maneuver and the green line represents the other two possible maneuvers of the obstacle.

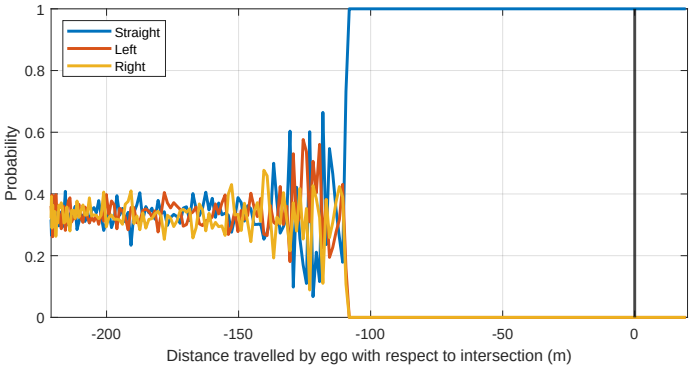


**Figure 19:** Estimated scenario probability of the obstacle vehicle as a function of distance traveled by ego $d(0|t)$ from the intersection. The black vertical line represents the starting of the intersection.

**Figure 20:** Example 3: Simulation results

# Chapter 7

# Conclusion & Future Work

We proposed a stochastic MPC approach to autonomous driving, focusing in particular on the case of uncontrolled intersections. By training a classifier on a single obstacle case which also returns the probabilities corresponding to each possible predictable categorical value, we set up the stochastic optimization problem to solve at each time $t$ to get the required command action on the ego vehicle. We have shown that using such probabilities is beneficial, as it makes SMPC less conservative than a robust MPC approach, in which they are not taken into account. And the corresponding results are closer to a Prescient MPC.

Future work will be devoted to extending the idea to multiple obstacles; in such a case, it will be important to develop methods to automatically reduce the number of scenarios considered in the SMPC formulation to ensure that the computational complexity of the approach remains manageable. Another important feature to investigate is more meaningful MPC performance indices that also take into account driving quality.

So far we have considered only uncontrolled intersections, in the future, we would like to extend our control approach for broader cases such as lane changing and normal driving scenarios where the possibility of collision cannot be avoided.

# Appendix A

# Appendix Title

## A.1 Bagged decision tree specifications

The parameters of the bagged decision tree are chosen based on Figures 21, 22. In 21 10-fold classification error is computed on the decision trees constructed with training dataset for trees with several number of splits. Figure 22 shows the classification error in the testing dataset for different number of learners.

## A.2 Scenario Tree Generation

This section clarifies the construction of the scenario tree mentioned in Chapter 4. It is constructed offline using the trajectories composing the training dataset. The probability estimate from the classifier for each trajectory in the training dataset is observed to choose the time instant or the distance $k_{12}, k_{13}, k_{23}$ at which each scenario is distinguishable from each other, i.e., when the probability of each scenario $\omega_t^1 = \omega_t^2 = \omega_t^3 = 1$ at last among all the training datasets as depicted in Figures 23, 24, and 25.

