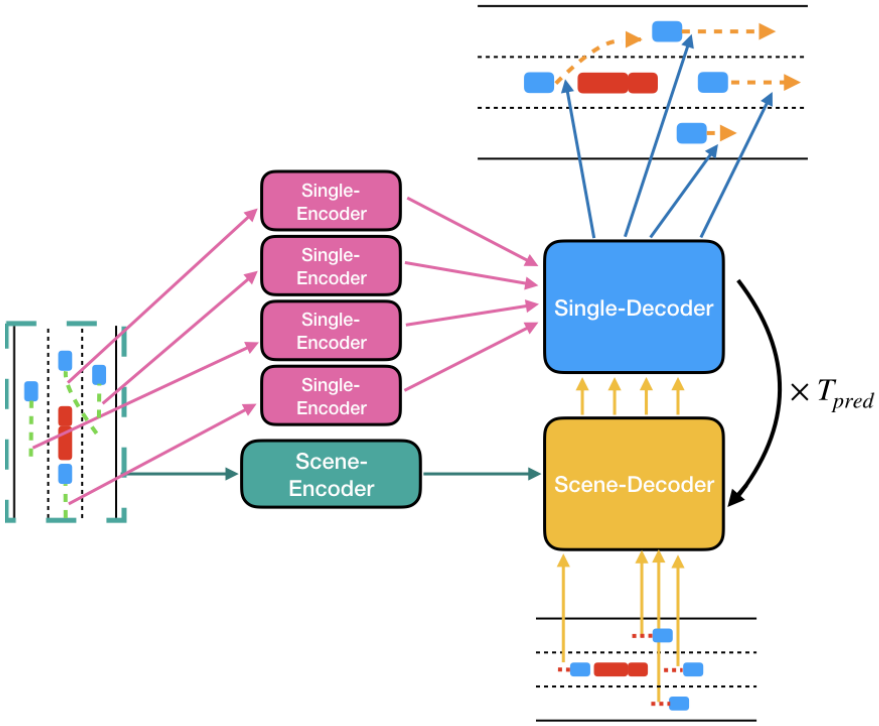
CHALMERS
UNIVERSITY OF TECHNOLOGY



Highway Autopilot using Robust Model Predictive Control with a Transformer based Observer

An exploration into the combination of deep machine learning and optimal control algorithms for trajectory planning

Master's thesis in Systems Control and Mechatronics

Richard Johnsson

Oskar Westerlund

DEPARTMENT OF ELECTRICAL ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2023
www.chalmers.se

Master's thesis 2023

Highway Autopilot using Robust Model Predictive Control with a Transformer based Observer

An exploration into the combination of deep machine learning and optimal control algorithms for trajectory planning

RICHARD JOHNSSON

OSKAR WESTERLUND

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Systems and Control
Mechatronics Research Group
Chalmers University of Technology
Gothenburg, Sweden 2023

Highway Autopilot using Robust Model Predictive Control with a Transformer based Observer
An exploration into the combination of deep machine learning and optimal control algorithms for trajectory planning
RICHARD JOHNSSON & OSKAR WESTERLUND

Supervisors:
Erik Börve, Volvo GTT/Department of Electrical Engineering, Chalmers University of Technology
Leo Laine, Volvo GTT/Department of Mechanics and Maritime Sciences, Chalmers University of Technology
Deepthi Pathare, Volvo GTT
Stefan Börjesson, Volvo GTT

Examiner: Nikolce Murgovski, Department of Electrical Engineering, Chalmers University of Technology

Master's Thesis 2023
Department of Electrical Engineering
Division of Systems and Control
Mechatronics Research Group
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: The developed observer and a display of its I/O relations and modular components.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2023

Highway Autopilot using Robust Model Predictive Control with a Transformer based Observer
An exploration into the combination of deep machine learning and optimal control algorithms for trajectory planning
RICHARD JOHNSSON & OSKAR WESTERLUND
Department of Electrical Engineering
Chalmers University of Technology

Abstract

In this thesis, the area of robust trajectory planning through model predictive control in combination with trajectory predictions is explored. The combination is used to obtain an autopilot algorithm controlling a truck that is operating in a simulation of a highway scenario with surrounding traffic. A novel transformer-based observer has been developed to address the problem of generating trajectory predictions for the robust model predictive controller. The observer is also capable of expressing uncertainty through the Monte Carlo dropout methodology. Further, the model predictive controller extends its collision avoidance constraints based on these uncertainties to increase the robustness of the solution. The results of the developed observer showed that it was capable of giving more accurate multi-modal predictions compared to a constant velocity baseline, in terms of ADE and FDE. Indications for a more robust solution in terms of safety were observed for the robust model predictive controller combined with the developed trajectory predictor. However, the robustness of the new combination came at the cost of reduced drivability performance caused by overcompensation for future trajectory predictions of surrounding vehicles. It was concluded that the results signify the usefulness of the developed combination in safety-critical scenarios and that a foundation for future work has been provided.

Keywords: deep machine learning, model predictive control, optimal control, robust optimal control, transformer, trajectory prediction, trajectory planning, autopilot.

Acknowledgements

Firstly, we would like to acknowledge, with our deepest gratitude, industrial PhD student Erik Börve at Volvo GTT and the Department of Electrical Engineering at Chalmers University of Technology. Erik has been one of our supervisors during the thesis and has provided us with the necessary foundation and knowledge base for the thesis. Furthermore, it is without a doubt that the thesis would not have been possible to perform without his support, guidance and help throughout the project. We would also like to acknowledge and emphasize the significant contributions of our examiner Professor Nikolce Murgovski at the Department of Electrical Engineering at Chalmers University of Technology. Nikolce was the first professor to introduce us to the world of model predictive control at the beginning of our graduate studies. Nikolce was also the main contributor to making the thesis possible through his renowned reputation in the industry. Lastly, we would like to thank our industrial partner Volvo GTT and our industrial supervisors adjunct professor Leo Laine at Chalmers University of Technology and Volvo GTT, industrial PhD student at Volvo GTT Deepthi Pathare and Stefan Börjesson at Volvo GTT. They have provided us with crucial tools, funding and guidance during the project.

Richard Johnsson & Oskar Westerlund, Gothenburg, June 2023

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

ADE	Average Displacement Error
ANN	Artificial Neural Network
BEV	Bird's Eye View
BNN	Bayesian Neural Network
cVAE	conditional Variational AutoEncoder
CV	Constant Velocity
DL	Deep Machine Learning
DM	Decision Master
FDE	Final Displacement Error
GAN	Generative Adversarial Network
GT	Ground Truth
IDM	Intelligent Driver Model
LSTM	Long Short Term recurrent neural network Model
L2	Euclidean Distance
MOBIL	Minimize Overall Breaking Induced by Lane change
ML	Machine Learning
MPC	Model Predictive Control
RMPC	Robust Model Predictive Control
RNN	Recurrent Neural Network
SMPC	Stochastic Model Predictive Control
SUMO	Simulation of Urban Mobility
TF	Transformer
TOT	Trajectory Observer Transformer
TOT-MC	Trajectory Observer Transformer with Monte Carlo dropout

Nomenclature

Below is the nomenclature of indices, parameters, variables, sets and functions that have been used throughout this thesis. Selected mathematical notation:

- Vectors are bold lower case letters, "$\mathbf{x}$"
- Matrices are bold upper case letters, "$\mathbf{G}$"
- Sets are in blackboard-bold letters, "$\mathbb{S}$"
- Predictions are marked by $\wedge$ and time derivatives by $\bullet$

Indices

(i)	Reference for vehicle i surrounding the ego vehicle
e	Ego vehicle reference
t	Time step index
n	Discrete time steps in optimization horizon
(l)	Reference for leading vehicle l in front of ego vehicle
ν	Reference index for left lane change and right lane change controllers
κ	Reference index for trailing, left and right lane change controllers
k	Index for head in multi-headed attention
R	Reference index for robust implementation
q	Index of trajectory prediction mode

Parameters

L	Length
$s_{\min}$	Minimum value of state limitations on the truck
$s_{\max}$	Maximum value of state limitations on the truck
$u_{\min}$	Minimum value of control actions for the truck

$u_{\max}$	Maximum value of control actions for the truck
$a_{\max,x}$	Maximum longitudinal acceleration of surrounding vehicle
γ	Acceleration exponent
$v_{\text{ref},x}$	Desired speed
b_d	Desired deceleration
T_s	Time headway needed to be safe for surrounding vehicle
d_0	Minimum allowed longitudinal distance between vehicles
p_a	Politeness factor
a_{safe}	Maximum allowed deceleration
$\Delta a_{\text{thresh},x}$	Acceleration threshold for when to initialize a lane change
k_p	Proportional gain in PI controller
k_I	Integral gain in PI controller
k_n	Proportional gain corresponding to close target in PI controller
k_f	Proportional gain corresponding to far away target in PI controller
T_{pred}	Last value time step in the prediction horizon
T	Time headway needed to be safe for ego vehicle
w	Maximum lateral span of truck
d	Desired minimum distance to vehicle in front
$\mathbf{Q}_s$	State cost
$\mathbf{R}$	Control cost
$\mathbf{P}_f$	Terminal state cost
a_1	Scaling parameter for controller cost
a_2	Scaling parameter for cost of changing controller decision
T_{obs}	First observed historical time step
N	Number of vehicles surrounding the ego vehicle
A	Number of vectors in the input sequence in a self attention module
d_{model}	Design parameter in transformer

$\mathbf{I}^O$	Linear projection matrix of learnable parameters for the concatenated output of multiheaded attention
$\mathbf{I}^Q$	Linear projection matrix of learnable parameters for the Queries of multiheaded attention
$\mathbf{I}^K$	Linear projection matrix of learnable parameters for the Keys of multiheaded attention
$\mathbf{I}^V$	Linear projection matrix of learnable parameters for the Values of multiheaded attention
t_s	Sampling time
t_f	Maximum simulation time
$d_{\max}$	Maximum allowed distance
N_s	Total number of simulations
m_x	Size of x quantified positional grid surrounding the ego vehicle
m_y	Size of y quantified positional grid surrounding the ego vehicle
$m_{\dot{x}}$	Size of x quantified velocity grid
$m_{\dot{y}}$	Size of y quantified velocity grid
N_m	Number of predicted modes in TOT-MC
τ	Number of heads in multi-headed attention module
ϕ	Scaling parameter for extending constraint based on uncertainty

Variables

x	Longitudinal position
y	Lateral position
$\mathbf{s}$	State vector
$\mathbf{u}$	Control vector
v_x	Longitudinal velocity
θ	Vehicle angle in internal frame
δ	Steering angle
a_x	Longitudinal acceleration
Δv_x	Longitudinal velocity difference between two vehicles

d_x	Longitudinal distance between two vehicles
$a_{fr,x}$	Acceleration on free road
$a_{\text{dec},x}$	Deceleration due to vehicle in front
$\Delta a_{\text{cur},x}$	Change in longitudinal acceleration for current following vehicle
$\Delta a_{\text{new},x}$	Change in longitudinal acceleration for new following vehicle
$\Delta a_{d,x}$	Change in longitudinal acceleration for a vehicle conducting a lane change
λ	Direction of target
ϵ_1	Longitudinal constraint scaling variable
ϵ_2	Longitudinal constraint scaling variable
$\mathbf{l}$	Scene context information
$\mathbf{s}_{\text{ref}}$	State reference vector
$\mathbf{u}_{\text{ref}}$	Control reference vector
$J_{\kappa,\text{tot}}$	Cost that lie as a basis for which controller to choose in the baseline MPC
$\mathbf{K}$	Keys in transformer
$\mathbf{Q}$	Queries in transformer
$\mathbf{V}$	Values in transformer
$\mathbf{C}$	Compatibility value of all data points in a sequence
$\mathbf{W}$	Compatibility value of all data points in a sequence run through a SoftMax function
$\mathbf{E}$	New embedding for Queries, Keys and Value combination
h	Head in multihead-attention mechanism
$\mathbf{M}$	Multihead-attention function
$\mathbf{p}$	Grid points for state space quantization
$\mathbf{g}_{\text{obs}}$	Labeled observed historical positions
$\mathbf{E}_{\text{obs}}$	d_{model} higher dimensional representation for labeled positions from $\mathbf{g}_{\text{obs}}$
$\mathbf{Z}_{\text{scene}}$	Latent variable representation produced by scene-encoder
$\mathbf{Z}_{\text{single}}$	Latent variable representation produced by single-encoder
$\dot{\mathbf{g}}_{\text{GT}}$	Labeled ground truth future velocities

$\dot{\mathbf{E}}_{\text{GT}}$	d_{model} higher dimensional representation for labeled velocities from $\dot{\mathbf{g}}_{\text{GT}}$
ζ	Slack variable for constraints
α_x	Uncertainty in predicted x positions from the TOT-MC model
α_y	Uncertainty in predicted y positions from the TOT-MC model
μ	Design parameter for positional encoding resolution
$J_{\kappa,\text{tot},R}$	Cost that lie as a basis for which controller to choose in the Robust MPC

Sets

$\mathbb{P}_{\text{obs}}$	The set containing all agents historical x and y coordinates from time step T_{obs} to current time step, i.e. $\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{N}$
$\mathbb{P}_{\text{pred}}$	The set containing all predicted positions, i.e $\{(\hat{\mathbf{x}}^{(i)},\ \hat{\mathbf{y}}^{(i)})\}_{i=1}^{N}$
$\mathbb{L}_{\text{obs}}$	The set containing all previous agents observed scene information, i.e. $\{\mathbf{l}^{(i)}\}_{i=1}^{N}$
$\mathbb{P}_{\text{pos}}$	The set which contains the grid point coordinates of the quantified area surrounding the ego vehicle
$\mathbb{V}_{\kappa}$	The set of vehicles i considered by the controller κ
$\mathbb{P}_{\text{vel}}$	The set which contains the quantified combinations of x and y velocities

Functions

$\rho(v_x, \Delta v_x)$	Allowed minimum distance between the vehicle in consideration and the vehicle in front
$\epsilon_0^{(i)}(\hat{y}^{(i)})$	Lateral constraint scaling
$\epsilon_3^{(i)}(\hat{y}, y^e)$	Shifts constraint laterally
$h_\kappa^e(\mathbf{s}_\kappa^e, \mathbf{u}_\kappa^e, \hat{x}^{(i)}, \hat{y}^{(i)})$	MPC baseline constraints including physical constraint on the ego vehicles

$J_\kappa\left(\mathbf{s}^e_{\text{init}}, \mathbf{u}^e_\kappa(t:t+T_{\text{pred}}-1\|t)\right)$	Cost function for optimization
$\Pi_\psi(\mathbf{x},\ \mathbf{y})$	Trajectory predictor with parameters ψ
$g(\mathbf{s}(t), \mathbf{u}(t))$	Plant model
$p(\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^N, \mathbb{P}_{\text{pos}})$	Euclidian distance label assignment function for positional quantization
$f(\{\mathbf{g}^{(i)}_{\text{obs}}\}_{i=1}^N, m_x, m_y)$	Embedding function for higher dimensional representation
$o(\{(\dot{\mathbf{x}}^{(i)}_{\text{GT}}, \dot{\mathbf{y}}^{(i)}_{\text{GT}})\}_{i=1}^N, \mathbb{P}_{\text{vel}})$	Euclidian distance label assignment function for velocity quantization
$\mathbf{h}^e_{\kappa,R}(\mathbf{s}^e_\kappa, \mathbf{u}^e_\kappa, \hat{x}^{(i)}, \hat{y}^{(i)}, \alpha_x^{(i)}, \alpha_y^{(i)}, \zeta^{(i)})$	Robust MPC constraint function including physical constraints on the ego vehicle

Contents

List of Figures

List of Tables

1

Introduction

More than one million people die in traffic accidents every year [1]. Rarely, these accidents have a root cause, which has a basis in the complexity of the driving scenarios. Although, human factors such as inattention are identified as the cause of the majority of traffic accidents [2]. Developing safe autonomous vehicles that mitigate dangerous human behaviors is key to reducing the number of traffic accidents. The main challenge of autonomous vehicles is decision-making in complex environments such as changing lane on a highway or crossing an intersection. With increased automation and companies, such as Volvo Group, working towards zero accidents [2], it is of the utmost importance to develop cutting-edge trajectory planning algorithms that can guarantee safe operation.

1.1 Background

From the perspective of autonomous driving and trajectory planning algorithms, the capabilities of accurately predicting other objects' future states such as pedestrians, traffic flow and surrounding vehicles, can help improve the performance of decision-making [3]. It is important in trajectory predictions to not only model the object's own properties but also how it interacts with and affects its environment. This could include contextual information (city environment, highway scenario) and scene information, such as social interaction between other surrounding vehicles.

In terms of one prediction model to rule them all, there is none, and several strategies exist on the market today. The prediction model domain spans wide from older models that try to incorporate physics with examples of Newton's law to model future motion to models built by learning from data via supervised learning. Narrowing down the perspective to predicting future trajectories of vehicles and objects such as pedestrians, several prominent supervised and unsupervised learning-based models exist [4, 5]. Supervised learning can be summarized as the process of learning from data where input is matched toward the correct output. One example of this is to produce a prediction of a future trajectory by conditioning on previous observations. The model then tries to predict this trajectory and then optimize its variables depending on how erroneous it was compared to the truth. For unsupervised learning such as reinforcement learning (RL), the training is often performed through active environment learning and the maximization of a predefined reward function.

Before 2017 models based on long short-term memory (LSTM) [6], gated recurrent unit (GRU) [7], convolution neural networks (CNN) [8] and graph neural networks (GNN) [9] were regarded as state-of-the-art for sequence modeling, especially for trajectory predictions. In 2017 a new way of modeling sequential data was introduced for natural language processing (NLP) tasks, the transformer network [10]. What the transformer mainly differs from previous models is that it is based on self-attention. The self-attention mechanism in the transformer structure results in a new way of processing that moves away from a sequential perspective, to reviewing the whole input directly. Attention mechanisms increase the capabilities of capturing long-term dependencies which allows for a higher degree of parallel training. This facilitates possibilities for producing faster and more accurate tools for trajectory forecasting with better deployment of graphics processing units (GPU) [10].

The predictions of the surrounding environment lie as a base for how an autonomous vehicle should act. Thus, good predictions are a necessity for safe autonomous decision-making. What is important to note and raise among these supervised Deep learning (supervised DL) and unsupervised RL models is how to guarantee safety if applied to an autonomous vehicle environment. Purely learning-based approaches have achieved promising results in terms of decision-making in complex environments but often lack robust safety constraints [11]. One idea here is to incorporate and fuse the strengths of learning-based models' capabilities of modeling complex environments with optimization-based models to guarantee safety constraints. One of these optimization-based models is the model predictive controller (MPC). By utilizing the strengths of the MPC's optimal decision-making and robust constraint modeling together with a learning-based observer for trajectory predictions the hypothesis is that it can generate a multi-sided agent that can take robust decisions and optimal actions.

1.2 Aim

This thesis aims at combining these two approaches and make use of their individual strengths implemented in a highway scenario for a truck. To achieve this, the objective is to develop a cutting-edge learning-based observer for environment analysis surrounding the ego vehicle and construct future predictions of these vehicles' trajectories as a base for decision-making. Following the predicted trajectories an MPC will provide optimal control decisions with robust consideration to the surrounding vehicles. With the development of this combined approach, the project aims at presenting results for this relatively unexplored area and evaluating its future potential.

1.3 Limitations

Given the aforementioned aim of the thesis to generate an autopilot for a highway scenario based on the combination of a learning-based observer together with a model predictive controller, certain limitations are considered. The specified learning-based observer will be based on supervised learning and mainly focus on models based on this strategy. This demarcation will therefore reduce the thesis focus on unsupervised learning models such as RL, which will not be explored. Moving to the trajectory planning algorithm, the project will not aim a developing an MPC from scratch. It will rather focus on extending the robustness of an already implemented trajectory planning algorithm. Therefore, an in depth review of MPC will not be provided, and the interested reader is referred to the sources presented in the thesis. The project environment for data generation and testing is based on supervisor Erik Börve's implementation of SUMO [12] driving environment models. A version of this environment can be accessed via his publicly available GitHub repository: [13]. The selection to use this environment is deemed essential in the perspective of time resources to enable the project scope of conducting research on model predictive control and supervised learning combinations. Using this environment puts limitations towards only using a bird's eye view (BEV) environment, thus taking a step away from real-world implementations. The thesis is based on a relatively unexplored research area and will focus on investigating potential implementation without the perspective of real-world applications. This limitation is further extended into the perspective of real-world computing times for the implemented solutions, which will not be considered in this thesis due to its conceptual focus. Important to note is though that the data may not be real-world comparable, driver models and agents interactions are constructed towards mimicking real-world interactions and actions.

1.4 Research questions

The established research questions that are to be investigated during the project are

- Is the developed observer able to capture the behavior of surrounding vehicles in a highway scenario and to what extent?
- How can a learning-based observer and MPC be combined to increase robustness in a trajectory planning algorithm for an autonomous truck in a highway scenario?
- What are the main benefits that can be gained by using the aforementioned combination from the perspective of drivability and safety?

1.5 Thesis overview

Chapter 2 presents the specific problem that is to be investigated together with the basis of the simulation environment in which the research is conducted in. The necessary theoretical background for the thesis is presented in chapter 3. The methodology for the developed trajectory prediction algorithm, i.e. observer, is presented in chapter 4 and the robust trajectory planner in chapter 5. Results for the evaluation of the observer and the robust controller are located in chapter 6. Discussions and recommendations for future work are presented in chapter 7 with final conclusions in chapter 8.

2

Vehicle modeling and problem formulations

The simulation environment used in this thesis is based on a simulation environment called Simulation of Urban Mobility (SUMO) [12], which is an open-source multi-modal simulation environment designed for the training and testing of autonomous vehicle implementations. A version of the simulation environment and the baseline ego vehicle controller used in this thesis can be viewed in our supervisor Erik Börve's GitHub repository [13]. The environment is a three-lane highway with the ego vehicle, surrounded by 5 other passenger vehicles. A typical simulation can be seen in Figure 2.1.

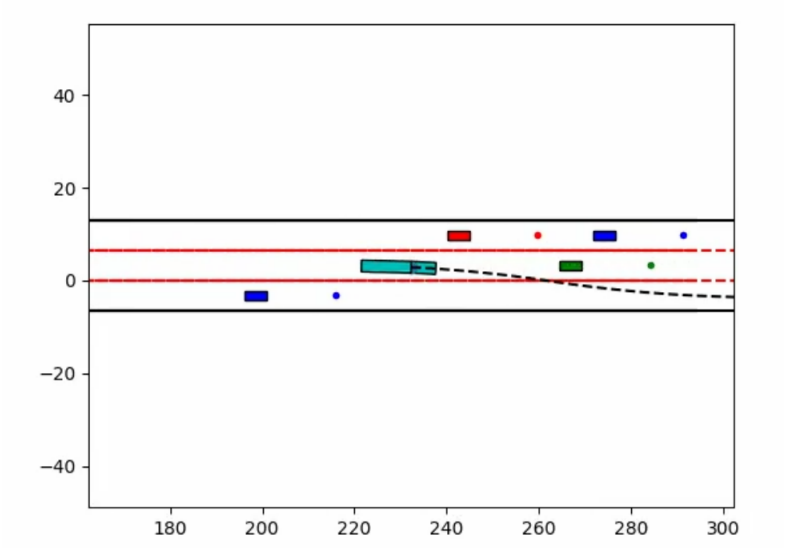


Figure 2.1: Simulation environment with four visible surrounding vehicles (red, blue and green) and truck and trailer combination as the ego vehicle (teal). The y-axis represents the lateral distance in meters and points upwards while the x-axis represents the longitudinal distance in meters and points in the right direction. The lane markings are represented by red solid lines, the planned trajectory for the ego vehicle is represented by the dashed black line, and the mixed color dots in front of the surrounding traffic represents their upcoming lane.