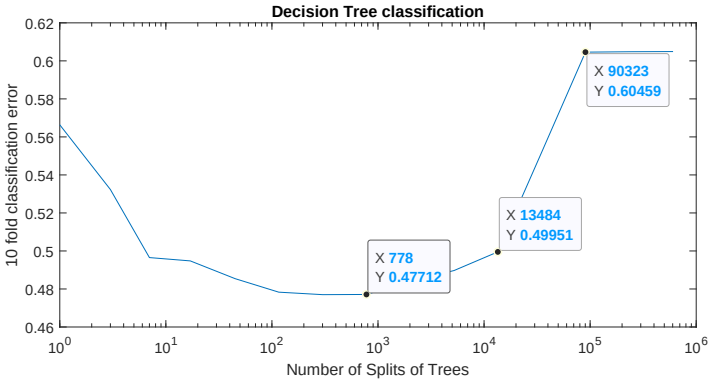


**Figure 21:** 10 fold classification error with respect to no of splits in the decision tree

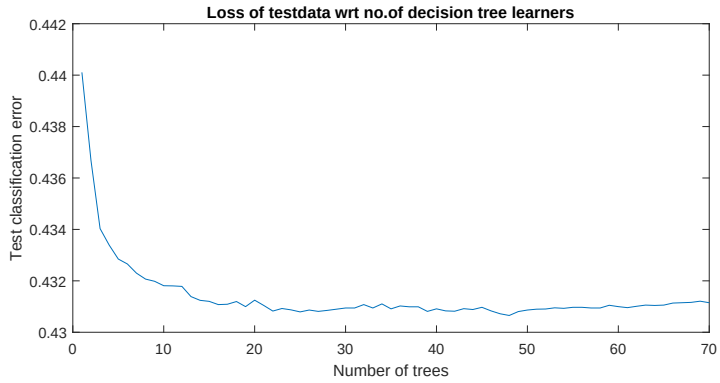


**Figure 22:** Classification error on testing dataset based on the no of learners

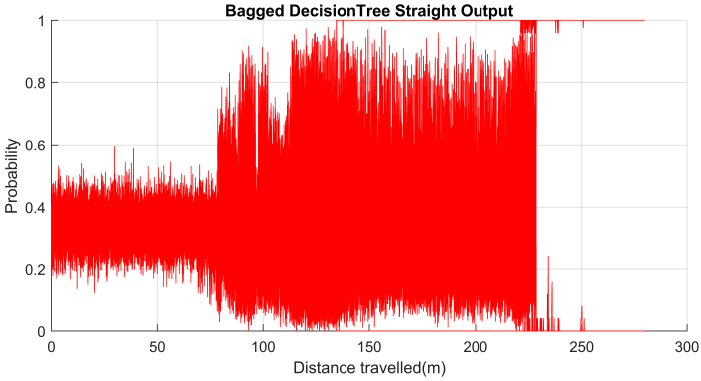


**Figure 23:** Probability estimate of all the training trajectories to be straight scenario

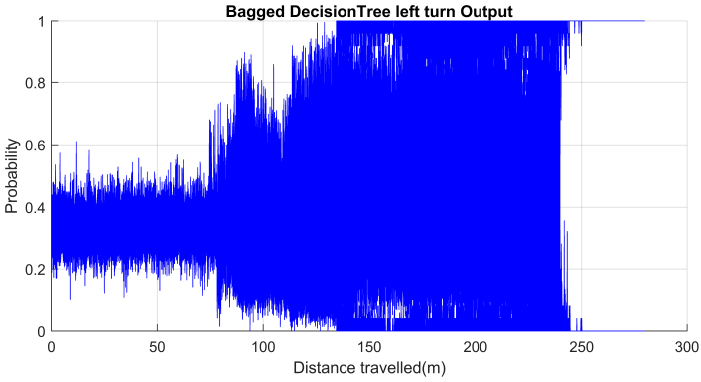


**Figure 24:** Probability estimate of all the training trajectories to be left scenario

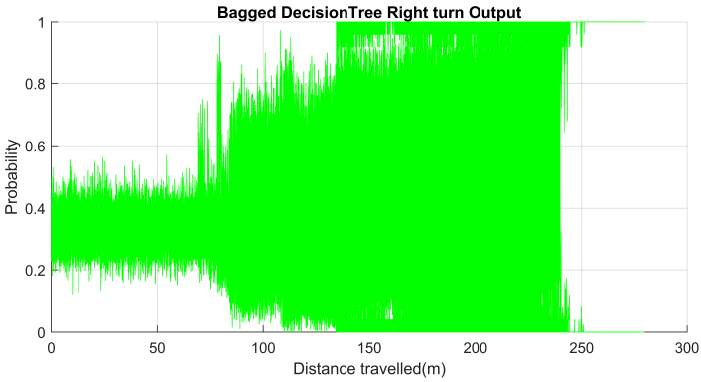


**Figure 25:** Probability estimate of all the training trajectories to be right scenario

# Bibliography


[1] Keshav Bimbraw. "Autonomous cars: Past, present and future a review of the developments in the last century, the present scenario and the expected future of autonomous vehicle technology". In: *12th international conference on informatics in control, automation and robotics (ICINCO)*. Vol. 1. 2015, pp. 191–198.

[2] Julius Ziegler et al. "Making bertha drive—an autonomous journey on a historic route". In: *IEEE Intelligent transportation systems magazine* 6.2 (2014), pp. 8–20.

[3] Jan Becker et al. "Bosch's vision and roadmap toward fully autonomous driving". In: *Road vehicle automation*. 2014, pp. 49–59.

[4] Claudine Badue et al. "Self-driving cars: A survey". In: *Expert Systems with Applications* 165 (2021), p. 113816.

[5] Zehang Sun, George Bebis, and Ronald Miller. "On-road vehicle detection using optical sensors: A review". In: *Proceedings. The 7th International IEEE Conference on Intelligent Transportation Systems (IEEE Cat. No. 04TH8749)*. 2004, pp. 585–590.

[6] Mahir Gulzar, Yar Muhammad, and Naveed Muhammad. "A Survey on Motion Prediction of Pedestrians and Vehicles for Autonomous Driving". In: *IEEE Access* 9 (2021), pp. 137957–137969. DOI: 10.1109/ACCESS.2021.3118224.

[7] Ashwin Carvalho et al. "Stochastic predictive control of autonomous vehicles in uncertain environments". In: *12th International Symposium on Advanced Vehicle Control*. 2014, pp. 712–719.

[8] Yonghwan Jeong and Kyongsu Yi. "Target Vehicle Motion Prediction-Based Motion Planning Framework for Autonomous Driving in Uncontrolled Intersections". In: *IEEE Transactions on Intelligent Transportation Systems* 22.1 (2021), pp. 168–177. DOI: 10.1109/TITS.2019.2955721.

[9] Nachiket Deo, Akshay Rangesh, and Mohan M Trivedi. "How would surround vehicles move? a unified framework for maneuver classification and motion prediction". In: *IEEE Transactions on Intelligent Vehicles* 3.2 (2018), pp. 129–140.

[10] Dachuan Li et al. "Behavior and Interaction-aware Motion Planning for Autonomous Driving Vehicles based on Hierarchical Intention and Motion Prediction". In: *IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. 2020, pp. 1–8. DOI: 10.1109/ITSC45102.2020.9294182.

[11] I. Batkovic et al. "A Computationally Efficient Model for Pedestrian Motion Prediction". In: *European Control Conference (ECC)*. June 2018, pp. 374–379.

[12] Yeping Hu, Wei Zhan, and Masayoshi Tomizuka. "Probabilistic prediction of vehicle semantic intention and motion". In: *IEEE Intelligent Vehicles Symposium*. 2018, pp. 307–313.

[13] Xulei Liu et al. "A Deep Learning-based Approach to Line Crossing Prediction for Lane Change Maneuver of Adjacent Target Vehicles". In: *IEEE International Conference on Mechatronics (ICM)*. 2021, pp. 1–6. DOI: 10.1109/ICM46511.2021.9385665.

[14] Seungje Yoon and Dongsuk Kum. "The multilayer perceptron approach to lateral motion prediction of surrounding vehicles for autonomous vehicles". In: *IEEE Intelligent Vehicles Symposium*. 2016, pp. 1307–1312.

[15] Tsung-Ming Hsu, Yu-Rui Chen, and Cheng-Hsien Wang. "Decision Making Process of Autonomous Vehicle with Intention-Aware Prediction at Unsignalized Intersections". In: *International Automatic Control Conference (CACS)*. 2020, pp. 1–4. DOI: 10.1109/CACS50047.2020.9289815.

[16] Youngmin Yoon and Kyongsu Yi. "Design of Longitudinal Control for Autonomous Vehicles based on Interactive Intention Inference of Surrounding Vehicle Behavior Using Long Short-Term Memory". In: *IEEE International Intelligent Transportation Systems*

*Conference (ITSC)*. 2021, pp. 196–203. DOI: 10.1109/ITSC48978.2021.9564986.

[17] Abdelmoudjib Benterki et al. "Artificial Intelligence for Vehicle Behavior Anticipation: Hybrid Approach Based on Maneuver Classification and Trajectory Prediction". In: *IEEE Access* 8 (2020), pp. 56992–57002. DOI: 10.1109/ACCESS.2020.2982170.

[18] Abdelmoudjib Benterki et al. "Prediction of Surrounding Vehicles Lane Change Intention Using Machine Learning". In: *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*. Vol. 2. 2019, pp. 839–843. DOI: 10.1109/IDAACS.2019.8924448.

[19] Hannes Weinreuter et al. "Intention prediction of car drivers at inner city junctions". In: *International Symposium on Electronics and Telecommunications (ISETC)*. 2020, pp. 1–4. DOI: 10.1109/ISETC50328.2020.9301066.

[20] D.Q. Mayne, J.B. Rawlings, and M.M. Diehl. *Model Predictive Control: Theory and Design*. 2nd ed. Madison,WI: Nob Hill Publishing, LCC, 2018.

[21] S. Gros et al. "From Linear to Nonlinear MPC: bridging the gap via the Real-Time Iteration". In: *Int. J. Control* 93.1 (2020), pp. 62–80.

[22] A. Bemporad et al. "Model Predictive Control of Turbocharged Gasoline Engines for Mass Production". In: *WCX$^{TM}$: SAE World Congress Experience*. Detroit, MI, USA, Apr. 2018.