In the sections below is the functionality of this simulation environment presented. In section 2.1 the two vehicle models for the surrounding traffic and ego vehicle

are presented. The driver model for traffic around the ego vehicle is presented in section 2.2. The chapter also presents a description of the used trajectory planning algorithm for the ego vehicle and its problem statement in section 2.3. Following this statement, an introduction to the problem statement of predicting surrounding vehicles' trajectories is presented in section 2.4.

2.1 Vehicle models

A bicycle model is used to model the vehicles surrounding the ego vehicle. A schematic view of the model can be seen in Figure 2.2.

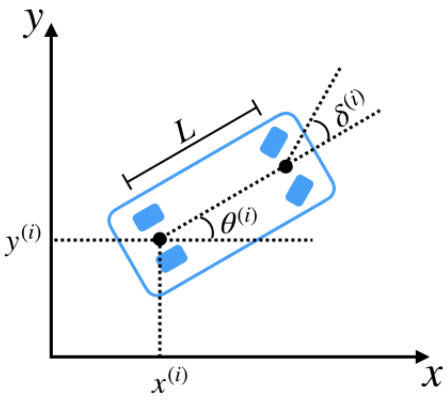


Figure 2.2: Vehicle model for the surrounding traffic where superscript (i) refers to which vehicle, L is the length, $x^{(i)}$ and $y^{(i)}$ are coordinates represented in the middle of the rear axis, $\theta^{(i)}$ is the angle of the vehicle in regards to the internal frame and $\delta^{(i)}$ is the steering angle.

The corresponding state representation of each vehicle, $\dot{\mathbf{s}}^{(i)}$, is described by

$$\dot{\mathbf{s}}^{(i)} = \begin{bmatrix} \dot{x}^{(i)} \\ \dot{y}^{(i)} \\ \dot{v}^{(i)} \\ \dot{\theta}^{(i)} \end{bmatrix} = \begin{bmatrix} v_x^{(i)} \\ v_x^{(i)} \tan(\theta^{(i)}) \\ a_x^{(i)} \cos(\theta^{(i)}) \\ v_x^{(i)} \frac{\tan(\delta^{(i)})}{L \cos(\theta^{(i)})} \end{bmatrix} \tag{2.1}$$

where $x^{(i)}$ and $y^{(i)}$ are coordinates represented in the middle of the front axis for vehicle i. $v_x^{(i)}$ is the longitudinal speed, $a_x^{(i)}$ is the acceleration along the global x-axis, $\theta^{(i)}$ is the angle of the vehicle in regards to the internal frame, and $\delta^{(i)}$ is the steering angle. The length of the vehicle is given by L, which is the same for all surrounding vehicles. The controllable states are the steering angle, $\delta^{(i)}$, for lateral

control and the acceleration, $a_x^{(i)}$, for longitudinal control. The control vector for each vehicle is therefore

$$\mathbf{u}^{(i)} = \begin{bmatrix} u_{\text{lat}}^{(i)} \\ u_{\text{long}}^{(i)} \end{bmatrix} = \begin{bmatrix} \delta^{(i)} \\ a_x^{(i)} \end{bmatrix} \tag{2.2}$$

where $u_{lat}^{(i)}$ and $u_{long}^{(i)}$ represents lateral and longitudinal control respectively.

The ego vehicle is a tractor with a trailer combination. It is modeled by an extension of the proposed bicycle model in (2.1) to give an accurate representation of the ego vehicle. A schematic view of the model for the tractor-trailer combination can be seen in Figure 2.3

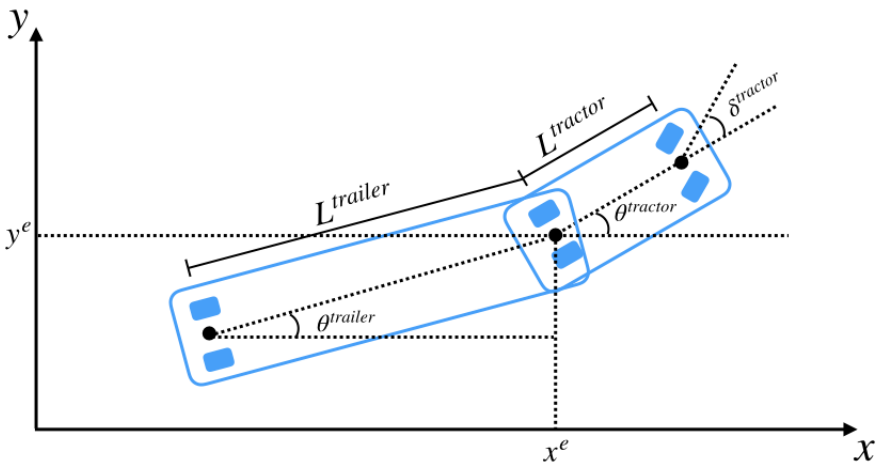


Figure 2.3: The ego vehicle tractor-trailer combination model where superscript e refers to the ego vehicle, L^{tractor} and L^{trailer} corresponds to the length of the tractor and the trailer, x^e and y^e represent coordinates in the joint of the tractor and the trailer in the internal frame, θ^{tractor} and θ^{trailer} represent the angles of the tractor and trailer and δ^{tractor} is the steering angle of the tractor.

The state representation for the ego vehicle, $\dot{\mathbf{s}}^e$, can be seen below

$$\dot{\mathbf{s}}^e = \begin{bmatrix} \dot{x}^e \\ \dot{y}^e \\ \dot{v}_x^e \\ \dot{\theta}^{\text{tractor}} \\ \dot{\theta}^{\text{trailer}} \end{bmatrix} = \begin{bmatrix} v_x^e \\ v_x^e \tan(\theta^{\text{tractor}}) \\ a_x^e \cos(\theta^{\text{trailer}}) \\ v_x^e \frac{\tan(\delta^{\text{tractor}})}{L^{\text{tractor}} \cos(\theta^{\text{tractor}})} \\ v_x^e \frac{\sin(\theta^{\text{tractor}} - \theta^{\text{trailer}})}{L^{\text{trailer}} \cos(\theta^{\text{tractor}})} \end{bmatrix} \tag{2.3}$$

where x^e and y^e represent coordinates in the joint of the tractor and the trailer in the internal frame. v_x^e represents the longitudinal velocity. θ^{tractor} and θ^{trailer} represent the angles of the tractor and trailer respectively seen from the internal frame. L^{tractor} and L^{trailer} corresponds to the length of the tractor and the trailer and δ^{tractor} is the steering angle of the tractor. The controllable states for the ego vehicle are

$$\mathbf{u}^e = \begin{bmatrix} u^e_{\text{lat}} \\ u^e_{\text{long}} \end{bmatrix} = \begin{bmatrix} \delta^{\text{tractor}} \\ a^e_x \end{bmatrix} \tag{2.4}$$

where u^e_{lat} and u^e_{long} represent lateral and longitudinal control respectively for the ego vehicle. The limitations on the truck, trailer, and inputs are represented as box constraints

$$\mathbf{s}^e_{\min} \leq \mathbf{s}^e \leq \mathbf{s}^e_{\max} \tag{2.5a}$$

$$\mathbf{u}^e_{\min} \leq \mathbf{u}^e \leq \mathbf{u}^e_{\max} \tag{2.5b}$$

Where $\mathbf{s}^e_{\min}$ and $\mathbf{s}^e_{\min}$ are the lower limits and $\mathbf{s}^e_{\max}$ and $\mathbf{u}^e_{\max}$ are the upper limits for the ego vehicle.

2.2 Driver model of surrounding vehicles

Three different models are used to simulate the drivers in the surrounding traffic around the ego vehicle. One is for controlling the longitudinal speed, one for lateral control, and one that decides upon lane change. The superscript notation to represent each vehicle i is omitted in this section for brevity in the equations.

To control the longitudinal speed of the surrounding vehicles, the model Intelligent Driver Model (IDM) is used [14]. The idea is to use their acceleration function to control the speed. IDM assumes that the acceleration of the vehicle is a constant function with respect to the velocity, the distance, and the difference in velocity between the vehicle in consideration and the vehicle in front of it. The equation for the acceleration is formulated as

$$a_x = a_{\max,x}\left(1 - \frac{(v_x)^{\gamma}}{v_{\text{ref},x}} - \frac{\rho(v_x, \Delta v_x)^2}{d_x}\right) \tag{2.6}$$

where $a_{\max,x}$ refers to the maximum allowed longitudinal acceleration, a_x is the current longitudinal acceleration, v_x is the vehicle speed, Δv_x is the velocity difference between the vehicles, d_x is the distance between the vehicles, $v_{\text{ref},x}$ corresponds to the desired speed and γ is the acceleration exponent. The two last-mentioned parameters are tunable. Equation (2.6) can be explained by the interpolation of two components, acceleration with no vehicle in front of it

$$a_{fr,x} = a_{\max,x}\left(1 - \frac{(v_x)^{\gamma}}{v_{ref,x}}\right) \tag{2.7}$$

and the deceleration, $a_{dec,x}$, which represents a deceleration due to the leading vehicle being closer than what is desired

$$a_{dec,x} = \left(-a_{\max,x} \frac{\rho(v_x, \Delta v_x)}{d_x}^2 \right). \tag{2.8}$$

The equation for the allowed minimum distance between the vehicle in consideration and the vehicle in front, $\rho(v_x, \Delta v_x)$, is

$$\rho(v_x, \Delta v_x) = \frac{v_x \Delta v_x}{2\sqrt{a_{\max,x} b_d}} + v_x T_s + d_0 \tag{2.9}$$

where b_d is the desired deceleration, T_s is the time headway needed to be safe and d_0 is the minimum allowed longitudinal distance between the vehicles. All these parameters are tunable.

The model Minimize Overall Breaking Induced by lane change (MOBIL) [15] is used to commence lane changes for the surrounding vehicles. The change in acceleration of the surrounding vehicles is approximated using the aforementioned IDM. A threshold of gained acceleration is defined to motivate a lane change if the estimated change in acceleration exceeds it. This leads to an incentive condition that is used to define when lane change should be initiated

$$\Delta a_{\text{thresh},x} < p_a \left(\Delta a_{\text{cur},x} + \Delta a_{\text{new},x} \right) + \Delta a_{d,x} \tag{2.10}$$

where $\Delta a_{\text{thresh},x}$ defines a threshold for gained acceleration when to initialize the lane change, $\Delta a_{\text{cur},x}$ and $\Delta a_{\text{new},x}$ is the change in acceleration for the current and new follower of the vehicle pending a lane change respectively and $\Delta a_{d,x}$ is the change in acceleration for the vehicle conducting a lane change. p_a represents a politeness factor of the closest neighboring vehicles stating to what extent the following vehicles should impact the lane change. A politeness factor of 0 means that the driver deciding on a lane change does not care at all about surrounding traffic when deciding upon lane change. Moreover, another measure is taken to minimize the risk of collision by defining the maximum allowed deceleration of the vehicles, a_{safe}, which is a tunable parameter. A lane change will commence if the criterion (2.10) is fulfilled and the deceleration of the vehicles all considered vehicles is greater than a_{safe}.

To control the vehicles laterally, a two-point visual control model derived from a PI controller is used as defined in [16]. The central factor of this model is that it can obtain stable control of the steering wheel angle by utilizing observed targets both close to and far away from the controlled vehicle. The equation for the PI controller can be defined as,

$$\delta = k_p \lambda + k_I \int \lambda \, dt \tag{2.11}$$

where λ is the direction of the target, k_p and k_I is the proportional and integral gain. Applying the derivative to (2.11) and extending it with the dual targets gives,

$$\dot{\delta} = k_n \dot{\lambda}_n + k_f \dot{\lambda}_f + k_I \lambda_n \tag{2.12}$$

where $\dot{\lambda}_n$ and $\dot{\lambda}_f$ describe the change in the direction of both the close-by target and the far away target respectively, λ_n symbolizes the direction to the close target. Euler's method is used to discretize (2.12) and obtain the steering wheel angle for the controlled vehicle.

2.3 Problem formulation for trajectory planning

In the past two decades, MPC has shown prominent results when dealing with the control of complex dynamical systems [17]. The popularity of the MPC comes from its ability to cope with complicated dynamics, multiple inputs and outputs, and its profound potential to perform optimal control actions given a set of constraints on inputs and states [18]. As a consequence, literature has shown that it is an effective tool to plan trajectories for autonomous vehicles [19, 20].

Model predictive control uses a model of the system to predict its future states depending on an applied control signal throughout a prediction horizon. Utilizing a user-defined cost function J, MPC aims to minimize it along the entire prediction horizon (T_{pred}), while satisfying a set of constraints. In terms of trajectory planning, these constraints can be physical limitations on the ego vehicle or constrained positions for surrounding vehicles. A specific MPC trajectory planning algorithm for a highway scenario can be seen in Figure 2.4.

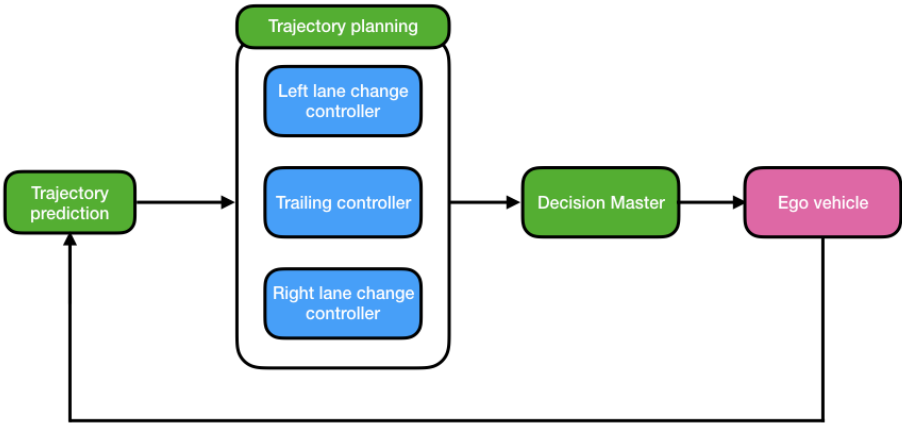


Figure 2.4: Highway autopilot consisting of a trajectory planning algorithm with three separate model predictive controllers for trailing and lane change trajectories. The trajectory planner utilizes a trajectory predictor for predicting trajectories of surrounding vehicles. The optimal cost-minimizing trajectory for the ego vehicle is chosen by a Decision Master

In Figure 2.4 one can see that the controller architecture uses a predictor, three controllers, and a Decision Master (DM) to plan a future trajectory. The predictor is used to predict the positions of surrounding vehicles throughout the prediction horizon to be able to form collision avoidance constraints. Based on the predictions,

three different MPC trajectory plans are generated where two are considered for lane change (left and right) and one is considered for a trailing trajectory. The trajectories are then compared to each other in terms of cost via a DM. The DM returns a safe cost-minimizing solution, whether it is to change lanes or continue trailing the vehicle in front. The algorithm is carried out at every time instance, thus forming a receding horizon control that inherently provides some robustness to prediction inaccuracies [21]. This specific algorithm follows similar applications as in the papers [22] and [23] and is implemented in our supervisor Erik Börve's GitHub repository [13]. The specific components will be further explained in the coming sections.

2.3.1 Constraint formulation

In order to obtain feasible solutions, constraints are constructed for the surrounding vehicles. The constraints are based on calculated trajectories from each model predictive controller at each time step. Through the prediction horizon, T_{pred}, the positions of the surrounding vehicles need to be estimated so that the constraints could be formed therein. In the baseline version, a constant velocity (CV) model is used to estimate surrounding vehicles' future x positions

$$\hat{x}^{(i)}(t+1) = \hat{x}^{(i)}(t) + \dot{x}^{(i)}(0)t_s,\ \hat{x}^{(i)}(0) = x^{(i)}(0), \tag{2.13}$$

$\hat{x}^{(i)}(t+1)$ is the predicted x position for time step $t+1$ given previous prediction $\hat{x}^{(i)}(t)$. Note that the prediction starts from the current position $\hat{x}^{(i)}(0) = x^{(i)}(0)$ and that the velocity, $\dot{x}^{(i)}(0)$, is kept constant for all predictions thus forming a constant velocity model. The y position for each vehicle is assumed to be constant during the prediction horizon. The predicted positions of the vehicles are used to form constraints that dismiss MPC solutions leading to a collision. Two different structures of constraints are used, one for the trailing MPC and one for the two lane-change MPCs. Note that in the collision avoidance constraints, t has been omitted for brevity.

Firstly, for the trailing MPC, the constraints constitute of holding a specified safe longitudinal distance from the vehicle ahead of the truck in the same lane

$$c_{\text{trail}}(x^e, \hat{x}^{(l)}, \hat{v}_x^{(l)}) = d + L^{\text{tractor}} + T\hat{v}_x^{(l)} + x^e - \hat{x}^{(l)} < 0 \tag{2.14}$$

where e is a reference for the ego vehicle, l reference for the leading vehicle, x^e is the longitudinal positions of the ego vehicle and $\hat{x}^{(l)}$ is the predicted x position of the leading vehicle. T represents the minimum required headway in time for a possible deceleration and d represents the desired minimum longitudinal distance to the vehicle in front in the same lane. Furthermore, the trailing MPC solution is also constrained laterally by the lane markings and the maximum lateral span of the truck, w, to avoid solutions deviating too much into adjoining lanes. A scenario where the MPC trailing constraint is present can be seen in Figure 2.5.

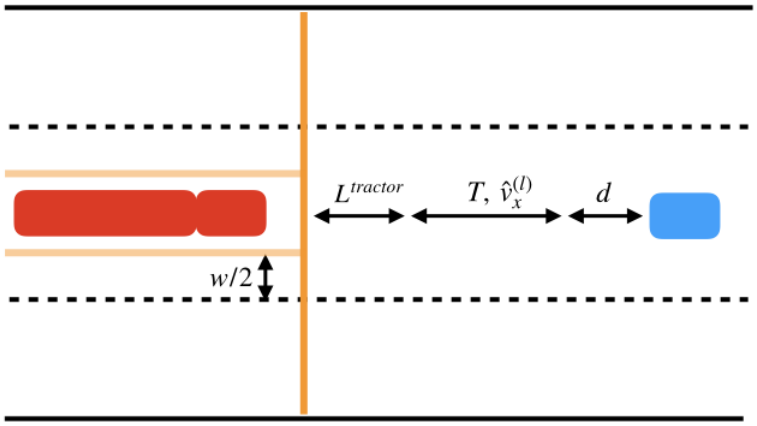


Figure 2.5: Trailing constraint where the ego-vehicle is depicted in red and the leading vehicle is depicted in blue. The longitudinal constraint, depicted in orange, is determined by holding a specified safety distance that is dependent on the length of the tractor L^{tractor}, the time headway T, the predicted velocity of the leading vehicle $\hat{v}_x^{(l)}$, and a required minimum longitudinal distance to the vehicle in front d. The ego vehicle is constrained laterally by the lane markings including a safety distance of maximal lateral span of the truck $w/2$.

In the case of the two lane-change MPCs, the distances to surrounding vehicles are constrained both laterally and longitudinally. The boundaries of the constraints are formulated as tanh-functions in a continuous form

$$b^{(i)}(x^e, y^e, \hat{x}^{(i)}, \hat{y}^{(i)}) = \xi_1(\xi_2 + \xi_3) + \xi_4 \tag{2.15a}$$

$$\xi_1 = \frac{\epsilon_0^{(i)}(\hat{y}^{(i)})}{2} \tag{2.15b}$$

$$\xi_2 = \tanh(x^e - \hat{x}^{(i)} + \epsilon_1^{(i)}) \tag{2.15c}$$

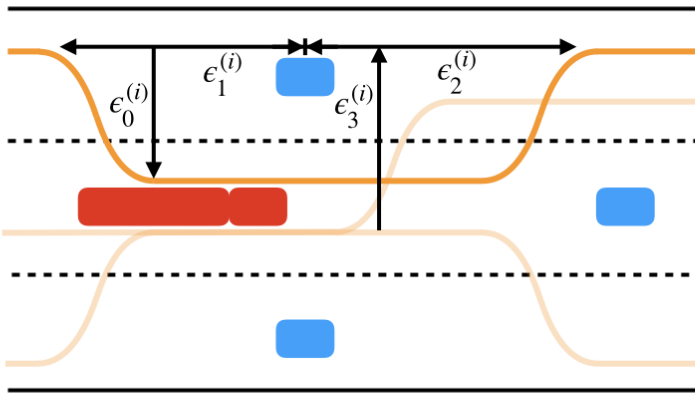
$$\xi_3 = \tanh(\hat{x}^{(i)} - x^e + \epsilon_2^{(i)}) \tag{2.15d}$$

$$\xi_4 = \epsilon_3^{(i)}(y^e, \hat{y}^{(i)}) \tag{2.15e}$$

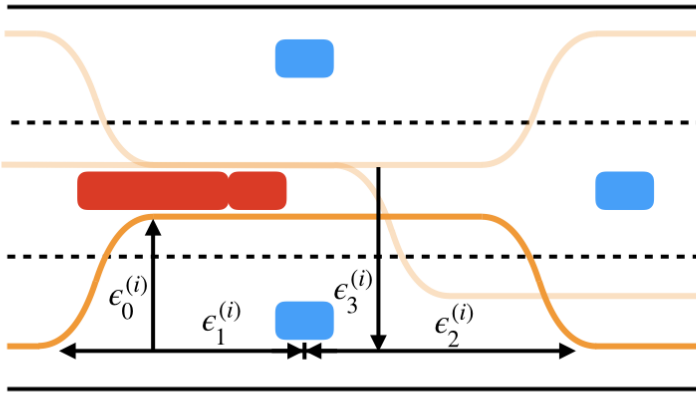
where the constraints are laterally scaled by the parameter $\epsilon_0^{(i)}$ which depends on the planned y-position of the ego vehicle including its maximum lateral span and safety margin and the predicted y-position of the surrounding vehicle i including its width. Furthermore, scaling the constraints in the longitudinal direction is done by the parameters $\epsilon_1^{(i)}$ and $\epsilon_2^{(i)}$, for vehicle i and incorporates a defined safety margin and its length. $\epsilon_3^{(i)}$ places the constraint laterally based on the predicted y position of vehicle i and the planned y position of the ego vehicle. $F_\nu^{(i)} \in \{-1, 1\}$ mirrors the constraints pending on which lane the ego vehicle is located and which vehicle is observed. It also considers which controller is being optimized, referenced as index $\nu = \{\text{left change},\ \text{right change}\}$. Consequently, the lane change collision avoidance constraints are

$$c_\nu(x^e, y^e, \hat{x}^{(i)}, \hat{y}^{(i)}) = F_\nu^{(i)} \left(b^{(i)}(\cdot) - y^e + w \right) < 0 \tag{2.16}$$

where the dependency variables in $b^{(i)}$ has been omitted for a more clear representation. The constraints (2.16) are assembled around all surrounding vehicles, i, for all time steps in the prediction horizon. Two scenarios where the constraints for one time step are visualized for left and right lane change MPCs can be seen in Figures 2.6a and 2.6b respectively.



(a) Left lane change constraints.



(b) Right lane change constraints.

Figure 2.6: Lane change constrains where the ego-vehicle is depicted in red and the surrounding vehicles are depicted in blue. The constraint, depicted in orange for vehicle i is defined by a continuous tanh function that is scaled longitudinally by $\epsilon_1^{(i)}$ and $\epsilon_2^{(i)}$. $\epsilon_0^{(i)}$ scales the constraint laterally and $\epsilon_3^{(i)}$ shifts the constraint to the correct lateral position on the highway.

2.3.2 Formulation of the optimization and definition of decision master

The different MPCs have the same objective, to track a lateral position and a longitudinal speed reference. Another goal is to lessen the needed control input. As the defined objective is the same for the different MPCs, the same cost function is used.

$$\begin{aligned} J(\mathbf{s}^e, \mathbf{u}^e) = L_f(\mathbf{s}^e(T_{\text{pred}})) + \sum_{t=0}^{T_{\text{pred}}-1} (\mathbf{s}^e(t) - \mathbf{s}^e_{\text{ref}}(t))^T \mathbf{Q}_s (\mathbf{s}^e(t) - \mathbf{s}^e_{\text{ref}}(t)) \\ + \sum_{t=0}^{T_{\text{pred}}-1} (\mathbf{u}^e(t) - \mathbf{u}^e_{\text{ref}}(t))^T \mathbf{R} (\mathbf{u}^e(t) - \mathbf{u}^e_{\text{ref}}(t)) \end{aligned} \tag{2.17}$$

where $L_f(\mathbf{s}^e(T_{\text{pred}}))$ is equal to the terminal state cost

$$L_f(\mathbf{s}^e(T_{\text{pred}})) = \left(\mathbf{s}^e(T_{\text{pred}}) - \mathbf{s}^e_{\text{ref}}(T_{\text{pred}})\right)^T \mathbf{P}_f \left(\mathbf{s}^e(T_{\text{pred}}) - \mathbf{s}^e_{\text{ref}}(T_{\text{pred}})\right) \tag{2.18}$$

and $\mathbf{Q}_s$ and $\mathbf{R}$ are weighting matrices that penalize deviation from the state and control reference respectively

$$Q_s = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 40 & 0 & 0 & 0 \\ 0 & 0 & 300 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 5 \end{bmatrix}, R = \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix}. \tag{2.19}$$

$\mathbf{P}_f$ is the positive definite terminal cost matrix. The terminal cost matrix is acquired by the usage of an infinite horizon linear quadratic regulator and solving the discrete-time Ricatti equation for linearized dynamics around the current references for the ego vehicle. The reader is referred to [24] for more detailed information regarding the computation of the terminal cost matrix. Since the goal of the MPCs is to track a lateral reference position and a reference longitudinal speed while lowering the control input, the reference vectors can be written as.

$$\mathbf{s}^e_{\text{ref},\kappa} = \begin{bmatrix} 0 \\ y^e_{\text{ref},\kappa} \\ v^e_{x,\text{ref}} \\ 0 \\ 0 \end{bmatrix}, \mathbf{u}^e_{\text{ref}} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{2.20}$$

where $y^e_{\text{ref},\kappa}$ is one of the three highway lane centers, depending on which controller, $\kappa = \{\text{trail}, \text{left change}, \text{right change}\}$ the cost is calculated for and $v^e_{x,\text{ref}}$ is defined as longitudinal reference speed.

General constraints for the optimization of each controller can be formulated by considering the constraints based on which controller is being optimized for and the surrounding vehicles' future predicted positions (2.14) and (2.16) together with the physical limitations on the ego vehicle (2.5)

$$h_\kappa^e(\mathbf{s}_\kappa^e, \mathbf{u}_\kappa^e, \hat{x}^{(i)}, \hat{y}^{(i)}) \leq 0, \ \forall i \in \mathbb{V}_\kappa \tag{2.21}$$

where h_κ^e is the general constraint function and $\mathbb{V}_\kappa$ is the set of vehicles i considered by the controller κ, i.e only leading vehicle when $\kappa =$ trail and all vehicles otherwise. With the cost function defined in (2.17) the MPC optimization can be formulated

$$\begin{aligned}
\min_{\mathbf{u}_\kappa^e(t:t+T_{\text{pred}}-1|t)} \quad & J_\kappa\left(\mathbf{s}_{\text{init}}^e, \ \mathbf{u}_\kappa^e(t:t+T_{\text{pred}}-1|t)\right) \\
\text{s.t. } & \mathbf{s}_\kappa^e(t+n+1|t) = g(\mathbf{s}_\kappa^e(t+n|t), \mathbf{u}_\kappa^e(t+n|t)) \\
& (\hat{x}^{(i)}(t+n|t), \hat{y}^{(i)}(t+n|t)) = \Pi_{\text{CV}}(x^{(i)}(t|t), y^{(i)}(t|t)), \ \forall i \in \mathbb{V}_\kappa \\
& h_\kappa^e(\mathbf{s}_\kappa^e(t+n|t), \mathbf{u}_\kappa^e(t+n|t), \hat{x}^{(i)}(t+n|t), \hat{y}^{(i)}(t+n|t)) \leq 0 \\
& \mathbf{s}_\kappa^e(t|t) = \mathbf{s}_{\text{init}}^e \\
& \forall \ n \in \ \{0, \ldots, T_{\text{pred}}-1\}
\end{aligned} \tag{2.22}$$

where Π_{CV} is the constant velocity trajectory predictor for the surrounding vehicles positions through out the prediction horizon and $g(\cdot)$ is the plant model for the ego vehicle. Furthermore, the terminal state at timestep T_{pred} also needs to conform to the inequality constraints stated in the optimization. The optimization formulation is solved using the Interior Point Optimizer (IPOPT), implemented in the Casadi framework [25]. To be able to choose the optimal cost-minimizing control input a DM is implemented, as developed in [22]. The control input for the different controllers is calculated as in (2.22). Thereafter the DM chooses the controller that minimizes the following equation

$$J_{\kappa,tot} = a_1 J_\kappa + a_2 J_{\text{change}} \tag{2.23}$$

where J_{change} is the cost of changing between the active controller. $a_{1->2}$ are scaling parameter that weighs the importance of each cost. The decision master is more thoroughly explained in [22], for the interested reader.

One part that becomes apparent while observing the optimization formulation in (2.22), is the need for an observer to predict surrounding vehicles' trajectories to generate the constraints involved in the optimization. Important to highlight, is that the trajectories of the surrounding vehicles in the optimization are predictions. Since these are predictions there will be an included uncertainty present in the trajectory planner. Another identified problem also extends to the notion of how can these prediction uncertainties be involved in the optimization to formulate a more robust controller.

2.4 Problem statement for predicting surrounding vehicles' future trajectories

A trajectory can be defined as the curve that an object travels through in space. Extending this further to the notion of prediction the problem definition changes to estimating how an will object travel through space. One way to solve this problem is to look at the object's previous states and use a model to predict the future position

$$(\hat{x}(t+1),\ \hat{y}(t+1)) = \Pi_{\psi}(\mathbf{s}(t)) \tag{2.24}$$

where Π_{ψ} represents a predictor function based on parameters ψ that uses previous state information $\mathbf{s}(t) \in R^D$ where D can represent any number of features (position, velocity, heading, etc.) to predict one time steps position into the future, namely $(\hat{x}(t+1),\ \hat{y}(t+1))$. The feature inclusion for $\mathbf{s}$ can be designed for the required problem to solve, e.g. input previous velocity and position and then output position or position and velocity. For brevity in this chapter the input to the predictor function Π_{ψ} will be selected as x- and y-positions with identical states for output. Continuing on the I/O relations the problem can further be extended towards a time series prediction problem where the function should not be limited by only predicting one time step into the future.

$$(\hat{\mathbf{x}},\ \hat{\mathbf{y}}) = \Pi_{\psi}(\mathbf{x},\ \mathbf{y}) \tag{2.25}$$

where $(\mathbf{x},\ \mathbf{y})$ represents previous state information from historical time T_{obs} up to the current time step and $(\hat{\mathbf{x}},\ \hat{\mathbf{y}})$ represent the predicted trajectory up to time step T_{pred}.

Continuing on the notion of trajectory prediction, not only the object's current state in terms of x- and y-coordinates are interesting to involve. An important feature that can also be involved, especially for trajectory prediction in a vehicle environment, is the notion of scene context. An example of this from a traffic scenario is the information from traffic signs, lanes, sidewalks, etc. The model for the trajectory prediction should therefore consider, or condition on, prior and current state information together with the scene context information to perform the prediction. Therefore (2.25) can be updated with the inclusion of the scene,

$$(\hat{\mathbf{x}},\ \hat{\mathbf{y}}) = \Pi_{\psi}(\mathbf{x},\ \mathbf{y},\ \mathbf{l}) \tag{2.26}$$

where $\mathbf{l}$ represent previous scenes context from T_{obs} to current time step.

In the trajectory prediction problem two perspectives can be identified, scene and agent perspective. From an agent perspective, the problem can be defined as predicting the trajectory for one agent, a car for example, based on the surrounding scene context information. This then becomes an iterative approach where each agent's trajectory is computed separately to produce a prediction for all vehicles. An agent perspective model can therefore be formulated as

$$(\hat{\mathbf{x}}^{(i)},\ \hat{\mathbf{y}}^{(i)}) = \Pi_{\psi}(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \mathbf{l}^{(i)}),\ i = 1, 2, ..., N \tag{2.27}$$

for N agents. Furthermore, the scene perspective can be seen as a model that includes all agents' information and produces a trajectory prediction for the whole scene at once, thus computing trajectories for all cars directly

$$\mathbb{P}_{\text{pred}} = \Pi_{\psi}(\mathbb{P}_{\text{obs}}, \mathbb{L}_{\text{obs}}) \tag{2.28}$$

where the set $\mathbb{P}_{\text{obs}} = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{N}$ represent the set containing all previous agents positional information and its observed scene information $\mathbb{L}_{\text{obs}} = \{\mathbf{l}^{(i)}\}_{i=1}^{N}$. $\mathbb{P}_{\text{pred}}$ is the set containing all predicted positions, i.e $\{(\hat{\mathbf{x}}^{(i)},\ \hat{\mathbf{y}}^{(i)})\}_{i=1}^{N}$, for all the surrounding vehicles. Depending on the conditions of the problem formulation either approach can be beneficial since they will produce the same output in the end.

The objective of the trajectory prediction problem is to minimize the prediction error with respect to some model parameters ψ

$$\min_{\psi} |\Pi_{\psi}(\cdot) - (\mathbf{x}_{\text{GT}}, \mathbf{y}_{\text{GT}})| \tag{2.29}$$

where $(\mathbf{x}_{\text{GT}}, \mathbf{y}_{\text{GT}})$ represents the ground truth, GT, for future positions up to time step T_{pred}. Two common error metrics are often used in the evaluation of the performance: average displacement error (ADE) and final displacement error (FDE) for x and y prediction errors respectively. The ADE focuses on a holistic perspective over the whole prediction horizon where it computes the average prediction error

$$ADE = \frac{\sum_{i=1}^{N}\sum_{t=0}^{T_{\text{pred}}} \sqrt{((\hat{x}^{(i)}(t),\ \hat{y}^{(i)}(t)) - (x_{\text{GT}}^{(i)}(t),\ y_{\text{GT}}^{(i)}(t)))^2}}{NT_{\text{pred}}} \tag{2.30}$$

thus giving an average prediction error for all agents and all time steps into the future. The FDE metric is defined by

$$FDE = \frac{\sum_{i=1}^{N} \sqrt{((\hat{x}^{(i)}(T_{\text{pred}}),\ \hat{y}^{(i)}(T_{\text{pred}})) - (x_{\text{GT}}^{(i)}(T_{\text{pred}}),\ y_{\text{GT}}^{(i)}(T_{\text{pred}})))^2}}{N} \tag{2.31}$$

which only focuses on the last predicted state in the computation of the error metric.

3

Background theory on robust model predictive control and trajectory prediction

The theoretical background chapter aims at presenting the necessary and applied background to the theory used in the thesis. The chapter is divided into a trajectory planning part consisting of robust MPC, followed by previous work in the field. The section continues with a focus into trajectory predictions, with an overview of transformers. The section concludes with previous work in the area of trajectory prediction and techniques for modeling uncertianty in predictions.

3.1 Robust model predictive control

For real world systems or systems that exhibit uncertainties and inaccuracies, nominal models can only give an approximation of the dynamics of the system. As explained in earlier chapters, the receding horizon technique of the MPC provides some robustness to these inaccuracies through its iterative approach. Although, it is not possible to beforehand state if this controller is stable or not given uncertainties in the system [26]. Robust MPC (RMPC) aims to maintain a stable control policy for all feasible realizations of states in the model, subjected to bounded uncertainties. Two common ways to approach RMPC, are min-max MPC [27] and tube-based MPC [24]. Min-max MPC aims to minimize a defined cost function based on the worst outcome realization of states. Although, since the approach cannot overlook any realization in the optimization, meaning that many constraints will be formed, it is computationally heavy. Since tube-based MPC is less intensively computationally heavy a similar approach is applicable in this thesis.

Tube-based MPC is a way to include model uncertainty in the problem specification. In other words, the objective is to make a robust control policy for a system that exhibits bounded uncertainties in its modeling.

$$\mathbf{s}^e(t+1) = g(\mathbf{s}^e(t), \mathbf{u}^e(t), \mathbf{w}(t)) \tag{3.1}$$

where $\mathbf{w}(t)$ is the propagated bounded uncertainty at time step t. Since there is uncertainty present in the dynamics of the system, a multitude of states will be viable

at every sample time. Robust tube-based MPC aims to keep all viable states inside the defined constraints in the optimization and is done by forming a region/tube around the reference trajectory based on the uncertainty of the nominal system. A representation of the approach can be seen in Figure 3.1.

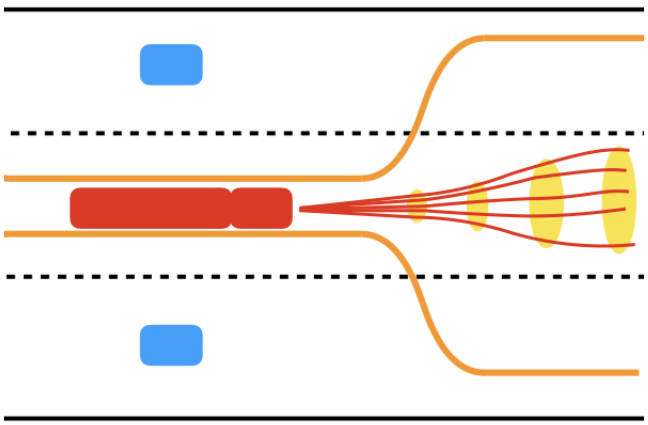


Figure 3.1: Tube-constrained MPC scenario where the red rectangle is the ego vehicle and the blue rectangles are surrounding traffic with corresponding constraints in orange. Multiple state realizations are generated due to uncertainty in the model of the ego vehicle and are depicted as red lines. The yellow 2D surface corresponds to all possible state realizations and must conform to the surrounding constraints.

There are multiple viable options of trajectories coming from the ego vehicle in Figure 3.1 which are dependent on the nature of the uncertainty the mathematical model is subject to. A 2D region is formed that surrounds all possible trajectories from the ego vehicle. In this example, the yellow surfaces, which represent all possible trajectories, must conform with the collision avoidance constraints, depicted in orange lines surrounding the other vehicles.

3.1.1 Previous work in trajectory planning using robust and stochastic model predictive control

In the field of robust and stochastic model predictive control (RMPC, SMPC), there are some publications that show significant benefits of using uncertainties and a probabilistic setting in the problem formulation. In [28], a grid-based SMPC is used for the trajectory planning. The surrounding environment is divided into grids with assigned probabilistic values of a vehicle occupying it. The trajectory planner can thus include the stochastic nature of the surrounding vehicles allowing for increased robustness in the MPC. Further, [29] presents an SMPC in combination with a fail-safe trajectory planner. This enables an efficiently planned trajectory in a non-deterministic environment whilst ensuring that safety constraints are met. The SMPC is formed by using chance constraints, letting the optimal control problem account for uncertainties by including a modeling risk factor. The fail-safe trajectory planning is then used as a backup, considering the worst-case uncertainty

comprehension. Accordingly, the SMPC is used to plan efficient paths whilst the pessimistic fail-safe trajectory planner ensures that the vehicle is kept in a feasible state.

The research field of combining learning-based prediction approaches with model predictive control for optimal trajectory planning is not widely explored. However, there is one paper that highlights the interesting benefits of the proposed method. In [3], lane-change maneuver control based on model predictive control and neural networks is proposed. More specifically, a framework that coordinates optimal control inputs from a MPC with predicted driver intentions generated from a generative adversarial network (GAN) is presented. The method applies adaptive safety boundaries to improve performance in practice. Notably, for some metrics such as maximum acceleration the simulation results show that the suggested method does not outperform a Game tree method. However, in terms of smoothness in trajectory planning, robustness to traffic density, and number of infeasible simulations, the proposed method shows prominent results.

3.2 Transformer

Transformers (TF), similar to Recurrent neural networks (RNN), are outlined to facilitate sequential data with tasks including natural language processing (NLP). RNNs use sequential processing to incorporate an expression of the state up to the current data point with the information of all previous data points to form a new state expression. However, the vanishing gradient problem [30] diminishes the usefulness of the approach, as vanishing gradients limit the preciseness of the information of the prior data points. Further, as the data points are handled sequentially, training can be seen to be ineffective for long sequences. TFs have the capability to handle all the data simultaneously which allows for faster training times due to the possibility for parallelization [10]. This is made possible by utilizing a mathematical technique called self-attention which allows for obtaining knowledge from any state in the prior sequence. The base architecture of a transformer model can be seen in Figure 3.2.

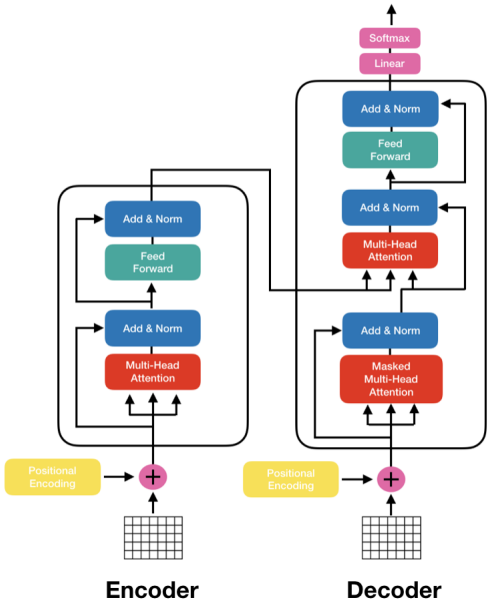


Figure 3.2: Transformer architecture based on encoder-decoder structure. The encoder, seen in the left part of the figure, transforms the input into a latent representation and identifies important aspects of the data. The decoder, seen in the right part of the figure, receives as input the latent variable representation from the encoder and its own input to produce its output.

Encoders and decoders are used to process the data in a transformer network. The encoder part of the TF transforms the input into a latent representation and identifies important aspects of the data which consequently is passed to the decoder. The decoder of the TF combines the output from the encoder and its own input to produce an output. The number of encoders and decoders layers in the TF is a design parameter that can be changed depending on the requirements of the problem at hand. While the first model was created for NLP, it has been found an interesting tool to handle time series problems such as trajectory predictions. This is mainly due to its possibility to capture long-term dependencies in a time series [31].

3.2.1 The self-attention mechanism

Self-attention mechanism serves as a remedy to the problem of vanishing gradients as it has the capability to identify relationships between all data points in a sequence, not only in a sequential manner. The idea of attention is to focus on the parts of the input that are relevant to the objective in mind. Much like the human's analytical behavior to attend only to important parts of a message, attention mechanisms let models disregard insignificant data points. The result is an effective model that can focus its processing abilities on relevant data. An attention function can be identified using three components, Queries($\mathbf{Q}$), Keys($\mathbf{K}$), and Values ($\mathbf{V}$) which are all matrices [10]. Each vector in the value matrix, $\mathbf{V}$, is given a specific weight that captures its importance. The weight is computed by using a compatibility function for the query with the analogous key. The output is obtained by considering the

weighted sum of the values. The Keys, Queries, and Values can be written in a vector notation seen below,

$$\mathbf{k}_j = \mathbf{I}_k \mathbf{r}_j \tag{3.2a}$$

$$\mathbf{q}_j = \mathbf{I}_q \mathbf{r}_j \tag{3.2b}$$

$$\mathbf{v}_j = \mathbf{I}_v \mathbf{r}_j \tag{3.2c}$$

where $\mathbf{k}_j$, $\mathbf{q}_j$ and $\mathbf{v}_j$ are the jth vector in the key, query and value matrices respectively. $\mathbf{r}_j$ corresponds to one input vector, here vector j, in the set of input vectors and $\mathbf{I}_k, \mathbf{I}_q$ and $\mathbf{I}_v$ are matrices of trainable parameters. For every Query and Key pair, a corresponding normalized C value representing their compatibility is calculated

$$C_{i,j} = \frac{\mathbf{k}_i^T \mathbf{q}_j}{\sqrt{d_{\text{model}}}} \tag{3.3}$$

where $i, j \in 1, \ldots, A$ and A is the number of vectors in the input sequence. $C_{i,j}$ is the compatibility for key vector i and query vector j and d_{model} is the number of features in one input vector.

To obtain the weights, W, that are positive and sum up to one for the corresponding values, the compatibility value is propagated through a SoftMax function.

$$[W_{1,j}, \ldots, W_{A,j}] = \text{softmax}(C_{1,j}, \ldots, C_{A,j}) \tag{3.4}$$

The new embedding or as called in [10], "the attention function" is obtained by summing all values against the corresponding weights.

$$[\mathbf{e}_{1,j}, \ldots, \mathbf{e}_{A,j}] = [\sum_{j=0}^{A} \mathbf{v}_j W_{1,j}, \ldots, \sum_{j=0}^{A} \mathbf{v}_j W_{A,j}] \tag{3.5}$$

where $[\mathbf{e}_{1,j}, \ldots, \mathbf{e}_{A,j}]$ is the new embedding representing the attention for input vectors 1 to A connected to input vector j.