[23] Chang Liu et al. "Path planning for autonomous vehicles using model predictive control". In: *IEEE Intelligent Vehicles Symposium (IV)*. 2017, pp. 174–179.

[24] M. Graf Plessen et al. "Spatial-based predictive control and geometric corridor planning for adaptive cruise control coupled with obstacle avoidance". In: *IEEE Trans. Contr. Systems Technology* 26.1 (2018), pp. 38–50.

[25] Ali Mesbah. "Stochastic model predictive control: An overview and perspectives for future research". In: *IEEE Control Systems Magazine* 36.6 (2016), pp. 30–44.

[26] D. Bernardini and A. Bemporad. "Stabilizing Model Predictive Control of Stochastic Constrained Linear Systems". In: *IEEE Trans. Automatic Control* 57.6 (2012), pp. 1468–1480.

[27] S. Di Cairano et al. “Stochastic MPC With Learning for Driver-Predictive Vehicle Control and its Application to HEV Energy Management”. In: *IEEE Trans. Contr. Systems Technology* 22 (3 2014), pp. 1018–1031.

[28] M. Bichi et al. “Stochastic Model Predictive Control with Driver Behavior Learning for Improved Powertrain Control”. In: *Proc. 49th IEEE Conf. on Decision and Control*. Atlanta, GA, USA, 2010, pp. 6077–6082.

[29] G. Ripaccioli et al. “A Stochastic Model Predictive Control Approach for Series Hybrid Electric Vehicle Power Management”. In: *Proc. American Contr. Conf.* Baltimore, MD, 2010, pp. 5844–5849.

[30] Siddharth H. Nair et al. “Stochastic MPC with Multi-modal Predictions for Traffic Intersections”. In: *IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*. 2022, pp. 635–640. DOI: 10.1109/ITSC55140.2022.9921751.

[31] I. Batkovic et al. “Real-Time Constrained Trajectory Planning and Vehicle Control for Proactive Autonomous Driving with Road Users”. In: *European Control Conference (ECC)*. June 2019, pp. 256–262.

[32] Ivo Batkovic et al. “A Robust Scenario MPC Approach for Uncertain Multi-Modal Obstacles”. In: *IEEE Control Systems Letters* 5.3 (2021), pp. 947–952. DOI: 10.1109/LCSYS.2020.3006819.

[33] Ivo Batkovic et al. “Safe Trajectory Tracking in Uncertain Environments”. In: *IEEE Transaction on Automatic Control* (). (conditionally accepted), https://arxiv.org/abs/2001.11602.

[34] Ivo Batkovic et al. “Experimental Validation of Safe MPC for Autonomous Driving in Uncertain Environments”. In: *IEEE Transaction on Control Systems Technology* (2023). (in press).

[35] David Madås et al. “On path planning methods for automotive collision avoidance”. In: *IEEE intelligent vehicles symposium (IV)*. 2013, pp. 931–937.

[36] Karl Berntorp. “Path planning and integrated collision avoidance for autonomous vehicles”. In: *American Control Conference (ACC)*. 2017, pp. 4023–4028.

[37] Carl Hynén Ulfsjöö and Daniel Axehill. "On Integrating POMDP and Scenario MPC for Planning under Uncertainty – with Applications to Highway Driving". In: *IEEE Intelligent Vehicles Symposium (IV)*. 2022, pp. 1152–1160. DOI: 10.1109/IV51971.2022.9827005.

[38] Rui Oliveira, Siddharth H Nair, and Bo Wahlberg. "Interaction and Decision Making-aware Motion Planning using Branch Model Predictive Control". In: *arXiv preprint arXiv:2302.00060* (2023).

[39] Pablo Alvarez Lopez et al. "Microscopic Traffic Simulation using SUMO". In: *21st International Conference on Intelligent Transportation Systems (ITSC)*. 2018, pp. 2575–2582.

[40] Eric Ziegel. *Numerical recipes: The art of scientific computing*. 1987.

[41] Francesco Borrelli, Alberto Bemporad, and Manfred Morari. *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[42] Rajesh Rajamani. *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

[43] Luigi Chisci, J Anthony Rossiter, and Giovanni Zappa. "Systems with persistent disturbances: predictive control with restricted constraints". In: *Automatica* 37.7 (2001), pp. 1019–1028.

[44] David Q Mayne, Maria M Seron, and SV Raković. "Robust model predictive control of constrained linear systems with bounded disturbances". In: *Automatica* 41.2 (2005), pp. 219–224.

[45] David Q Mayne and Eric C Kerrigan. "Tube-based robust nonlinear model predictive control1". In: *IFAC Proceedings Volumes* 40.12 (2007), pp. 36–41.

[46] Johannes Köhler, Matthias A Müller, and Frank Allgöwer. "A novel constraint tightening approach for nonlinear robust model predictive control". In: *Annual American control conference (ACC)*. IEEE. 2018, pp. 728–734.

[47] Tor Aksel N Heirung et al. "Stochastic model predictive control—how does it work?" In: *Computers & Chemical Engineering* 114 (2018), pp. 158–170.

[48] Ali Mesbah. "Stochastic model predictive control: An overview and perspectives for future research". In: *IEEE Control Systems Magazine* 36.6 (2016), pp. 30–44.

[49] Ali Mesbah, Ilya V Kolmanovsky, and Stefano Di Cairano. "Stochastic model predictive control". In: *Handbook of Model Predictive Control* (2019), pp. 75–97.

[50] Abebe Geletu et al. "Advances and applications of chance-constrained approaches to systems optimisation under uncertainty". In: *International Journal of Systems Science* 44.7 (2013), pp. 1209–1232.

[51] Ali Mesbah et al. "Stochastic nonlinear model predictive control with probabilistic constraints". In: *American control conference*. IEEE. 2014, pp. 2413–2419.

[52] Daniele Bernardini and Alberto Bemporad. "Scenario-based model predictive control of stochastic constrained linear systems". In: *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*. 2009, pp. 6333–6338. DOI: 10.1109/CDC.2009.5399917.

[53] Sergio Lucia and Sebastian Engell. "Robust nonlinear model predictive control of a batch bioreactor using multi-stage stochastic programming". In: *2013 European Control Conference (ECC)*. IEEE. 2013, pp. 4124–4129.

[54] Andriy Burkov. *The hundred-page machine learning book*. Vol. 1. Andriy Burkov Quebec City, QC, Canada, 2019.

[55] L. Rokach and O. Maimon. "Top-down induction of decision trees classifiers - a survey". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 35.4 (2005), pp. 476–487. DOI: 10.1109/TSMCC.2004.843247.

[56] Andreas Christmann and Ingo Steinwart. "Support vector machines". In: (2008).

[57] G.P. Zhang. "Neural networks for classification: a survey". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 30.4 (2000), pp. 451–462. DOI: 10.1109/5326.897072.

[58] Anil K. Jain. "Data clustering: 50 years beyond K-means". In: *Pattern Recognition Letters* 31.8 (June 2010), pp. 651–666. DOI: 10.1016/j.patrec.2009.09.011. URL: https://doi.org/10.1016%2Fj.patrec.2009.09.011.

[59] Marco Wiering and Martijn Van Otterlo. *Reinforcement Learning: State of the Art*. English. Springer, 2012. ISBN: 978-3-642-27644-6. DOI: 10.1007/978-3-642-27645-3.

[60] Thomas G Dietterich. “Ensemble methods in machine learning”. In: *International workshop on multiple classifier systems*. Springer. 2000, pp. 1–15.

[61] Nitesh V Chawla and David A Cieslak. “Evaluating probability estimates from decision trees”. In: *American Association for Artificial Intelligence*. 2006.

[62] Feng-Jen Yang. “An Implementation of Naive Bayes Classifier”. In: *International Conference on Computational Science and Computational Intelligence (CSCI)*. 2018, pp. 301–306. DOI: 10.1109/CSCI46756.2018.00065.

[63] Eddy Mayoraz and Ethem Alpaydin. “Support vector machines for multi-class classification”. In: *International Work-Conference on Artificial Neural Networks*. Springer. 1999, pp. 833–842.

[64] Alexandru Niculescu-Mizil and Rich Caruana. “Predicting good probabilities with supervised learning”. In: *Proceedings of the 22nd international conference on Machine learning*. 2005, pp. 625–632.

[65] Andrés F Acosta, Jorge E Espinosa, and Jairo Espinosa. “TraCI4Matlab: Enabling the integration of the SUMO road traffic simulator and Matlab® through a software re-engineering process”. In: *Modeling Mobility with Open Data: 2nd SUMO Conference 2014 Berlin, Germany, May 15-16, 2014*. Springer. 2015, pp. 155–170.

[66] M. Treiber, A. Hennecke, and D. Helbing. “Congested traffic states in empirical observations and microscopic simulations”. In: *Physical Review E* 62 (2 Aug. 2000).

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

[68] Andreas Wächter and Lorenz T Biegler. “On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming”. In: *Mathematical programming* 106 (2006), pp. 25–57.