To get a more dense representation, (3.3) can be written on matrix notation.

$$\mathbf{C} = \frac{\mathbf{K}^T \mathbf{Q}}{\sqrt{d_{\text{model}}}} \tag{3.6a}$$

$$\mathbf{K} = [\mathbf{k}_1, \ldots, \mathbf{k}_A] \tag{3.6b}$$

$$\mathbf{Q} = [\mathbf{q}_1, \ldots, \mathbf{q}_A] \tag{3.6c}$$

where $\mathbf{C}$ is the compatibility for all input vectors. Following the matrix notation, (3.4) and (3.5) can be written in a more condensed form

´

$$\mathbf{W} = \text{softmax}(\mathbf{C}) \tag{3.7a}$$

$$\mathbf{E} = \mathbf{VW} \tag{3.7b}$$

where

$$\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_A] \tag{3.8}$$

and $\mathbf{E}$ is the new embedding for all input vectors. This particular method of calculating the attention function is called Scaled dot-product attention. A figurative representation of the attention mechanism can be observed in Figure 3.3.

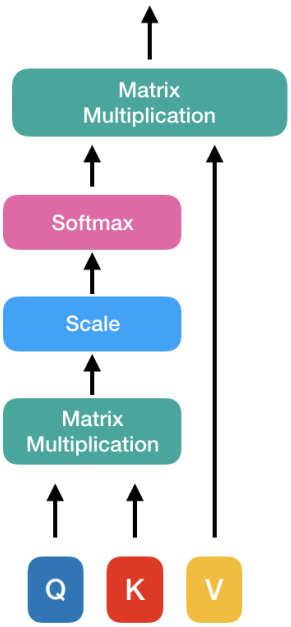


Figure 3.3: Single head attention mechanism and how the Queries (Q), Keys (K), and Values (V) get processed.

3.2.2 Multi head-attention

Alternatively to using a single attention mechanism shown in (3.7), one can project the keys, values and queries to perform multi-head attention [10]. By doing this, several different aspects of the input data can be captured in each separate head. The Multi-headed attention mechanism can be seen in Figure 3.4.

Each attention head, h, is computed as seen in Figure 3.3, which is a linear projection of the input as queries, keys and values. Based on the $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$ the attention function is performed τ times capturing unique aspects of the input data. Thereafter, the outputs are concatenated and linearly projected to a predefined size. As the $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$´s are randomly initialized, the multi-head attention mechanism lets the model catch information from different sub-spaces, allowing it to learn connecting information from the different spaces. The equations below describe the multi-headed attention mechanism

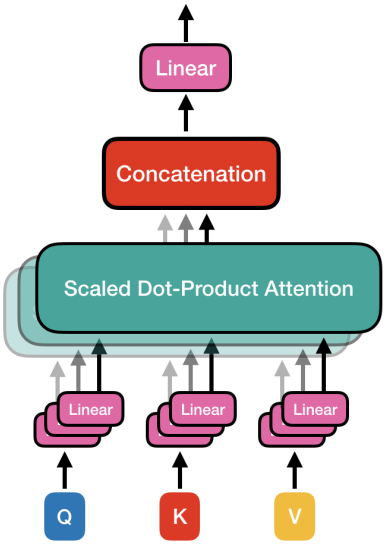


Figure 3.4: Multihead-attention module that is built upon multiple singe head attention mechanisms, where their outputs are concatenated, with the purpose to catch information form different sub-spaces.

$$M = \text{concat}(h_1, \ldots, h_\tau)\mathbf{I}^O \tag{3.9a}$$

$$h_k = \text{head}_k = \text{softmax}\left(\frac{(\mathbf{Q}\mathbf{I}_k^Q)(\mathbf{K}\mathbf{I}_k^K)^T}{\sqrt{d_{\text{model}}}}\right)\mathbf{V}\mathbf{I}_k^V, \quad k = 1, \ldots, \tau. \tag{3.9b}$$

In the above equation $\mathbf{I}^O, \mathbf{I}^Q, \mathbf{I}^K$ and $\mathbf{I}^V$ correspond to linear projection matrices for the queries, keys and values and are learnable parameters. Together with the aforementioned mechanism, an optimization strategy for the learnable parameters and the structure in figure 3.2, a multi-sided DL model capable of numerous tasks can be created. For the interested reader, a more detailed explanation of the transformer structure is outlined in [10].

3.2.3 Positional encoding

Using transformers allows for whole sequences to be processed simultaneously. Although, as no recurrence is present compared to RNN models, the transformer cannot monitor the position of individual features in the sequence by itself. Thus, there is a need to supply the model with instructions about the relative position of each data point in the sequence. This is not a novel problem identification that sprung from this project and is clearly recognized in the first transformer paper [10]. Positional encoding implementation can take many forms but a distinction can be made between learned and fixed positional encoding. The learned positional encoding is an encoding that is modified during training. A fixed encoding is in contrast not altered during training. Similar performance of the different encoding procedures has been shown in [10]. A common and frequently used positional encoding is the

fixed sine/cosine positional encoding procedure

$$PE_{\text{pos},2i} = \sin(\frac{\text{pos}}{\beta^{2b/d_{model}}}) \tag{3.10a}$$

$$PE_{\text{pos},2\mu+1} = \cos(\frac{\text{pos}}{\beta^{(2\mu+1)/d_{\text{model}}}}) \tag{3.10b}$$

where pos represents the index of which element in the input sequence, β a design parameter often set to the maximum length of the sequence input in order to get a reasonable resolution for the encoding and μ an index reference towards which value in the d_{model} dimensional embedded input sequence. $PE_{\text{pos},2b}$ acts on even μ and $PE_{\text{pos},2\mu+1}$ on odd μ. Together with the varying parameters pos it can generate unique values for each of the input features in the sequence. The encoding in (3.10) then adds to the input sequence to give a fixed bias that the model can utilize and learn its dependency.

3.3 Previous work in trajectory prediction

The literature review focuses on different methods for trajectory predictions. Firstly, common prediction methods using sequential processing is investigated in section 3.3.1. This is followed by an investigation into the transformer approach for predictions in section 3.3.2.

3.3.1 Motion prediction through sequential processing

Before the big era of utilizing DL, statistical models were mainly used for trajectory prediction [32]. Among these statistical models is the Kalman filter which simply is explained as a model to predict unobservable state trajectories by utilizing state space modeling. The neural network-based deep machine learning models have in the recent decade extinguished themselves as state-of-the-art models for trajectory predictions in complex vehicle environments. The use of sequential DL models such as RNN has shown good results in trajectory predictions due to its capabilities of capturing and modeling sequential data [33, 34]. RNN models process data sequentially where the information from each time step is propagated through the network [35]. The information is then stored from previous propagations through the model via a hidden state for future input. A weakness of the RNN is that it has problems modeling long-term dependencies during processes that have long sequences. The problem expresses itself mainly through the vanishing/exploding gradient problem first introduced in [30]. The problem arises during the gradient descent optimization where computing of gradients may have exponential decrease/increase during backpropagation. This decrease/increase may result in the updating of the parameters being either too large or too small leading to a model that doesn't learn. One approach to addressing this issue is through a modification of the RNN by implementing a long-short-term memory (LSTM) module. The motivation for utilizing an LSTM model improves the capabilities of modeling long-term dependencies.

LSTM models have been used in multiple application domains [36, 37] and showed great abilities in predicting sequential dependencies. One aspect of the sequential pattern that some of these models did not consider was the notion of social interactions. Object's single trajectories can be modeled but it is important to also weigh in the reasoning of how these objects are affected by each other in a more complex environment with multiple interactions. It can be identified that there is a social aspect also to be considered in these environments which in [38] is considered by the authors Social-LSTM. The Social-LSTM incorporates a social pooling mechanism that fuses information from single predicted trajectories of objects into a pooling mechanism for spatially close LSTM trajectories for predictions. The model was considered a big stepping stone for the incorporation of sequential and social modeling for trajectory forecasting. Refinements of solving the trajectory prediction problem have been conducted by modifications in the inclusion of the social aspects. One example of this is shown in [39] where the Social-BiGAT model incorporates the social modeling through a graph attention network combined with recurrent generative adversarial training. Further models based on LSTM to model temporal dependencies have emerged where the social context modeling comprises of different strategies [40, 34].

3.3.2 Transformer based approaches

With the creation of the transformer model in [10] a new approach towards solving sequential tasks emerged. By the use of multiple encoder-decoder structures, the multi-head transformer model was able to obtain superior results on NLP tasks. Clear strengths identified with the transformer are the possibility of modeling long-term dependencies and predicting trajectories through a non-sequential fashion which enables greater possibilities to fully capitalize on the strength of parallel computing and graphics processing unit (GPU) training. The field of transformers is somewhat limited in its exploration of trajectory prediction applications due to its tender age. However, there are some interesting results in terms of multi-trajectory predictions in the application area of both humans and vehicles. In [41] the problem of multi-trajectory prediction is separated into two sub-problems: sequence modeling and social+context modeling. Previous approaches have tried to tackle these problems by a two-model integration approach where one model is used for the social modeling, for example, a GNN structure, and one is used for the sequential data, RNN-based model. Instead of this traditional approach, the authors present the approach of simply predicting individual trajectories with a vanilla transformer motivated by its capabilities of capturing non-linear sequential data through its attention mechanism. In conjunction with applying a vanilla transformer they also investigated the Bidirectional Encoder Representations from Transformers (BERT) [42]. The basic transformer displayed promising results on the challenging TrajNet [43] compared to models without a transformer base [39, 4, 44].

Interesting to note about the transformer is its capability of modeling spatial and social interactions without constituting a mechanism specifically designed for the task such as graph-based methods. In [45] the authors present the hypothesis that the lack of spatial modeling components in the transformer structure is actually its

strength. This strength is identified via the perspective that the transformer becomes a relatively simple construction and therefore facilitates better interpretability and explainability. Whether or not this statement is true in reality is up for debate. Nevertheless, the overall size and complexity of modules to be involved in an "only transformer" based structure can be argued to be simpler than mixing different networks such as CNN, GNN and RNN into one complex model. In [46] the combination of using transformer modules for both vision and trajectory encoding are utilized in a regressive and future-aware transformer model. Motivated by this fact the transformers appear to be a structure capable of capturing both temporal and spatial relations.

Even though the transformer and its self-attention mechanism show promising results in application towards individual trajectory predictions the problem still remains at how to fully exploit and involve the social context and spatial relations. In [47] claims are stated that the two modular approaches of separate network structures for social and temporal relations is suboptimal. This is motivated by the fact that there could be a loss of information by this separation and a model to involve both would be beneficial. The authors therefore use their transformer based AgentFormer to incorporate both problems. The base constitutes from the original transformer [10] with multiple conditional variational autoencoders (CVAE) modules. The main novelty of the model is their agent-aware attention to remove index dependency that can occur during input embedding to enable a permutation invariant model for the agents. The AgentFormer is assessed on a data set constituted of both pedestrian and autonomous driving data with substantial improvements compared to models based on separate social and temporal modeling.

Methods for multi-trajectory predictions can also be viewed from the perspective of two method approaches: feature-based and proposal-based approaches [48]. The two approaches distinguish themselves where feature approaches try to model environmental impacts such as past position and velocities to predict future trajectories. However, this approach is claimed to lack strength in modeling multiple predictions for each time frame i.e. a shortfall of multi-modal predictions. The proposal-based approach instead focuses on generating multiple candidate trajectories which are then reduced towards achieving final proposals. Identified weaknesses in this approach however is the loss of latent features that can be overlooked and discarded during the proposal generation. To tackle this problem the multi-modal transformer presented in [48] tries to combine both feature and proposal approaches to reduce the shortcomings of both approaches. Claims towards the strength of their model are that they are able to capture the multi-modality of the problem through a region-based training strategy while still involving latent features in the proposals. The mmTransformer display promising results on the BEV dataset Argoverse [49].

Clear motivations together with results for transformer based networks are presented in the reviewed literature. However, it is important to emphasize the question of whether some claims in the articles about explainable and motivated implementation actually coincide with the truth. This involves some arguments and statements where certain implementations and extensions of the transformer network "actually do what they say they do?". Interpretability is an area of weakness of machine learn-

ing models and is especially applied to complex problems [50]. This characteristic is still a property of the transformer model which makes it somewhat questionable for applications in safety-critical environments. Following this it is still important to look at the actual results which speak and motivates an investigation towards the possibilities of applying transformer based networks for trajectory prediction.

3.4 Uncertainty modeling in Machine Learning

What is important in the development of machine learning algorithms is the notion of uncertainty in prediction, i.e. how certain is the model. Attacking uncertainty modeling from a probabilistic and statistical perspective is one way to approach uncertainty modeling. However, it is important to note that this modeling technique fails at distinguishing and capturing what the uncertainty is based on [51]. Section 3.4.1 introduces different types of uncertainties followed by techniques of uncertainty modeling for deep machine learning in section 3.4.2.

3.4.1 Cause of uncertainty

Uncertainty in deep machine learning development can be separated into two categories: aleatoric and epistemic uncertainty. For aleatoric, the uncertainty is conditioned on uncertainty from the data that the model receives. Aleatoric uncertainty becomes apparent when real-world data is used in for example sensor measurements where there is randomness in the data. Aleatoric uncertainty can be referred to as stochastic uncertainty where for example a sensor measurement of the position of a vehicle can differ while the actual ground truth position is the same for different measurements [51].

In connection to the aleatoric uncertainty, epistemic uncertainty focuses on the model itself, i.e. its parameters, and how uncertain it is in its predictions. In other words, it is an uncertainty measure expressed by how certain the model is for the data. Characterizing which of these uncertainties is analyzed can be done through the question if they can be reduced. For example in the aleatoric case which refers to stochastic uncertainty the performance cannot be reduced due to the deterministic nature of the I/O relation [51]. However, for the epistemic uncertainty, the problem is more defined towards how certain the model is compared to a perfect prediction. This epistemic uncertainty can often be reduced by means of an increase in the amount of training data. It is important to note that this separation of definition is not included in terms of its causality. An example of this presented in [51] is that a reduction of aleatoric uncertainty can be achieved by the projection of the input data to a higher dimensional space which actually may increase the epistemic uncertainty. Clearly, the two uncertainty definitions are interlinked.

In terms of representing epistemic uncertainty, one goal during inference of the model is to produce how uncertain it is about its predictions. The model will always produce a result given an input, but it can also be beneficial to produce a result on how certain the prediction by the model is. An epistemic uncertainty measure can be regarded as a movement for increasing a DL model's interpretability by expressing

its certainty during inference. This movement towards uncertainty measures is a highly south-after characteristic for machine learning models, especially for real-time usage in safety-critical situations [52]. Three fields for uncertainty expression are further analyzed in section 3.4.2: Bayesian Neural Networks [53], Deep Ensemble Learning [54] and Monte Carlo Dropout [55].

3.4.2 Techniques for uncertainty modeling

What the three different strategies try to achieve during inference time is a way to express a notion of certainty on its prediction. The Bayesian Neural network (BNN) model can be summarized by an extension from deterministic values of the parameters, and weights, of the model to a non-deterministic representation [53]. This can be achieved by regarding the parameters as probability distributions that are sampled during training. The BNN uses a probabilistic representation for the parameters which can be utilized for uncertainty measures in the model's predictive capabilities. Instead of utilizing and extending the model through non-deterministic parameters Deep Ensemble learning uses several deterministic models to produce several point estimates instead of a probability distribution [54]. Important to note here is that the ensemble of models are the same instances but with differently initialized parameters before training and trained independently. What this results in are that each separate model will be optimized slightly differently than the others due to the stochastic nature given by the training algorithm, thus producing different predictions.

The Monte Carlo Dropout method can be seen as an approximation of the Bayesian strategy. The dropout functionality has mostly been regarded as a dynamic and adaptive regularization technique for preventing over-fitting during training [56]. However, in [55] Monte Carlo Dropout is introduced as a measure to reduce the computational increase of implementing BNNs with promising results. Applying a Monte Carlo method is relatively straightforward since it involves simply applying a dropout functionality on the model parameters and performing multiple predictions for each input. These multiple predictions are then averaged to a selected representation to result in an estimate of the model's uncertainty.

4

Predicting surrounding vehicles' trajectories

Initially, the transformer was designed for NLP tasks with examples of text translation, text summarization, and sentence generation. Clearly, the problem that is presented in this thesis does not directly apply as an NLP problem. Based on the information about the foundation regarding the transformer structure an appropriate I/O structure had to be selected and generated. The problem of trajectory prediction can be seen as a multivariate time series problem where a model capable of regression can directly be applied. This is one approach that has shown some promising results for pedestrian predictions [41]. However, since the transformer structure is initially developed for NLP and classification tasks the decision was made to utilize this fact, i.e. adapt the data for a classification problem. Variations of a trajectory prediction model using a transformer base have been using some sort of classification approach. This includes applications of a K-means algorithm on the velocity states for quantization [41], using a trajectory proposal for multi-modality classification [48] or classification over joint modal prediction combinations [57]. Based on the foundation of the transformer model for classification tasks together with promising results in this domain the data for the predictions therefore needed to be quantified from its continuous representation towards application in a classification problem.

Successful models from literature have mostly been focused on the incorporation of a multitude of focus areas in their trajectory prediction tasks. Examples of focus areas are modeling of agent-relational information, scene incorporation, the likeliness of plausible future, temporal feature history, etc. Often there is one single or several encoding module/modules in the models which represent these areas of interest in a latent variable representation. These latent variables are then combined for decoding plausible future trajectories. By combining the classification formulation and involvement of multiple encoding and decoding modules a novel transformer-based architecture was created named Trajectory Observer Transformer (TOT) which can produce predictions for multiple surrounding vehicles. The TOT was developed with the open source deep machine learning framework PyTorch [58].

The chapter initiates with an overview of the TOT model and brief introduction to its modular components. Section 4.2 presents the data generation setup which was used to acquire the necessary data to train the supervised learning model. In sections 4.3 the details of the input embedding and process of the encoders are presented and similar for the decoders in section 4.4. The chapter continues in the description of the

steps used during implementation together with the selected method for expressing uncertainty during inference. The chapter finishes with section 4.6, which presents the outline for the result generation that the evaluation of the observer was based on.

4.1 Trajectory observer transformer structure

Figure 4.1 is an illustration of the full encoder/decoder structure of the TOT model presented during implementation and inference. The green dashed lines represent the previous embedded positions of surrounding vehicles, the red dashed dots represents the last embedded velocity of the surrounding vehicles and the orange dashed lines are the predicted future positions of each surrounding vehicle. In Figure 4.1 the strategy for producing predictions is also displayed by separating the encoding/decoding into N iterations, i.e one for each agent while still considering the whole scene and repeating T_{pred} times for rolling out future positions.

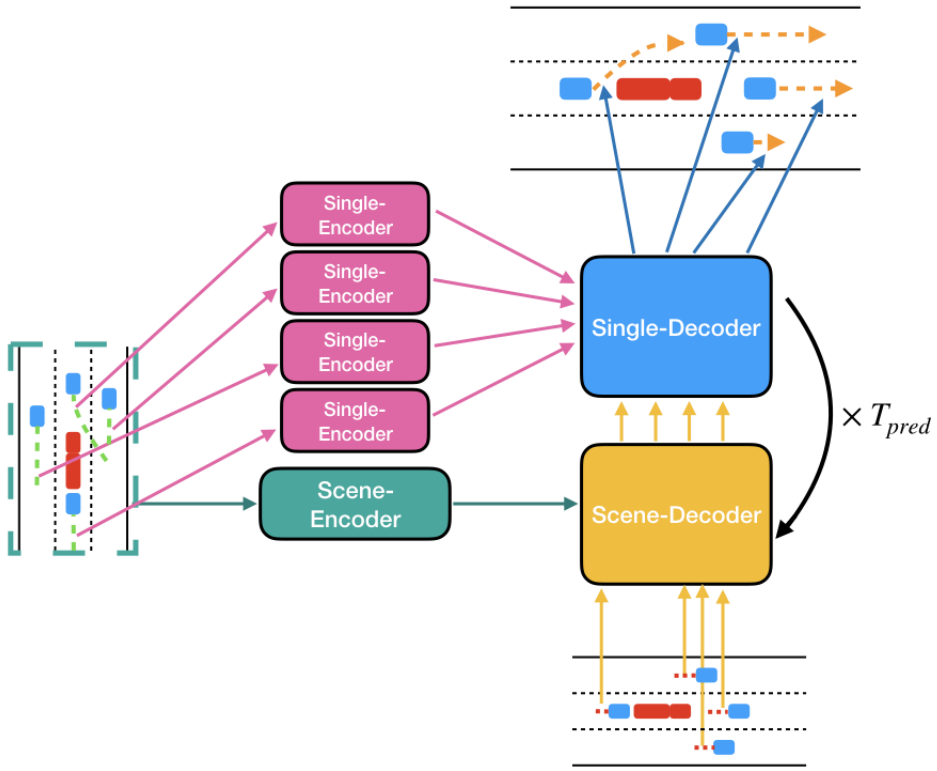


Figure 4.1: The model encodes observed embedded positions in both the scene-encoder for representation of the current scene and in the respective single-encoder for the surrounding vehicles. The decoding constitutes of a combination of the last observed velocity together with the encoded latent variable representation from the encoders. The ego vehicle is represented by the red rectangles and surrounding traffic by blue rectangles.

Details for the individual modules for the different encoders and decoders are presented in Table 4.1. It can be viewed that the same number of attention heads, number of repeated layers of each module and feed forward dimension are the same for each of the separate modules. The selected size of d_{model} which can be seen as a baseline size for the TOT is 256.

Table 4.1: The number of heads, layers and dimensions of feed forward network defined for the separate encoder and decoder modules.

Module	**# Heads**	**# Layers**	**Dim. feed forward**
scene-encoder	4	4	512
single-encoder	4	4	512
scene-decoder	4	4	512
single-decoder	4	4	512

4.2 Data generation

The surrounding traffic was modeled as in section 2.1 and initiated and controlled by the description in section 2.2. There are some parameters that needed to be defined to customize the generation of data. Firstly, the politeness factor of each surrounding vehicle, posed in equation (2.10), had to be defined. There are three settings that can be applied: passive, normal, or aggressive. These settings are highly correlated with the number of lane changes the surrounding vehicles will do during a simulation, where an aggressive driver will likely perform more lane changes than a passive driver. Further, the starting position and initial velocity of each vehicle needed to be defined. The number of surrounding vehicles was 5 and their initial states can be seen in Table 4.2.

Table 4.2: Simulation setup for surround vehicles, where politeness, relative velocity compared to the ego vehicle, longitudinal position and the initial lane is defined for each of the 5 surrounding vehicles.

Vehicle	**Politeness**	**Relative velocity**	**Longitudinal Position[m]**	**Lane**
1	Normal	90%	30	middle
2	Passive	80%	45	right
3	Normal	85%	-65	left
4	Aggressive	140%	130	middle
5	Aggressive	120%	40	left

Most of the drivers were stated as aggressive or normal. The reason for this was to generate more lane changes for the observer to train on. The relative velocity was based on the initial reference velocity of the ego vehicle which was 50 km/h. The

reference velocity after an initial settling period for the ego vehicle was 60 km/h. Motivated by generating more lane changes, aggressive drivers are set at a higher initial velocity than the reference velocity due to their inherent nature to change lanes more often.

The surrounding vehicles are often moving faster or slower than the ego vehicle which results in that the distance between them grows with time. Surrounding traffic that are far away from the ego vehicle are not of interest for the MPC solution. Therefore, vehicles that were further than +/-200 meters longitudinally were subjected to a respawn function that spawns the vehicles either behind or in front of the ego vehicle and in a random lane. The speed of the respawned vehicles is uniformly distributed around the reference velocity. If the speed was greater than the reference velocity, they were spawned behind, otherwise in front.The politeness was kept the same as initialized in the simulation.

The ego vehicle was modeled as described in subsection 2.1 and controlled by the MPC solution described in section 2.3. The prediction horizon was set to $T_{\text{pred}} = 30$ with a sampling interval of $t_s = 0.2$ seconds. The initial ego vehicle setup is presented in Table 4.3.

Table 4.3: Simultation setup for the ego vehicle, which is controlled by model predictive controller structure presented in 2.3.2.

Vehicle	**Control**	**Velocity [km/h]**	**Longitudinal Position [m]**	**Lane**
EGO	MPC	50	0	middle

When a simulation either reached the maximum allowed time, $t_f = 1000$ or the maximum allowed distance $d_{max} = 600$ m the surrounding traffic was reset in a random lane and longitudinal position around the ego vehicle. The vehicles were spawned with a uniformly distributed velocity around the reference velocity. The politeness factor was the same as initialized. The ego vehicle never re-spawned during a simulation. However, at the end of a simulation it was reinitialized with the same settings as in Table 4.3. The total number of simulations carried out was $N_s = 1000$ where states such as position and velocity of each vehicle were recorded every 0.2 seconds.

4.2.1 Data pre-processing

The observer is required to produce predictions of length T_{pred} for all N agents given each time step during inference. The selected input to the model is the previous historical positions of all surrounding vehicles given the current time step. This I/O relationship can be described by

$$\{(\hat{\mathbf{x}}^{(i)}, \hat{\mathbf{y}}^{(i)})\}_{i=1}^{N} = \Pi_{\psi}(\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{N}) \tag{4.1}$$

In Table 4.4 the selected sampling time, observed information for input to the model, and prediction length is presented. The prediction horizon of T_{pred} was selected

with the motivation that the implemented baseline MPC needs this horizon length to comfortably complete a full decision toward a lane change. During empirical investigations, shorter prediction horizons resulted in the MPC being strongly biased towards trailing decisions which resulted in a non-dynamical environment with no lane changes for the ego vehicle. The sampling time of t_s=0.2 seconds was the same used original data generation and kept for simplistic reasons and removing the need for any up- or down-sampling operations.

Table 4.4: T_{obs} is the number of discrete time steps from observed information to include into the model predictions. Furthermore T_{pred} is the prediction horizon length of the model output.

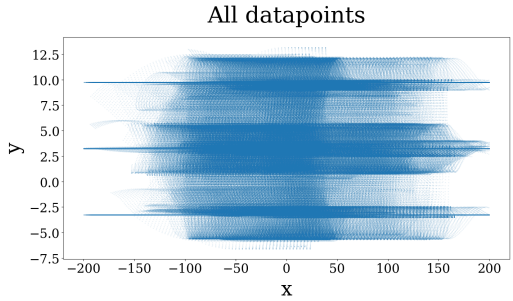
	# Samples	**Sampling time [s]**
T_{obs}	15	0.2
T_{pred}	30	0.2

Since the designed model is a supervised machine learning model, data needed to be processed in order to acquire accurate data for both historical and future positions. Pre-processing was mainly concerned with generating complete and meaningful scenes where each vehicle surrounding the ego vehicle needed to have a continuous historical and future trajectory. One example of scenes that were removed was if the future trajectory of some surrounding vehicle had a transition into a re-spawn as described in section 4.2 and therefore not had a full historical and future trajectory.

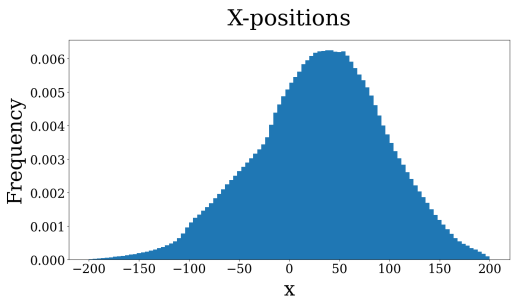
The second step of the pre-processing consisted of generating an ego vehicle centered coordinate frame for the x-positions of the surrounding vehicles. This step was necessary in order to remove any bias toward how far the vehicles had traveled on the highway. The field of view for the ego vehicles observer predictions for vehicles to include in the simulation was also reduced to an interval of [-200, 200] meters in x-position relative to the ego vehicle. This strategy was used to mimic a scenario where the ego vehicle can not view the whole highway at once. However, in order to keep representation for the lanes, y-positions were not related to an ego vehicle centered coordinate system since these are static values due to the straight driving direction of the highway.

Following the requirements for continuous scenes the pre-processing resulted in 52529 scenes with 5 surrounding vehicles' historical and future states. All data points of all sequences in the dataset for the model can be seen in Figure 4.2a. To further gain insights into the balance of the dataset a distribution of all x-position values and y-position values can be seen in Figures 4.2b and 4.2c.

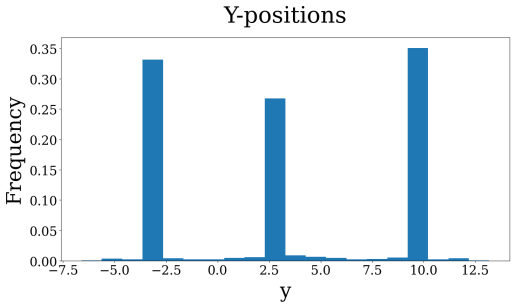
At first glance of observing Figure 4.2a, the dataset looks balanced over the highway with almost all positions occupied in the field of view for the ego vehicle. However, when analyzing Figure 4.2c it can be clearly stated that most of the data points in the dataset for training are placed in the middle of the three lanes. This is a reasonable result for a dataset that constitutes of simulations for highway driving, i.e. that vehicles drive mostly straight on highways. In Table 4.5 the scenes and sequences



(a) All coordinates from the pre-processed ego vehicle coordinate frame centered sequences unveil at first glance a relatively balanced dataset. However, the number of coordinates in regions with y-coordinates of either 9.75, 3.25 and -3.25 are far over-represented due to the highway behavior of surrounding vehicles



(b) The distribution of x-coordinates from (a) for the pre-processed ego vehicle centered coordinate frame displays a normally distributed behavior. There is a higher likeliness of removal during pre-processing for sequences from the original dataset that contains x-coordinates closer to the ego vehicle field of view limits thus explaining the distribution.



(c) The distribution of y-coordinates from (a) for the pre-processed ego vehicle centered coordinate frame exposes the unbalance of the dataset in relation towards positions other than the center of lanes located at 9.75, 3.25 and -3.25.

Figure 4.2: The resulting distribution of ego vehicle centered coordinate frame sequences after pre-processing displays an imbalance towards coordinates located in the center of the lanes. The result is deemed reasonable due to the behavior of highway driving, i.e vehicles mostly drive straight

of the dataset are analyzed. This analysis is based on displaying occurrences of y-position variations through both simple y-position variations and full lane changes in the sequences.

Table 4.5: Analysis of the dataset is presented in two motion categories with further details of their occurrences in percentages of both the total number of sequences and scenes which contains a motion category. The result presents an imbalance towards a low frequency of y-position variations in the sequences

Motion category	**Of all sequences**	**Of all scenes**
Full lane change	2.6%	12.7%
Vehicle lane variation	10.6%	43.8%

From Table 4.5 the unbalance of the total number of lane changes can be viewed. In all sequences of all scenes only 2.6% contains a full lane change and 12.7% of the scenes contain at least one vehicle that performs a lane change. The few amount of lane changes for the model to train on is problematic since these are of high importance in the trajectory planning algorithm.

4.3 Encoders

The selected input to the model's encoders are 2D Cartesian coordinates (x, y positions) which needed to be quantified. Inspiration for the quantization of the positions was taken from [41] K-means approach, but in order for good resolution together with a controlled cluster generation the ego vehicle centered area was split into a grid. The process for the positional grid generation was to first divide both the x- and y- defined ranges into evenly spaced points. The x-positions were divided into spaces of 0.4 meters and y-positions into 0.2 meters, giving two vectors of evenly spaced out points of size m_x for the x grid points and m_y for the y grid points. Inspired by [41], in order to give some stochastic behavior in the evenly spaced grid assignment a normally distributed noise was applied to each element in the vectors

$$\mathbf{p}_x = \tilde{\mathbf{p}}_x + \mathcal{N}(0, 0.2) \tag{4.2a}$$

$$\mathbf{p}_y = \tilde{\mathbf{p}}_y + \mathcal{N}(0, 0.2) \tag{4.2b}$$

where $\tilde{\mathbf{p}}_x$ and $\tilde{\mathbf{p}}_y$ are the evenly spaced out points without noise and $\mathbf{p}_x$ and $\mathbf{p}_y$ are the points with the added noise. By combining each resulting element of (4.2a) with (4.2b) and assigning each combination an integer label, a set of size $(m_x \times m_y)$ grid points with respective coordinates could be constructed giving the set $\mathbb{P}_{\text{pos}}$.

All data points, i.e positions of the vehicles, were assigned a label based on the closest grid point

$$\{\mathbf{g}_{\text{obs}}^{(i)}\}_{i=1}^{N} = p(\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{N}, \mathbb{P}_{\text{pos}}) \tag{4.3}$$

where p computes the euclidean distance (L2) separately for each observed position for all N agents towards all the grid labeled coordinate combinations in the set $\mathbb{P}_{\text{pos}}$

and labels each position based on the minimum distance grid point. This operation resulted in a labeled representation for each observed position for all N agents, $\{\mathbf{g}_{\text{obs}}^{(i)}\}_{i=1}^{N}$. The labeled observed historical positions for each vehicle, $\mathbf{g}^{(i)}$, is of size T_{obs} elements. Important to highlight is that each element only consists of an integer label and not a 2D Cartesian coordinate.

In order to give the encoders more parameters to work with than the integer labeled position grid representation, each assigned label was embedded onto a higher dimensional space. The operation of the embedding can be represented by a look-up table embedding function conditioned on the size of $\mathbb{P}_{\text{pos}}$, i.e $(m_x \times m_y)$

$$\{\mathbf{E}_{\text{obs}}^{(i)}\}_{i=1}^{N} = f(\{\mathbf{g}_{\text{obs}}^{(i)}\}_{i=1}^{N}, m_x, m_y) \tag{4.4}$$

where the embedding function f takes each labeled position in $\{\mathbf{g}_{\text{obs}}^{(i)}\}_{i=1}^{N}$ and transforms it to a unique higher dimensional representation of size d_{model}. The size of each agents observed historical positions therefore goes from a size of T_{obs} in (4.3) to $(T_{\text{obs}},\ d_{\text{model}})$ in $\mathbf{E}_{\text{obs}}^{(i)}$. For better performance and faster convergence during training all values in $\{\mathbf{E}_{\text{obs}}^{(i)}\}_{i=1}^{N}$ were normalized to the range of $[0, 1]$.

4.3.1 Encoder process

The strategic foundation for the encoding modules of the developed model is composed of two encoders referred to as scene- and single-encoder. The scene encoding utilizes a scenic perspective where it encodes the scene, meaning the previous positional history of all agents and encodes this information into a latent variable representation. The input to the scene-encoder, $\{\mathbf{E}_{\text{obs}}^{(i)}\}_{i=1}^{N}$ from equation (4.4), was transformed into a 3D representation of size $(N,\ T_{\text{obs}},\ d_{\text{model}})$. Sine/cosine positional encoding according to (3.10) was applied to give the model information about the temporal ordering. The scene-encoder was constructed in accordance with the same encoder structure presented in section 3.2. The embedded and positional encoded input was flattened and propagated through the scene-encoder to result in a latent variable representation $\mathbf{Z}_{\text{scene}}$ with a dimension of $(N \times T_{\text{obs}},\ d_{\text{model}})$.

The single-encoder is constructed in a similar fashion compared to the scene-encoder. The same higher dimensional embedding from (4.4) and positional encoding (3.10) strategy was applied. The main difference in the single-encoder is that it encodes each embedded trajectory separately. Meaning that input to the single-encoder is of dimension $(N,\ T_{\text{obs}},\ d_{\text{model}})$ but each agents trajectory is encoded separately from each other thus resulting in an output latent variable representation, $\{\mathbf{Z}_{\text{single}}^{(i)}\}_{i=1}^{N}$, of the same size $(T_{obs},\ d_{model})$ for the N surrounding agents.

By utilizing this two-module encoding strategy the idea is to both capture information and represent the scenes' previous and current evolution together with a focus on each separate vehicle's previous behavior. Separating into two modules lets each encoder focus on transforming the input into an appropriate representation for its specific encoding.

4.4 Decoders

The TOT model is constructed to predict one velocity at a time and then use previously predicted velocities for the next prediction. Since this was the approach for generating predictions during inference the training had to match this process. This is where one of the transformer structure strengths and mainly its self-attention mechanism shines. If the model would have been a sequential processing model such as RNN or LSTM, the training would have been significantly slower. This is since for each prediction of a sequence each element would have had to be fed through the network sequentially. The transformer can instead process all the predictions of the whole sequence at once without the need for sequential processing. This non-sequential approach can be achieved by using a combination of teacher forcing approach [59] and masks for the input to the decoders. The idea of the mask for the prediction can be viewed in Figure 4.3.

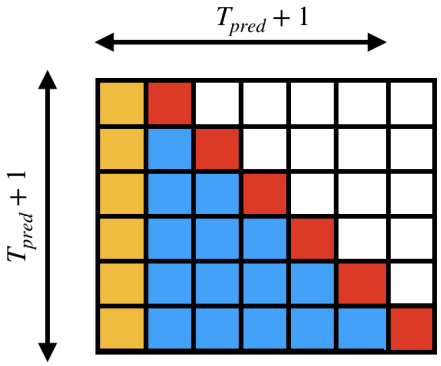


Figure 4.3: Teacher forcing training applied to sequential multi-step time series prediction can be structured by combining one step predictions (red) conditioned on observed information (yellow), ground truth future predictions (blue) and masking out information not to consider (white).

The grid represents time steps of given information for input to the decoder where moving from left to right means increasing time. One row in the grid represents one full future sequence and each row is this same sequence but with different masks. The red squares represent the time step for which to predict and the white squares represent ignored time steps for input. Both the red and the white squares are masked out to simulate a teacher forcing training. The blue squares in the figure are the actual GT future velocities and the yellow squares represent the last observed velocity, both of which are not masked. By utilizing the strategy of giving the transformer the last observed velocity and different numbers of the correct GT in blue to predict the next time step a training strategy capable of processing and generating multi step sequences was created. Since the transformer is also capable of non-sequential processing and does not need to update hidden states a faster training process could be achieved. All the predictions in the red squares were saved for computation of the total loss for the parameter optimization via an optimizer. To summarize the aforementioned approach, the decoder part receives the future GT velocities and utilizing a masking approach it can achieve a one step predictive

process. This process is different during inference since only the first row of figure 4.3 is available, but repeating each step and inputting all previous predictions back into the model will finally roll out and generate predictions for all time steps.

The input to the decoders is in contrast to the input to the encoders, velocities of the future trajectories during training. The process for the velocity grid generation is similar to the position grid generation. The first step was to compute the minimum and maximum velocity measured for the vehicles in the dataset. The maximum and minimum values for both x- and y-velocities were used as limits towards a range that was split into intervals of 0.01 meters per time step for x- and 0.008 meters per time step for y-velocities. The result was two vectors of size $m_{\dot{x}}$ for x velocities and $m_{\dot{y}}$ for y velocities. A normally distributed noise was applied to each element of the vectors

$$\mathbf{p}_{\dot{x}} = \tilde{\mathbf{p}}_{\dot{x}} + \mathcal{N}(0, 0.002) \tag{4.5a}$$

$$\mathbf{p}_{\dot{y}} = \tilde{\mathbf{p}}_{\dot{y}} + \mathcal{N}(0, 0.0002) \tag{4.5b}$$

where $\tilde{\mathbf{p}}_{\dot{x}}$ and $\tilde{\mathbf{p}}_{\dot{y}}$ are the evenly spaced out velocity grid points without noise and $\mathbf{p}_{\dot{x}}$ and $\mathbf{p}_{\dot{y}}$ are with the added noise. Each element of both (4.5a) were combined with each element of (4.5b) and was assigned an integer label which resulted in a set of size $(m_{\dot{x}} \times m_{\dot{y}})$ referenced as $\mathbb{P}_{\text{vel}}$.

Each future velocity of every surrounding agent was given an label

$$\{\dot{\mathbf{g}}_{\text{GT}}^{(i)}\}_{i=1}^{N} = o(\{(\dot{\mathbf{x}}_{\text{GT}}^{(i)}, \dot{\mathbf{y}}_{\text{GT}}^{(i)})\}_{i=1}^{N}, \mathbb{P}_{\text{vel}}) \tag{4.6}$$

where o computes the L2 distance for all observed velocities for all N agents towards the grid labeled velocity set $\mathbb{P}_{\text{vel}}$ and selects the minimum distance. The new integer labeled velocities results in $\{\dot{\mathbf{g}}_{\text{GT}}^{(i)}\}_{i=1}^{N}$. For the higher dimensional space embedding the same function e presented in (4.4) was used

$$\{\dot{\mathbf{E}}_{\text{GT}}^{(i)}\}_{i=1}^{N} = f(\{\dot{\mathbf{g}}_{\text{GT}}^{(i)}\}_{i=1}^{N}, m_{\dot{x}}, m_{\dot{y}}) \tag{4.7}$$

where $\{\dot{\mathbf{E}}_{\text{GT}}^{(i)}\}_{i=1}^{N}$ is the higher dimensional representation. This works since the function f conditions on the size of the total number of combinations to create unique vectors of d_{model} for each integer label. All values of $\{\dot{\mathbf{E}}_{\text{GT}}^{(i)}\}_{i=1}^{N}$ were also normalised to the range of $[0, 1]$ with the aforementioned motivation for faster convergence.

4.4.1 Decoder process

Following the encoder, the decoding part of the TOT comprises of two specific decoder modules: scene-decoder and single-decoder. What differentiates these two modules is mainly what is included from which encoder and how they are connected. The main idea of a transformer decoder is to receive two inputs: one latent variable representation from an encoder, often referred to as memory, and one separate embedded input. The decoding structure follows a two step iterative approach for each

surrounding agent of the ego vehicle. The first step is the scene-decoder which receives $\mathbf{Z}_{\text{scene}}$ for memory and inputs an embedded velocity representation according to (4.7) for one agent with dimension $(T_{\text{obs}}, d_{\text{model}})$. The output from the scene-decoder is of the same dimension as the embedded velocity input $(T_{\text{pred}}, d_{\text{model}})$.

The second decoder, the single-decoder, receives the output from the scene-decoder as input. The memory for the single-decoder is the specific surrounding vehicle's observed embedded latent variable representation, $\mathbf{Z}_{\text{single}}^{(i)}$, of the dimension of size $(T_{\text{obs}}, d_{\text{model}})$. Important to highlight is that the expected output from the transformer model is a matrix containing the probabilities of which class from the set $\mathbb{P}_{\text{vel}}$ for each time step up to T_{pred}. Therefore, the output from the single-decoder gets passed through a linear layer to transform from dimension $(T_{\text{pred}}, d_{\text{model}})$ to $(T_{\text{pred}}, S_{\mathbb{P}_{\text{vel}}})$ where $S_{\mathbb{P}_{\text{vel}}}$ is the size of the set $\mathbb{P}_{\text{vel}}$. This process was repeated for each surrounding vehicle which finally resulted in an output of size $(N, T_{\text{pred}}, S_{\mathbb{P}_{\text{vel}}})$ from the model.

Since the output from the model is a probability distribution of class labels of the velocity grid, a multi class loss function could be applied in order to get a loss for the gradient descent optimization of the parameters. If the model is instead used during inference the most probable velocities get selected at each iteration to produce a prediction of required length T_{pred}. During inference time and implementation, the respective predicted velocities were also combined with the last observed position to generate future positions of the surrounding vehicles.

4.5 Uncertainty modeling

The selected method for uncertainty expression during inference was the Monte Carlo Dropout method [55]. Presented in section 3.4.2 there exist a number of strategies to express uncertainty from DL models. What differentiates the Monte Carlo Dropout method is the avoidance of having to substantially increase the number of optimization parameters. Utilizing a BNN strategy results in an increase in the number of parameters if for example each parameter was expressed as a distribution defined by a mean and variance. Applying a Deep Ensemble learning strategy would instead force to produce several model instances increasing the number of parameters. Since the size and number of parameters for transformer-based models are fairly large, limiting options for uncertainty modeling strategies, the Monte Carlo method was deemed a favorable option. What the Monte Carlo dropout method will produce is several different predicted modes during inference. The difference between the predicted modes will be a measure of the uncertainty of the scene evolution, meaning that if all modes produce the same trajectory the model is certain and vice versa. The selected number of modes during the inference evaluation of the observer was 10 modes.

4.6 Implementation and Evaluation of observer

The observer was trained for 23 epochs using the Adam optimizer [60] with a learning rate of 0.001. The learning rate was scheduled for the decay of 90% of the previous epoch's learning rate after 10 epochs. A linear warm-up learning rate of the first 3 epochs was utilized to reduce early over-fitting of the highly unbalanced dataset. The learning rate curve can be seen in Figure 4.4. The loss was computed with a cross-entropy function which compares the predicted probabilities of which velocity in $\mathbb{P}_{\text{vel}}$ to GT velocities. Train/Validation/Test split was selected to 80/10/10 by the fact that this partition was needed in order to give the validation and test set a similar distribution of scene variations compared to the training data. The training process used an NVIDIA Quadro T1000 with 4GB of RAM GPU with a batch size of 3, i.e. 3 scenes with 5 surrounding vehicles in each scene.

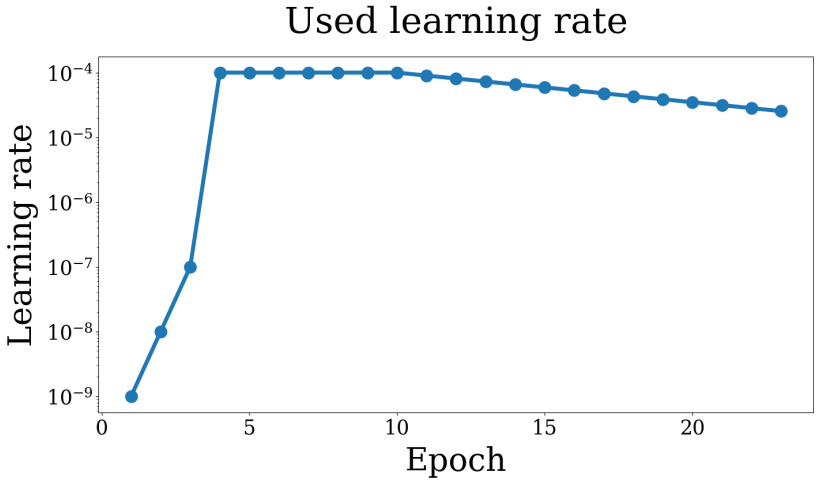


Figure 4.4: The selected learning rates for training constitute of an initial learning rate warm-up phase of 3 epochs. The initial learning rates are followed by a constant learning rate until epoch 10 where a 10% decrease in the previous epoch's learning rate is implemented until the end of training. The learning rate is plotted on a log-based scale.

With the development of an uncertainty aware DL model, the option of creating two instances for implementation together with the model predictive controller was possible. One that is a deterministic model that utilizes no Monte Carlo dropout method, referenced as TOT, and one multi-modal and uncertainty expressing model, referenced as TOT-MC. Important to highlight is that the only difference between the deterministic and the uncertainty-expressing model is how the inference process is generated. The TOT-MC applies dropout functionality and mode iterations to predict several outcomes. However, both the TOT and the TOT-MC are instances of the same trained model.

The two variations of the TOT model with and without the Monte Carlo Dropout method were evaluated against an extension of the baseline CV prediction model. The extended CV model also incorporates a constant heading for y-positions changes in its trajectory prediction. This extension produces a more accurate prediction re-

sult, i.e. a more difficult baseline for comparison. In order to produce a comparison between the models, the TOT-MC error metrics were compared as the mean of each time step of all predicted modes. The analysis was performed through two areas of interest. The first focused on a quantitative analysis of the testing dataset. This was performed through the respective model's ADE and FDE metrics together with a time-step error investigation. The second part focused on a qualitative study through case investigations. The cases were selected to display the intricate behavior of the different models during interesting scenes and how their predictions differentiate from each other.

5

Trajectory planning using robust model predictive control

The robust MPC architecture for the highway autopilot primarily adopts the same structure as described for the baseline version from section 2.3. That is, plan an optimal trajectory of states and belonging control actions by utilizing three separate MPCs and a DM to choose the cost-minimizing trajectory based on (2.23). However, in this case, the trajectories of the surrounding vehicles were not assumed to be deterministic, therefore their uncertainties had to be taken into account in the optimization.

5.1 Constraint formulation

Instead of using the constant velocity model shown in (2.13) to estimate the positions of the surrounding vehicles, the output from the TOT-MC model was used. Here, the prediction of the future trajectories of the surrounding vehicles is the mean of all modes from the TOT-MC. The purpose of the vehicle trajectory predictions is the same as for the baseline, to be able to properly confine the constraints to the surrounding vehicles during the prediction horizon of T_{pred}. However, since the trajectory predictions of the surrounding vehicles coming from the TOT-MC are non-deterministic, the formulation of the constraints differs. Furthermore, to extend the analysis of robustness, slack variables were included in the constraints and optimization. Slack variables are used to soften collision avoidance constraints in order to always obtain feasible solutions, even if a trajectory plan lies inside one of these constraints. A schematic view of the trajectory predictions of the surrounding vehicles can be seen in Figure 5.1.

Similar to the idea with the tube-based MPC approach, where modeling uncertainties of the ego vehicle are accounted for in the optimization problem, the approach used in this RMPC instead accounts for prediction uncertainties of the surrounding vehicles. In Figure 5.1, 2D regions are formed around all trajectory modes provided by the TOT-MC model and represent the uncertainty of the predictions. These uncertainty regions were accounted for in the RMPC solution by modifying the collision avoidance constraints accordingly. For the collision avoidance constraint equations, t has been omitted for brevity but is included in the cost function and final optimization formulation for completeness.

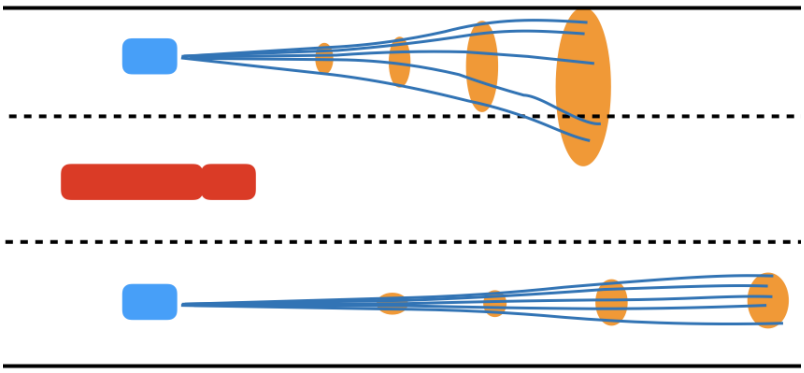


Figure 5.1: Blue lines correspond to uncertain trajectory predictions of the surrounding vehicles from the TOT-MC model, yellow 2D areas around the predictions are the corresponding uncertainty measures in x- and y directions. The red rectangle corresponds to the ego vehicle and the blue rectangles are the surrounding vehicles.

For the trailing RMPC, as in the case of the deterministic MPC, the constraint is determined by holding a specified distance to the vehicle in front. Additionally, an extension parameter was added to the constraint for the purpose of including the uncertainty from the TOT-MC model. The modified robust constraint including the slack variable can be seen in the equation below

$$c_{\text{trail,R}}(x^e, \hat{x}^{(l)}, \hat{v}_x^{(l)}, \alpha_x^{(l)}, \zeta^{(l)}) = \zeta^{(l)} + d + L^{\text{tractor}} + T\hat{v}_x^{(l)} + \phi\alpha_x^{(l)} + x^e - \hat{x}^{(l)} < 0 \quad (5.1)$$

where subscript R is a reference towards the robust constraint. The variables in (5.1) that differ from the constraint in (2.14) is $\phi\alpha_x^{(l)}$, which represents the uncertainty in the predicted x-position of the leading vehicle. ϕ is a scaling parameter defining how much the RMPC trusts the uncertainty from the TOT-MC during the prediction. The uncertainty parameter for a vehicle i was obtained by computing the standard deviation of all modes in every time step

$$\alpha_x^{(i)} = \sqrt{\frac{\sum_{q=0}^{N_m} \left(\hat{x}^{(i)} - \hat{x}_q^{(i)}\right)^2}{N_m - 1}} \quad (5.2)$$

where $\hat{x}_q^{(i)}$ is the predicted x-position of mode q of vehicle i, $\hat{x}^{(i)}$ is the average over all predicted modes of vehicle i at time step t and N_m is the total number of modes. A scenario where the modified constraint is visualized can be seen in Figure 5.2.

In the case of the two lane change RMPCs, the constraints were modified in a similar manner as for the trailing RMPC. The structure of the constraint is the same as in (5.3), however, prediction uncertainties from the TOT-MC model are added to extend the constraints of the surrounding vehicles. The following equation represents the modified robust constraints

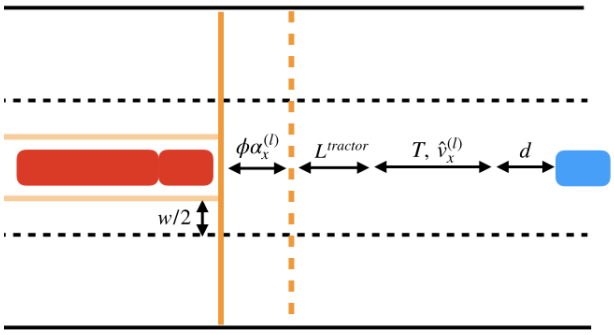


Figure 5.2: Robust trailing constraint where the ego-vehicle is depicted in red and the leading vehicle is depicted in blue. The longitudinal constraint, depicted in orange, is determined by holding a specified safety distance that is dependent on the length of the tractor L^{tractor}, the safe time headway T, the predicted velocity of the leading vehicle $\hat{v}_x^{(l)}$, a required minimum longitudinal distance to the vehicle in front d and prediction uncertainty parameter $\phi\alpha_x^{(l)}$ that defines the uncertainty of the predicted x-position of the leading vehicle. The ego vehicle is constrained laterally by the lane markings including a safety distance of the maximum lateral span of the truck, w.

$$b_R^{(i)}(x^e, y^e, \hat{x}^{(i)}, \hat{y}^{(i)}, \alpha_x^{(i)}, \alpha_y^{(i)}) = \xi_{1,R}(\xi_{2,R} + \xi_{3,R}) + \xi_{4,R} \tag{5.3a}$$

$$\xi_{1,R} = \frac{\epsilon_0^{(i)}(\hat{y}^{(i)}, \alpha_y^{(i)})}{2} \tag{5.3b}$$

$$\xi_{2,R} = \tanh(x^e - \hat{x}^{(i)} + \epsilon_1^{(i)} + \phi\alpha_x^{(i)}) \tag{5.3c}$$

$$\xi_{3,R} = \tanh(\hat{x}^{(i)} - x^e + \epsilon_2^{(i)} + \phi\alpha_x^{(i)}) \tag{5.3d}$$

$$\xi_{4,R} = \epsilon_3^{(i)}(y^e, \hat{y}^{(i)}) \tag{5.3e}$$

where $\alpha_y^{(i)}$ is introduced as uncertainty in y positions of vehicle i at time step t

$$\alpha_y^{(i)} = \sqrt{\frac{\sum_{q=0}^{N_m} \left(\hat{y}^{(i)} - \hat{y}_q^{(i)}\right)^2}{N_m - 1}} \tag{5.4}$$

and is used to scale the constraint laterally where $\hat{y}_q^{(i)}$ is the predicted y-position of mode q of the vehicle i and $\hat{y}^{(i)}$ is the average over all predicted modes of the vehicle i at time step t. The robust lane change constraint can be formulated as

$$c_{\nu,\mathrm{R}}(x^e, y^e, \hat{x}^{(i)}, \hat{y}^{(i)}, \alpha_x^{(i)}, \alpha_y^{(i)}, \zeta^{(i)}) = F_\nu^{(i)} \left(b_{\mathrm{R}}^{(i)}(\cdot) - y^e + w + \zeta^{(i)}\right) < 0 \tag{5.5}$$

where $c_{\nu,\mathrm{R}}$ represents the robust lane change constraints. A scenario where the robust lane change collision avoidance constraints are visualized can be seen in Figure 5.3.

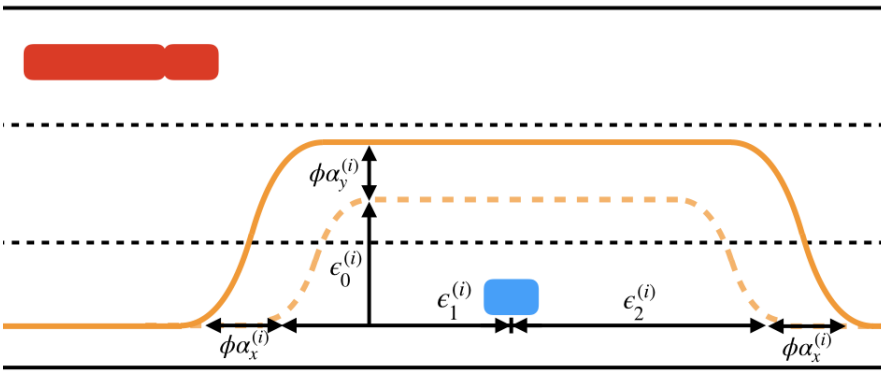


Figure 5.3: Robust lane change constraint where the ego-vehicle is depicted in red and the surrounding vehicle is depicted in blue. The constraint, depicted in orange, for vehicle i is defined by a continuous tanh-function that is scaled longitudinally by $\epsilon_1^{(i)}$ and $\epsilon_2^{(i)}$. $\epsilon_0^{(i)}$ scales the constraint laterally and $\epsilon_3^{(i)}$ shifts the constraint to the correct lateral position on the highway. Furthermore, uncertainty from the TOT-MC model scales the constraint longitudinally and laterally by the variables $\phi\alpha_x^{(i)}$ and $\phi\alpha_y^{(i)}$ respectively.

5.2 Robust optimization formulation

The robust optimization formulation to a large extent follows the method used in subsection 2.3.2. Although, drivability metrics and slack were added to the baseline MPC cost function in order to have a larger variety of comparison measures. Consequently, the adapted cost function for each controller can be expressed as

$$\begin{aligned} J_R(\mathbf{s}^e, \mathbf{u}^e) &= L_f(\mathbf{s}^e(T_{\text{pred}})) + q_\zeta \boldsymbol{\zeta}(T_{\text{pred}})^T \boldsymbol{\zeta}(T_{\text{pred}}) \\ &+ \sum_{t=0}^{T_{pred}-1} (\mathbf{s}^e(t) - \mathbf{s}^e_{\text{ref}}(t))^T \mathbf{Q}_{s,R} (\mathbf{s}^e(t) - \mathbf{s}^e_{\text{ref}}(t)) + q_\zeta \boldsymbol{\zeta}(t)^T \boldsymbol{\zeta}(\boldsymbol{t}) \\ &+ \sum_{t=0}^{T_{pred}-1} (\mathbf{u}^e(t) - \mathbf{u}^e_{\text{ref}}(t))^T \mathbf{R} (\mathbf{u}^e(t) - \mathbf{u}^e_{\text{ref}}(t)) \\ &+ \sum_{t=0}^{T_{pred}-2} (\mathbf{u}^e(t+1) - \mathbf{u}^e(t))^T \mathbf{R}_c (\mathbf{u}^e(t+1) - \mathbf{u}^e(t)). \end{aligned} \tag{5.6}$$

The state deviation penalty was updated by removing penalties for deviating from the reference angles, while the control and terminal state penalty matrix were kept the same. This was a motivated change in order to give further incentive for the ego vehicle to perform lane changes.

$$\mathbf{Q}_{s,R} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 40 & 0 & 0 & 0 \\ 0 & 0 & 300 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \mathbf{R} = \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix} \tag{5.7}$$

The reference vectors are equal to those found in (2.20). The drivability metrics, referred to as jerk, are included as the change in control input between two time steps and have a penalty matrix defined as

$$\mathbf{R}_c = \begin{bmatrix} 10^7 & 0 \\ 0 & 10^5 \end{bmatrix} \tag{5.8}$$

The slack variable ζ represents how much the collision avoidance constraints have to be loosened in order to obtain feasible solutions from the MPCs at every time step. The cost of slack, $q_\zeta = 10^{10}$, was chosen to this value so that optimization would not benefit from slacking the constraints if it was not forced to.

General constraints for each robust controller can be formulated by combining the physical limitations on the ego vehicle (2.5) together with the robust constraints in (5.1) and (5.5)

$$h^e_{\kappa,R}(\mathbf{s}^e_\kappa, \mathbf{u}^e_\kappa, \hat{x}^{(i)}, \hat{y}^{(i)}, \alpha^{(i)}_x, \alpha^{(i)}_y, \zeta^{(i)}) \leq 0, \ \forall i \in \mathbb{V}_\kappa \tag{5.9}$$

where $h^e_{\kappa,R}$ is the constraint function for the robust MPCs. With the cost function defined in (5.6) and the constraints (5.9), the robust optimization could be formulated

$$\begin{aligned} \min_{\mathbf{u}^e_\kappa(t:t+T_{\text{pred}}-1|t)} \quad & J_{\kappa,R}\left(\mathbf{s}^e_{\text{init}}, \ \mathbf{u}^e_\kappa(t:t+T_{\text{pred}}-1|t)\right) \\ \text{s.t.} \ & \mathbf{s}^e_\kappa(t+n+1|t) = g(\mathbf{s}^e_\kappa(t+n|t), \mathbf{u}^e_\kappa(t+n|t)) \\ & (\hat{\mathbf{x}}^{(i)}, \hat{\mathbf{y}}^{(i)}, \boldsymbol{\alpha}^{(i)}_x, \boldsymbol{\alpha}^{(i)}_y) = \Pi_{\text{TOT-MC}}(\{\mathbf{x}^{(k)}, \mathbf{y}^{(k)}\}_{k=1}^N), \ \forall i \in \mathbb{V}_\kappa \\ & h^e_{\kappa,R}(\mathbf{s}^e_\kappa(t+n|t), \mathbf{u}^e_\kappa(t+n|t), \hat{x}^{(i)}(t+n|t), \hat{y}^{(i)}(t+n|t), \\ & \qquad \alpha^{(i)}_x(t+n|t), \alpha^{(i)}_y(t+n|t), \zeta^{(i)}(t+n|t)) \leq 0 \\ & \mathbf{s}^e_\kappa(t|t) = \mathbf{s}^e_{\text{init}} \\ & \forall \ n \in \ \{0, \ldots, T_{\text{pred}}-1\} \end{aligned} \tag{5.10}$$

where $\Pi_{\text{TOT-MC}}$ corresponds to the TOT-MC predictor, which predicts surrounding vehicles' positions throughout the prediction horizon and produces prediction uncertainties. The TOT-MC model receives as input all surrounding vehicles observed positions to produce the trajectory predictions for vehicles currently of interest for controller κ. Furthermore, the terminal state at time step T_{pred} also needs to conform to the inequality constraints stated in the optimization.

To choose the cost-minimizing trajectory out of the three robust MPCs an equivalent DM to the one defined in (2.23) was used but with the robust cost function

$$J_{\kappa,tot,R} = a_1 J_{\kappa,R} + a_2 J_{\text{change}}. \tag{5.11}$$

5.3 Evaluation of robust MPC

To evaluate the robust MPC, two baseline MPC versions were used for comparison. The first baseline constitutes of the MPC described in section 2.3, with the additional cost function elements for slack and jerk. The other baseline uses the optimization formulation as in (2.22) with the additional cost function elements, although, instead of using the CV-model in (2.13), it uses deterministic TOT-predictions. Regarding the evaluation of the robust MPC, one sigma level uncertainty, i.e ϕ=1 in (5.1) and (5.3), was applied during the entire prediction horizon.

The comparison between the different controllers was made in terms of the total cost, constraint violation, and drivability through jerk. The first method of comparison was to simulate each controller over 40 episodes for quantitative analysis. These simulation scenarios represented normal driving scenarios, similar to those the TOT model had been trained on. All controllers were evaluated on the same simulations. Regarding the total cost, the cost function (5.6) was post-computed but only for the first time-step in every optimization. This method was selected in order to be able to analyze the actual outcome of decisions made from each controller

$$\begin{aligned} J_T = (\mathbf{s}^e(t) - \mathbf{s}^e_{\text{ref}}(t))^T \mathbf{Q}_s (\mathbf{s}^e(t) - \mathbf{s}^e_{\text{ref}}(t)) + q_\zeta \boldsymbol{\zeta}(t)^T \boldsymbol{\zeta}(t) \\ + (\mathbf{u}^e(t) - \mathbf{u}^e_{\text{ref}}(t))^T \mathbf{R} (\mathbf{u}^e(t) - \mathbf{u}^e_{\text{ref}}(t)) \\ + (\mathbf{u}^e(t+1) - \mathbf{u}^e(t))^T \mathbf{R}_c (\mathbf{u}^e(t+1) - \mathbf{u}^e(t)) \end{aligned} \tag{5.12}$$

where J_T corresponds to the total cost at time-step t. These costs were summed up into a total cost for all simulations. Constraint violation is highly correlated with the safety of the trajectory since violation of a constraint might lead to a crash. The cost of violating constraints was determined by observing the slack cost based on the deterministic collision avoidance constraints posed in (2.14) and (2.16). A positive slack cost means that the solution at a specific time step is violating a constraint. The cost of slack was obtained by extracting the specific term of slack from the cost function

$$J_s = q_\zeta \boldsymbol{\zeta}(t)^T \boldsymbol{\zeta}(t) \tag{5.13}$$

where J_s corresponds to the slack cost at time-step t. The drivability cost was obtained by

$$J_{jk} = (\mathbf{u}^e(t+1) - \mathbf{u}^e(t))^T \, R_c \, (\mathbf{u}^e(t+1) - \mathbf{u}^e(t)) \tag{5.14}$$

where J_{jk} corresponds to the jerk cost at time-step t. For each controller, the individual state and control deviation costs were stored for comparison.

The second method of comparison was done by a qualitative study of a more challenging scenario where the importance of a good observer should be prominent. The scenario for the case study was selected with inspiration from the Volvo Trucks safety report 2017 [2]. From this report, common crash scenarios that could be mimicked in a highway scenario were further looked into.

The first case of interest is noted as case B4 in [2]. In case B4, the truck crashes into a slower-moving car that is traveling in the same direction. The report states that the collision is caused by the truck and that the common reasons for it to occur is inattention, low visibility, or that the other vehicle is not noticed by the truck driver. The other safety-critical case of interest is case B8. This is a case where a collision is imminent due to a surrounding vehicle or the truck itself is merging into another lane. Most often the fault lies on the truck driver, but it can also lie on the driver of the surrounding vehicle. Similar to case B4, the cause of this scenario is often a lack of attention, low visibility or too much swerving.

A situation was created such that the ego vehicle would approach a scenario where B4 and B8 were present after an initial settling period. The ego vehicle was initialized in the middle lane, with a passive vehicle with the same velocity close in front of it. An aggressive vehicle was initialized slightly in front of the ego vehicle in the left-most lane, with a lower velocity. A slow vehicle was initialized far ahead in the leftmost lane so that the aggressive vehicle would be forced to merge into the middle lane- initializing a merging scenario where the passive vehicle in front of the ego needs to heavily break, causing a safety-critical situation. An illustration of the safety critical scenario can be seen in Figure 5.4.

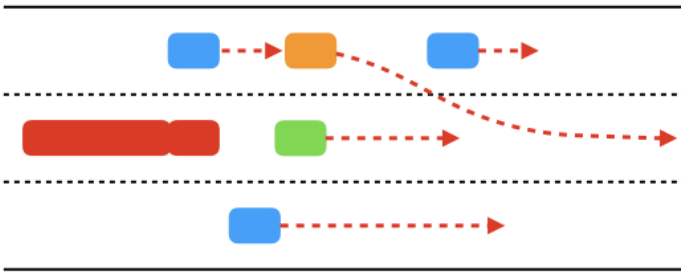


Figure 5.4: Case scenario created with inspiration from safety scenarios B4 and B8 from Volvos safety report. The aggressive orange vehicle in the left lane is merging into the adjacent lane in front of the green vehicle in the middle lane. This invokes a heavy break, which the ego vehicle, depicted as a red rectangle, has to respond to.

The controllers were analyzed and compared towards each other for the above mentioned case in terms of the total cost, slack cost, and drivability as seen in (5.12), (5.13) and (5.14). Although, as these are safety-critical situations where crashes often happen, the main focus area of comparison was the slack cost. Furthermore, the jerk cost was analyzed to deem whether the lane change could be captured earlier using the transformer-based observer.

6

Results and analysis of developed observer and robust controller

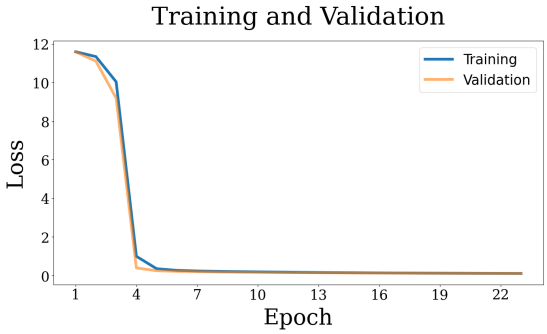
In this chapter, the obtained results through the used methodologies from chapters 4 and 5 are presented. Firstly, the qualitative and quantitative results for the TOT and TOT-MC models compared to the baseline CV are presented in section 6.1. Thereafter, the robust MPC results are presented in section 6.2.

6.1 Observer

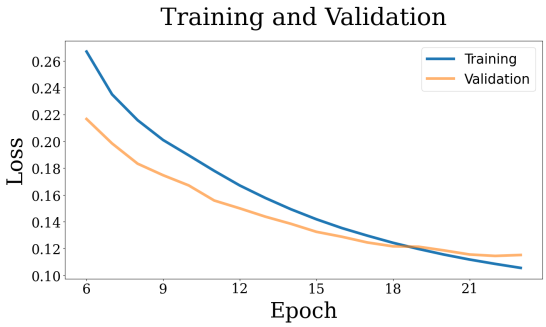
The result for training the TOT model on the training data set with evaluation on the validation data set for every epoch is displayed in Figure 6.1. In Figure 6.1a the training result can be seen for all the epochs. Clearly, the model has a slow start in optimizing the parameters for the first 3 epochs which is explained by the learning rate warm-up phase. After the first epochs, the learning rate is increased which significantly impacts the minimization of the loss. From Figure 6.1b it can be viewed that at epoch 19 a separation between the training loss and validation loss starts to occur. Training loss continues to decrease while the validation loss stagnates thus motivating to select the model from epoch 19 to remove the risk of overfitting. The model from epoch 19 is quantitatively evaluated on the test data in section 6.1.1 and qualitatively evaluated through specifically selected cases from the test data in 6.1.2.

6.1.1 Quantitative analysis

In Table 6.1 the ADE and FDE values for the three different models are presented. At first glance, it can be viewed that the TOT and TOT-MC models outperform the CV model in terms of ADE and FDE. The TOT and TOT-MC have slight variations in their performance where the model using the Monte Carlo dropout method performs slightly better for FDE error metrics. This could be interpreted as a result of the mean of the multiple modes. In scenarios where the TOT predicts erroneously the TOT-MC may also predict this for one mode but the other modes may coincide with the future GT. This may therefore minimize the final error for time steps far into the future. The better performance for the TOT-MC in longer horizons can also be seen in Figure 6.2 where the average root mean square error (RMSE) over each time step in the prediction horizon is displayed. Once again both



(a) The learning rate warm-up phase for the first epochs limits the training of the parameters in the model and thus the minimization of the loss. From epoch 3 the loss quickly converges towards lower values for both the training and validation loss.



(b) Indications towards potential overfitting and increase of variance start occurring from epoch 19 which motivates to select the model from epoch 19 for implementation. This is further motivated by the fact that the validation converges while the training loss continues to decrease.

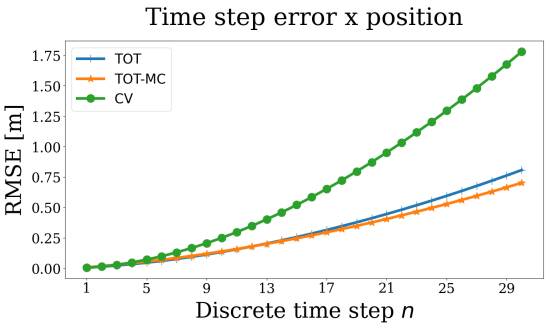
Figure 6.1: The training and validation loss during optimization of the developed model.

the TOT and TOT-MC models outperform the CV baseline model in terms of errors in predictions over the horizon. Interesting to note is that the CV model increases its error faster over the horizon compared to the TOT and TOT-MC. This could further give motivation for the developed models that they may be more appropriate for usage in prediction horizons longer than what is investigated.

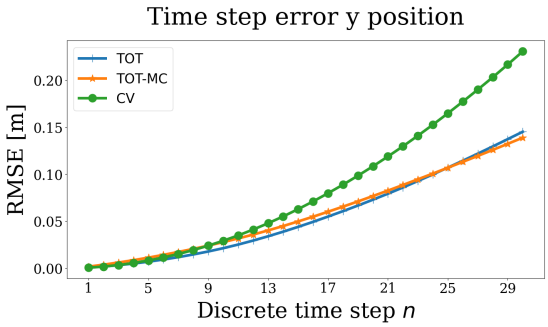
Table 6.1: The evaluation of the test data shows that the developed models performs better than the selected CV baseline model in terms of ADE and FDE metrics.

Model	**ADE (x/y) [m]**	**FDE (x/y) [m]**
Constant Velocity	0.673/0.084	1.779/0.231
TOT	0.319/**0.056**	0.808/0.145
TOT-MC	**0.293**/0.059	**0.702**/**0.139**

All models have an equal performance from an RMSE perspective for the first time steps in the prediction horizon. Instead of looking at the RMSE of the error, the distribution of error at each time step can be analyzed to observe how the error progresses through the prediction horizon. In Figures 6.3 and 6.4 the error distributions for the TOT and TOT-MC are presented against the error distribution of the CV model for time steps 1, 15 and 30 in the prediction horizon for x-positions. In the first time step error comparisons in Figures 6.3a and 6.4a is the behavior of all models relatively equal and achieve almost no error. Moving towards later time steps in the prediction horizon, all models prediction errors start to increase. However, it is interesting to also look at the distribution of the error in Figures



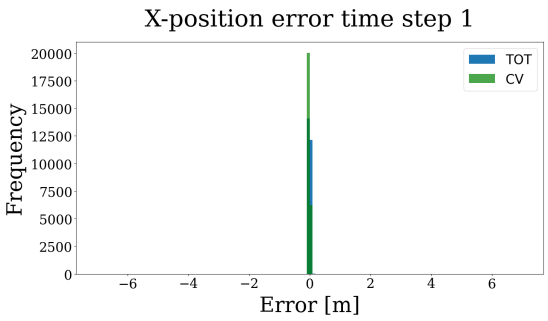
(a) RMSE error for x positions for each discrete time step n with a sampling time of 0.2 seconds in the prediction horizon.



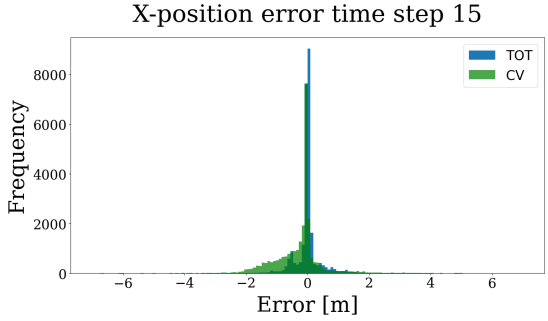
(b) RMSE error for y positions for each discrete time step n with a sampling time of 0.2 seconds in the prediction horizon.

Figure 6.2: The RMSE computed errors for the evaluated models shows favorable results for the developed TOT and the TOT-MC models over the prediction horizon.

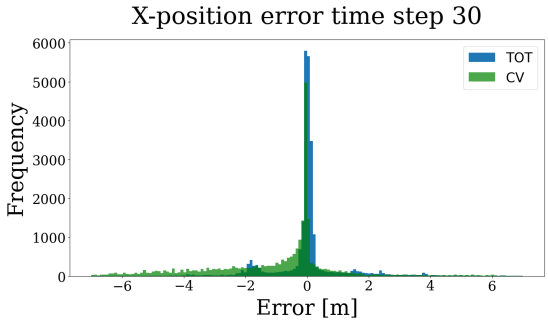
6.3c and 6.4c where the TOT and TOT-MC distributions are more centered around zero. The CV baseline error distribution displays a result with a balance toward predicting more negative errors. These negative errors are results that show that the CV model is more likely to overshoot predictions, i.e. predicts larger x-positions compared to GT future.



(a) Time step 1 error distribution

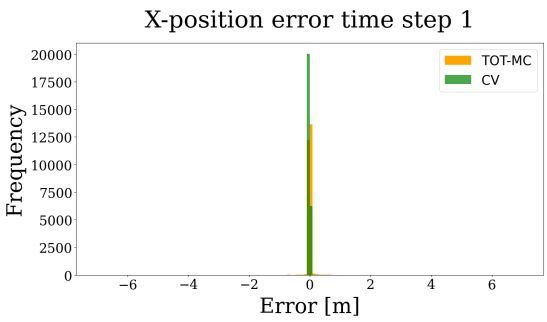


(b) Time step 15 error distribution

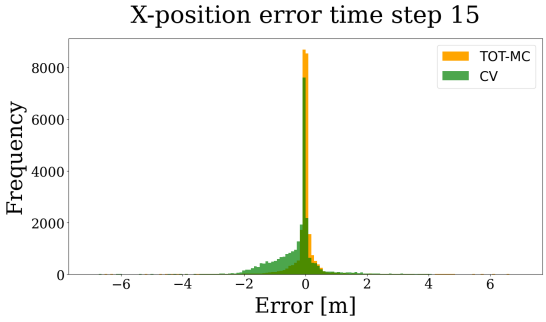


(c) Time step 30 error distribution

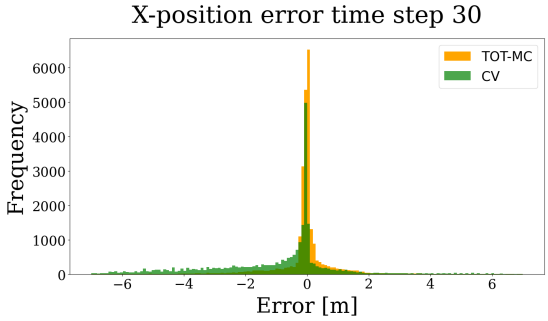
Figure 6.3: Time step error distribution for time steps 1, 15 and 30 for the TOT model compared to the CV baseline model.



(a) Time step 1 error distribution



(b) Time step 15 error distribution



(c) Time step 30 error distribution

Figure 6.4: Error distribution for time steps 1, 15 and 30 in the prediction horizon for the TOT-MC model compared to the CV baseline model.

Important to note is that in the Table 6.1 the results correspond to the evaluation of the entire test dataset. Previously mentioned is the notion of the high unbalance in the dataset towards straight driving. The displayed result above can therefore be interpreted as a result in terms of the models' predictive performance during straight highway driving since these scenarios highly impact the average prediction error computations. Table 6.2 displays the ADE and FDE results for the models' performance for predicting sequences of surrounding vehicles in scenes where at least one vehicle is performing a lane-changing maneuver. The lane-changing maneuvers are classified as a shift in y-position during the whole prediction horizon of more than 3.25 meters.

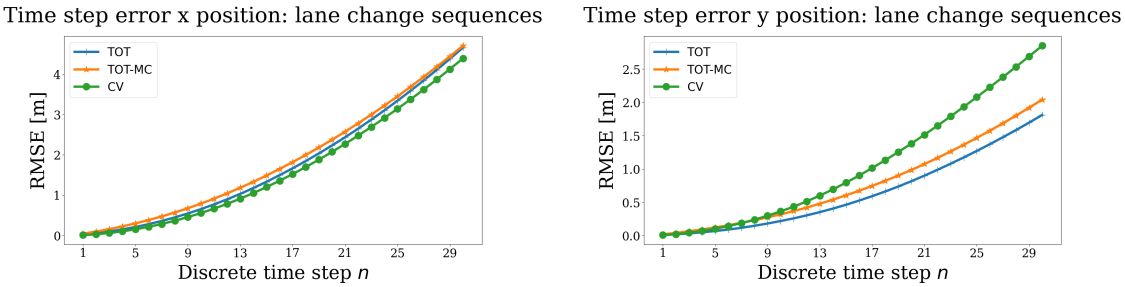
The result indicates a different outcome compared to the result in Table 6.1 where the CV model was outperformed in terms of minimization of the metrics. In table 6.2 the CV model is better at predicting x-positions compared to the TOT and the TOT-MC models. However, it is important to note that the prediction performance for y-positions is substantially better, especially considering FDE, for both the TOT and the TOT-MC compared to the CV model.

The RMSE time step errors for lane changing sequences for all models are further presented in Figure 6.5. The result emphasizes that the CV baseline model is better for x-position predictions while the TOT and TOT-MC outperform in terms of y-position errors. A clear motivation for this is that the TOT and TOT-MC are better at predicting difficult scenarios of lane-changing maneuvers. These results

are further strengthened in the case study presented in sections 6.1.2.

Table 6.2: Test data evaluation with ADE and FDE on scenes containing at least one lane-changing sequence.

Model	**ADE (x/y) [m]**	**FDE (x/y) [m]**
Constant Velocity	**1.610**/1.062	**4.395**/2.850
TOT	1.744/**0.647**	4.663/**1.811**
TOT-MC	1.858/0.781	4.709/2.042



(a) RMSE error for x positions for each discrete time step n with a sampling time of 0.2 seconds in the prediction horizon.

(b) RMSE error for y positions for each discrete time step n with a sampling time of 0.2 seconds in the prediction horizon.

Figure 6.5: The RMSE time step comparison showcase that x position errors are relatively equal for lane-changing sequences. However, for y positions the TOT and TOT-MC are still achieving a lower RMSE error than the CV model.

6.1.2 Qualitative analysis

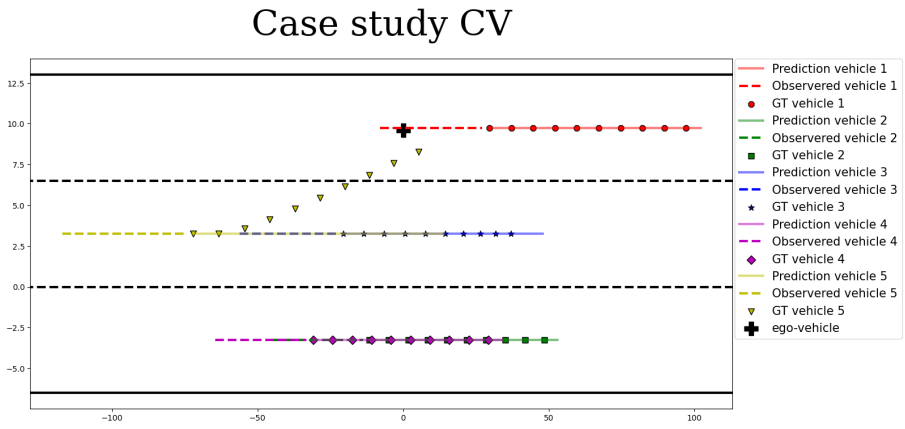
The case study aims to further give insight into the predictive capabilities of the models. It also aims to further extend the analysis of what the previously presented errors may look like in a driving scenario. The selected cases aims at displaying scenarios that focus on:

- Figure 6.6: Scene where there are multiple plausible future options for surrounding vehicle trajectories.
- Figure 6.7: Displays the common straight driving scenario on the highway.
- Figure 6.8: Scene where no lane change is present but TOT-MC suggests lane change.
- Figure 6.9: Scene that presents poor performances from the TOT and TOT-MC.
- Figure 6.10: Lane-changing scene where the TOT and TOT-MC are certain and accurate about the future trajectory.

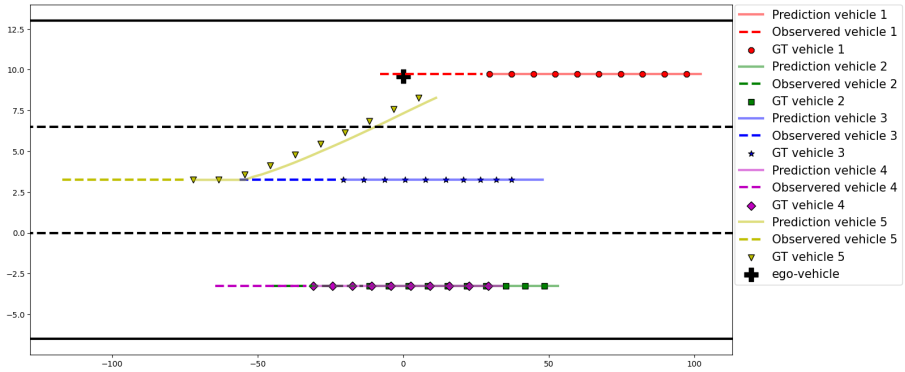
The cases presented in Figures 6.6-6.10 all follow the same design where colors represent separate surrounding vehicles with their observed positions in dashed lines and GT future trajectories by individual markers. An important thing to highlight is that the prediction length, T_{pred}, is 30 discrete time steps of 0.2 second intervals and the markers only display every third future time step. The observed positions of each surrounding vehicle are represented as a continuous dashed line, however, the actual values are not continuous. These design choices are selected since the idea of the case study is to give insight into the behavior of the predictions in a clear manner and not an error metric comparison. The solid lines are predictions from the models and the plus sign depicts the current ego vehicle position for when the trajectory predictions are performed. It can be noted that in Figures 6.6c-6.10c there are multiple solid lines for the predictions of the surrounding vehicles. These are the different predicted modes of the TOT-MC model. In scenes where the TOT-MC is certain about the outcome, the modes collapse into the same prediction. The mode converging scenarios are common in cases of straight highway driving presented in Figure 6.7c. On the other hand, when the TOT-MC is uncertain about the future, multiple different modes are predicted as in Figure 6.6c.

One important property regarding a non-deterministic model is that the suggested multi-modal predictions are connected to a reasonable future. Whether these multi-modal predictions are reasonable can be cumbersome to establish with error metrics since a suggested prediction may be far from the ground truth. One interesting case suggesting that the developed TOT-MC model achieves this property is presented in case 6.6c where the model suggests two plausible future lane changes. Another example of this multi-modal reasonable future prediction is also presented in Figure 6.8c.

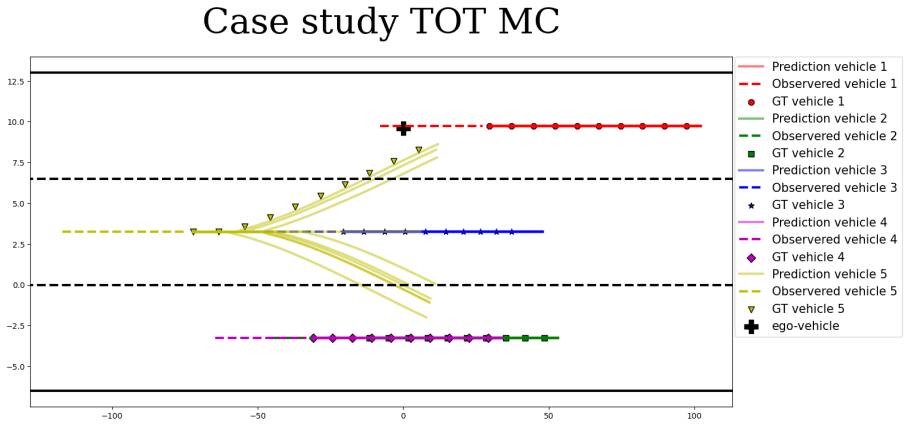
It is important to highlight that the developed models are not perfect and that there exist cases where both the TOT and the predicted different future trajectories of the TOT-MC do not capture the future evolution of the scene. One of these cases is presented in Figure 6.9b and 6.9c. In these uncertain scenarios, the TOT-MC expresses high uncertainty about the future evolution of the scene by predicting modes with significant differences. These modes may express high uncertainty in some cases, but the model can still miss expressing uncertainty in cases where it is not accurate such as in the predictions for the yellow vehicle in Figure 6.9c. It is important that the model is not always expressing high uncertainty in the rare lane-change scenarios if it is certain. One of these examples can be seen in the lane-changing maneuver scene in Figure 6.10, where the TOT-MC expresses high certainty with correct prediction compared to the GT future.



(a) CV

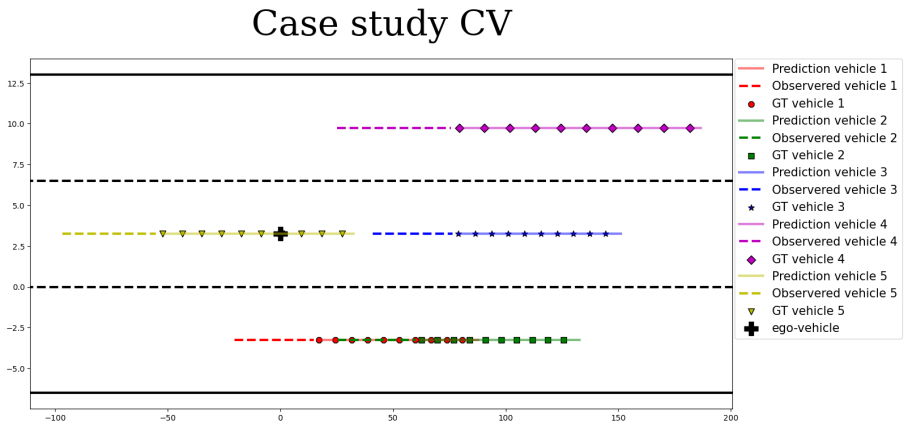


(b) TOT

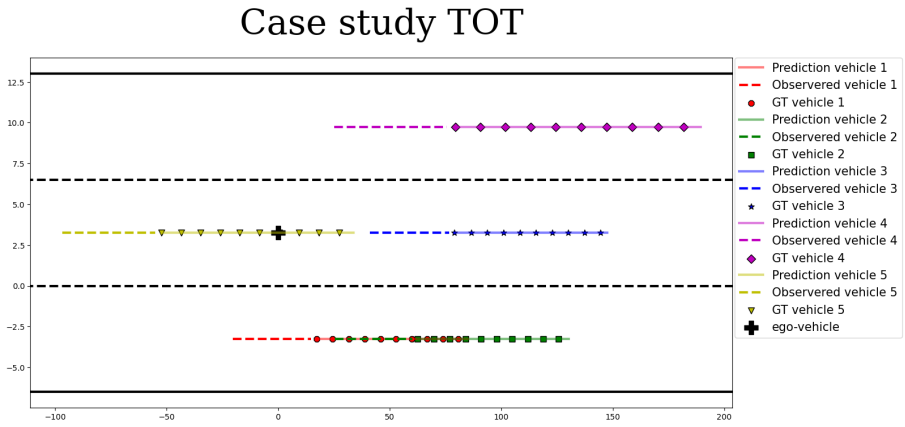


(c) TOT-MC

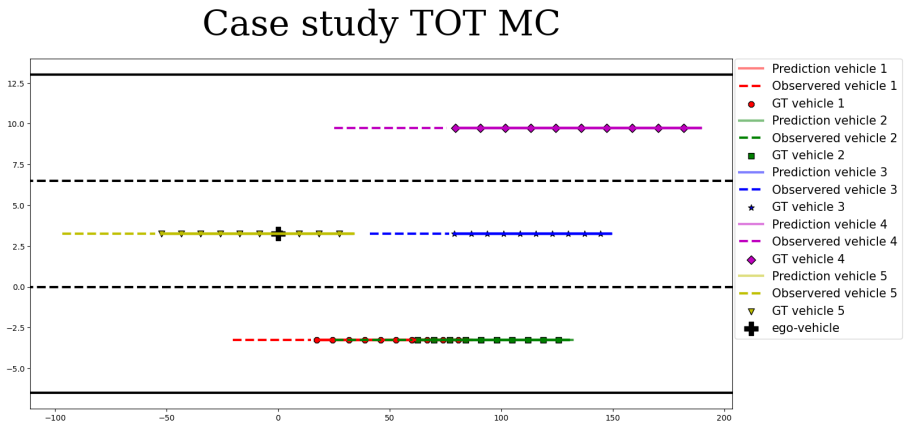
Figure 6.6: The scene displays one of the weaknesses of the CV model in terms of its inability to change prediction direction during a future state evolution. Interesting to highlight is the variations in the different modes predicted by the TOT-MC. Even though a left lane change is the correct GT future state evolution, a right lane change is still a reasonable trajectory to predict.



(a) CV

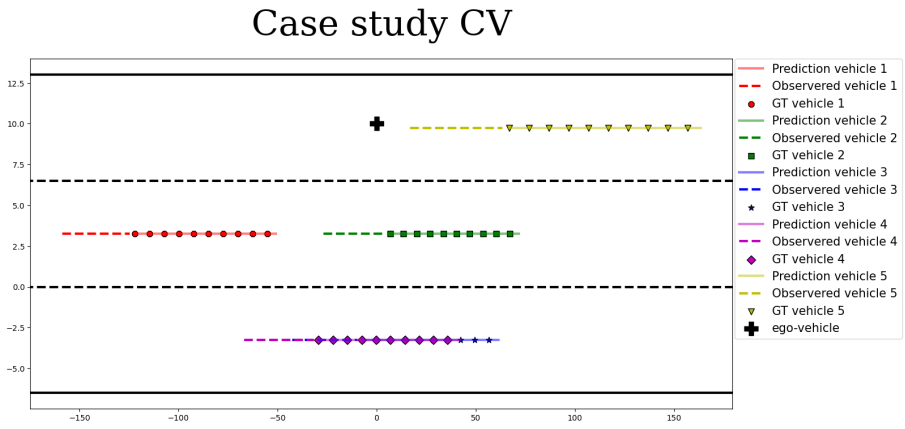


(b) TOT

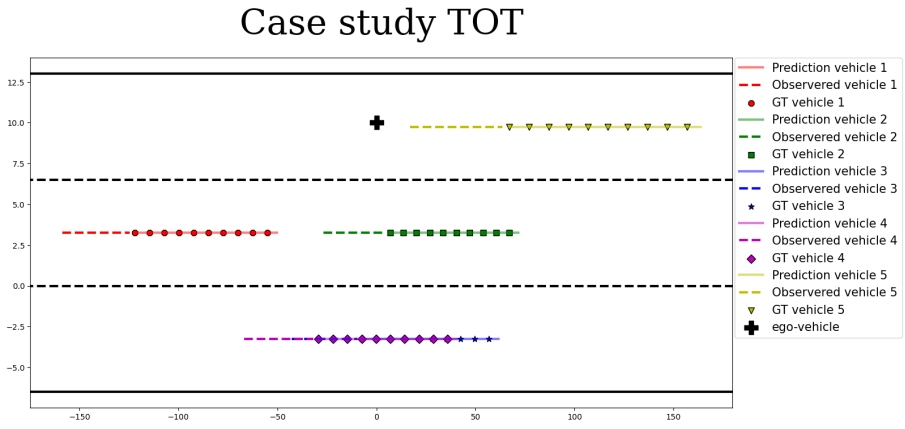


(c) TOT MC

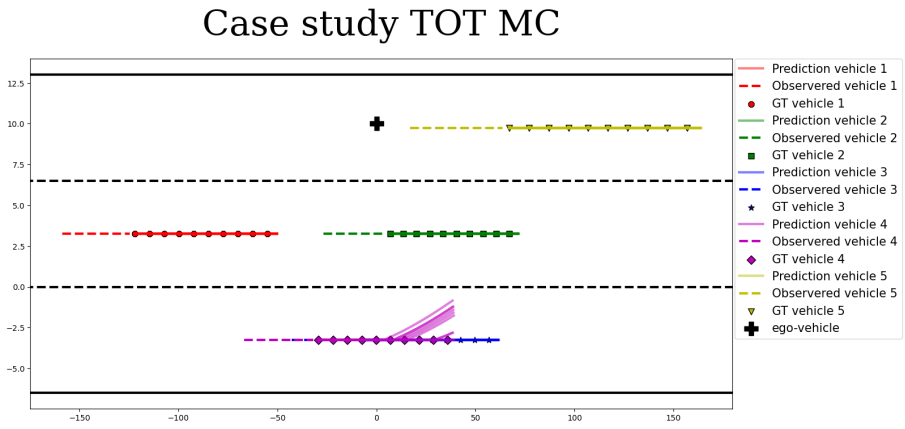
Figure 6.7: Straight driving is one of the most frequently occurring scenes on the highway. It is therefore important for the models to be accurate in these scenarios together with that the TOT-MC should not express uncertainty in these certain scenarios. In this case, all three models predict future positions that correspond to the ground truth.



(a) CV

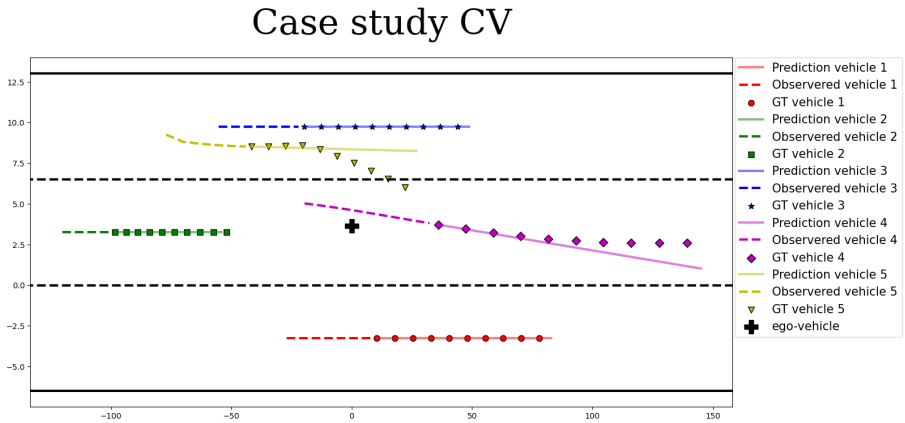


(b) TOT

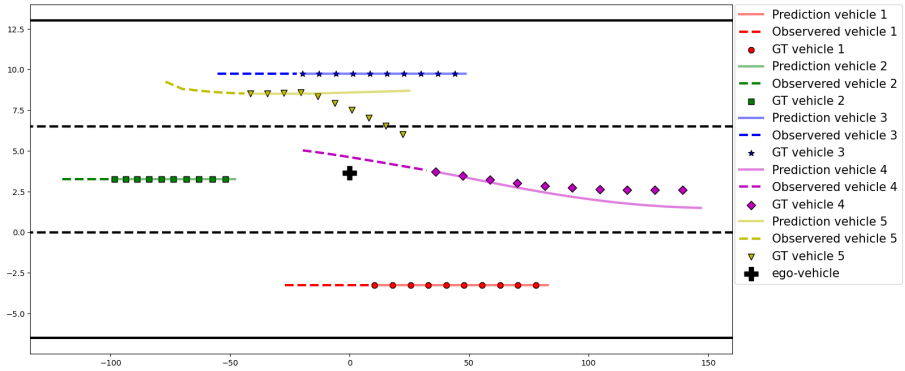


(c) TOT MC

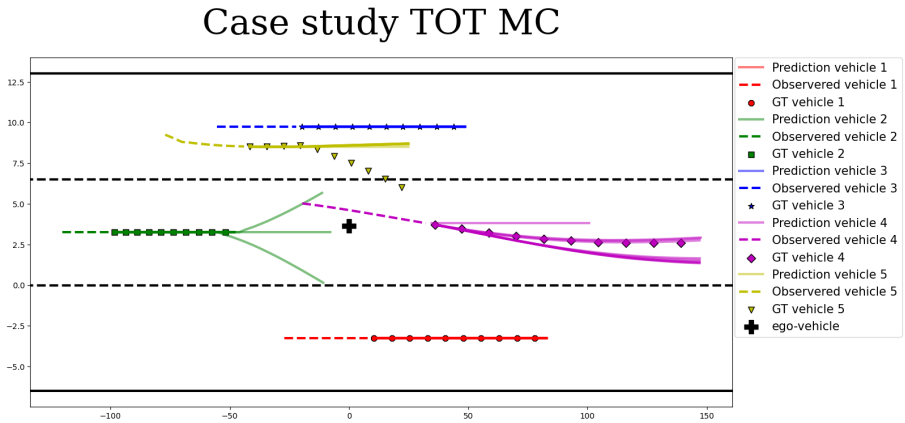
Figure 6.8: The scene displays a straight observed positional history with a GT future of straight driving for all vehicles. What is interesting to note is the suggestion from the TOT-MC regarding a potential lane change for some of the predicted modes. Even though this is incorrect compared to GT, the possibility of a lane change with a slower-moving vehicle in front is not an unreasonable trajectory to predict.



(a) CV

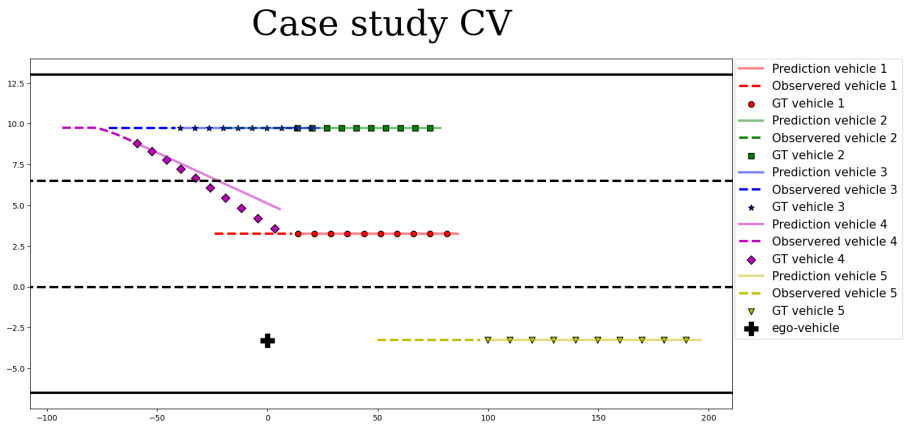


(b) TOT

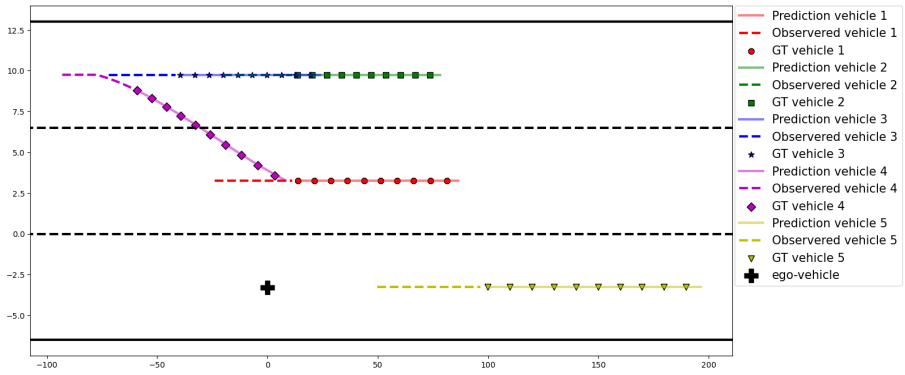


(c) TOT-MC

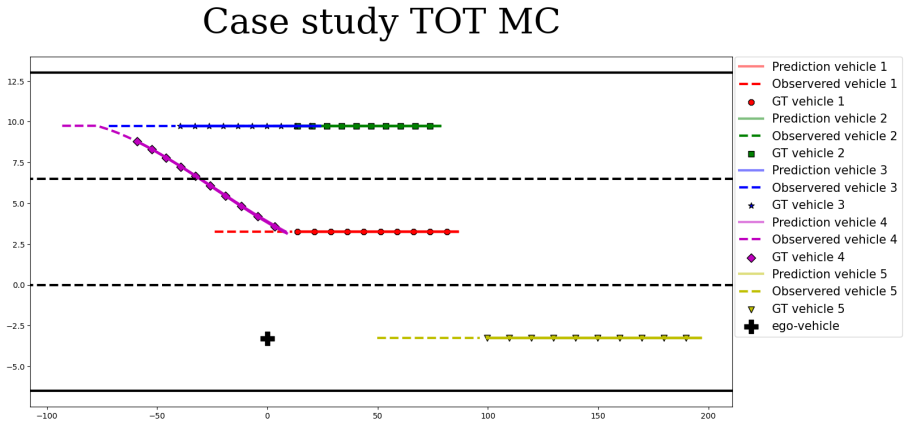
Figure 6.9: Important to remark is that the TOT-MC is not perfect and that there are cases where not even a single mode is capturing the correct GT future as for the yellow car in Figure c). In this case, neither the CV-model nor the TOT models accurately predict the ground truth.



(a) CV



(b) TOT



(c) TOT-MC

Figure 6.10: Even though lane changes are of low frequency, it is important to still be able to not always portray a measure of uncertainty in cases where the model is certain about the intentions of a surrounding vehicle. In this case, all modes collapse into one mode for the TOT-MC model, which corresponds to the ground truth. The TOT model also produces a prediction that corresponds to the ground truth while the prediction from the CV model does not correspond to the ground truth.

6.2 Observer and Model Predictive controller

The result for the three different observer and controller combinations are presented in two separate sections. Section 6.2.1 presents the results and analysis for the quantitative analysis, i.e. evaluation of running the different combinations in highway scenarios similar to what the observer was developed for. In section 6.2.2 the performance of the combinations in the enforced safety-critical situation is presented.

6.2.1 Quantitative analysis

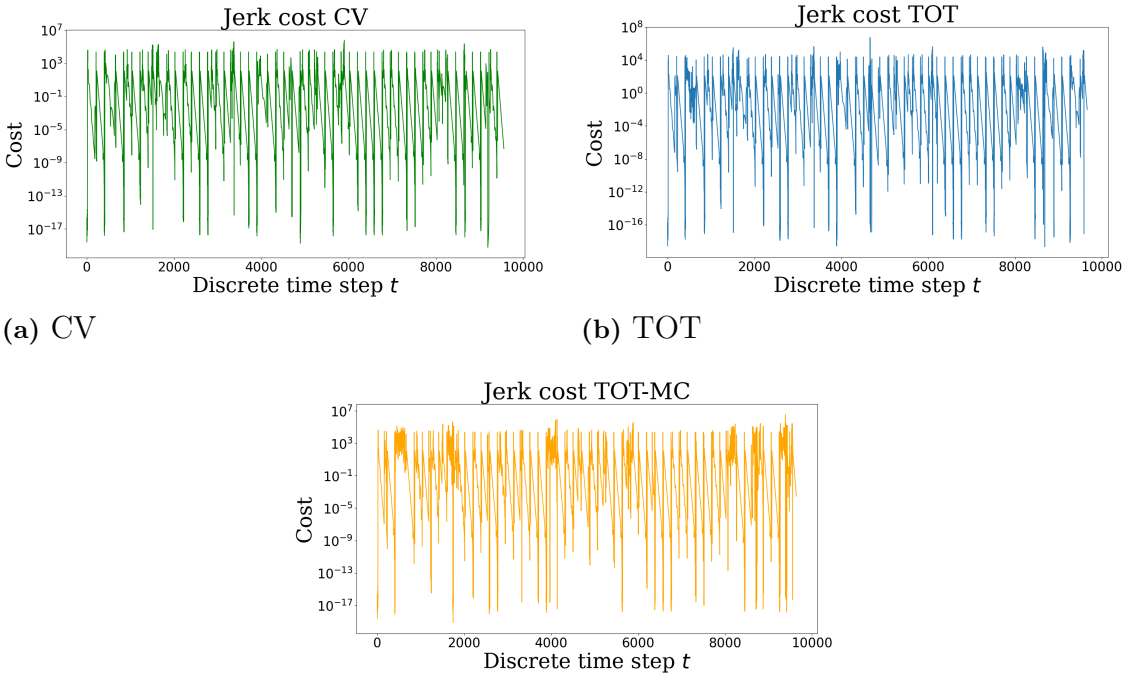
In Table 6.3 the results for the cost function evaluation are presented for the different controllers. In order to facilitate a more clear comparison all values in the table are normalized towards the total cost of the baseline CV model. The CV model combination has better performance from a total cost perspective compared to the TOT at 1.91 and TOT-MC at 1.43. It is important when doing cost function optimization to look at the total cost result for evaluation of best performance. However, it is interesting to identify how the different combinations solve the problem and what the main contributions to the resulting total cost are. The main contribution of why the TOT and TOT-MC models' total cost is higher than for the CV baseline is that they exhibit a significantly larger contribution of jerk. For the CV model, the jerk cost constitutes 54% of the total cost while for the TOT and TOT-MC this fraction is 76% and 68% respectively.

Table 6.3: Cost function evaluation that displays total cost and the separate costs state, control, jerk and slack cost for the three models. All values are normalized towards the baseline constant velocity total cost for the optimization over all time steps

Model	**Total cost**	**State**	**Control**	**Jerk**	**Slack**
CV	**1.0**	0.46	**7.98e-5**	**0.54**	2.93e-8
TOT	1.91	0.46	1.05e-4	1.45	**9.62e-25**
TOT-MC	1.43	0.46	8.42e-5	0.97	3.60e-24

In Figure 6.11 the jerk costs for all simulation time steps are presented as one long time series. There is a tendency for higher usage of jerk with higher frequency for the TOT and the TOT-MC models in Figures 6.11b and 6.11c. This behavior can be seen as reasonable since the predictions from the developed observers have a higher chance of changing their prediction for surrounding vehicles between time steps. One example of this is that in time step t the prediction for surrounding vehicles is to not change lane. Then in the next instance, $t + 1$, the models can predict a lane change for a vehicle in an adjacent lane to once again predict straight in time step $t + 2$. This change of predicted trajectory between time steps can then be overcompensated in the controller resulting in a jerk. It is a though balance from the perspective of safety to compensate with breaking which indicates jerk when lane changes may happen. Small compensations, even if they are correct or not, will

induce more jerk in the final result.



(a) CV

(b) TOT

(c) TOT-MC

Figure 6.11: Drivability analysis shows a more frequent use of jerk with the TOT-models compared to the CV-model. The values are plotted on a log-based scale.

In Figure 6.12 the result from Table 6.4 are further analyzed through the perspective of slack cost by a display of all values for all discrete time steps with a sampling time of 0.2 seconds during the simulations. From the perspective of total cost, slack only contributes with a small fraction of the cost. However, there is a behavior of significantly larger peaks of slack for the CV model compared to the TOT and TOT-MC models. As previously mentioned, the simulation scenarios evaluated in the quantitative study are common highway driving scenarios, i.e. no forced safety-critical situations and the expected results should be no slack. However, there is a clear behavior from the baseline CV model that peaks of slack are needed during the optimization. Consequences of this initial behavior of using slack become more apparent in the qualitative study in section 6.2.2 where a forced safety critical scenario result is analyzed.

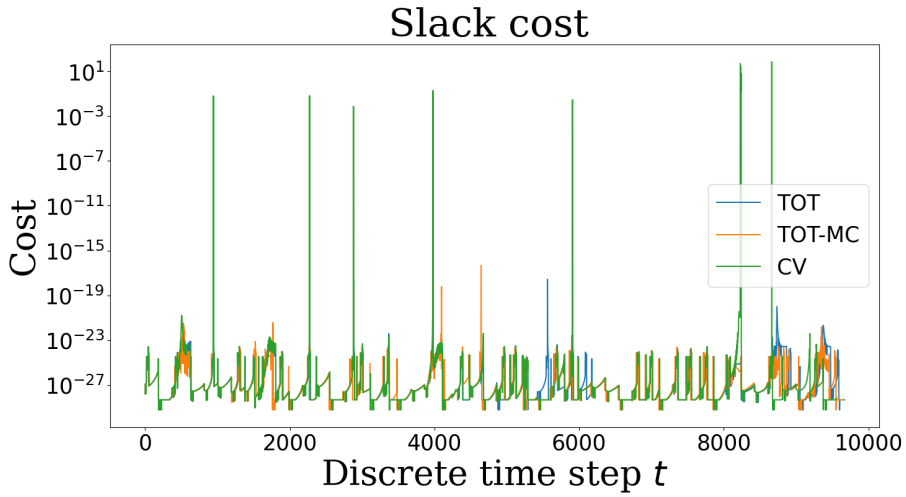


Figure 6.12: Slack cost evaluation shows that the constant velocity model is forced to slack the constraints in multiple simulations in order to obtain feasible solutions while the TOT-models do not. The costs are plotted on a log-based scale

6.2.2 Qualitative analysis

In Table 6.4 the total cost and individual cost components for the baseline CV and TOT-MC combinations are presented. The result of the qualitative study for the TOT combination was that it resulted in a collision for the specific scenario. An explanatory factor for this is that the TOT model is unable to express any uncertainty in its predictions which results in that the MPC algorithm is not compensating for uncertain scenarios. In contrast to the TOT, the TOT-MC model can compensate for uncertain scenarios and influence the controller's optimized trajectory, thus not resulting in a crash even though it may be uncertain about the predicted trajectories.

Initial analysis of Table 6.4 displays that the CV baseline combination performance is better from a total cost function perspective. Similar to the results in Table 6.3, the jerk cost for the TOT-MC is the main contribution towards the difference in total cost. However, it can be noted that this difference is slightly smaller compared to the quantitative results. Though smaller in magnitude, the main cost contribution for both components is from the jerk. This is an expected result since this is a safety-critical situation where heavy braking is to be expected.

Table 6.4: Case study cost function evaluation that displays total cost and the state, control, jerk and slack cost for the CV and the TOT-MC models. All values are normalized towards the baseline constant velocity total cost for the optimization over all time steps.

Model	**Total cost**	**State**	**Control**	**Jerk**	**Slack**
CV	**1.0**	**0.17**	1.00e-4	**0.82**	1.15e-3
TOT-MC	1.28	0.19	1.00e-4	1.09	**1.11e-29**

Even though the total cost of jerk is of smaller magnitude for the CV baseline model compared to the TOT-MC the realization of how and when this jerk cost arises differs significantly. The change of longitudinal acceleration and steering angle expressed in jerk cost for each time step in the simulation case for the CV and TOT-MC model is displayed in Figure 6.13. What is interesting to note is that the TOT-MC model starts to break around 3.6 seconds earlier than the CV model combination. The TOT-MC model applies several instances of breaking thus evoking more peaks of jerk cost. In contrast to this, the CV model applies one large breaking instance which manifests in one large peak of jerk cost. Even though the total cost for the CV combination is lower, it is interesting to remark that the usage of the control signal in the critical time step location for the peak of the jerk is 83% of the maximum possible deceleration. For the TOT-MC model, the deceleration is instead 56% of the hard constraint value for maximum possible deceleration.

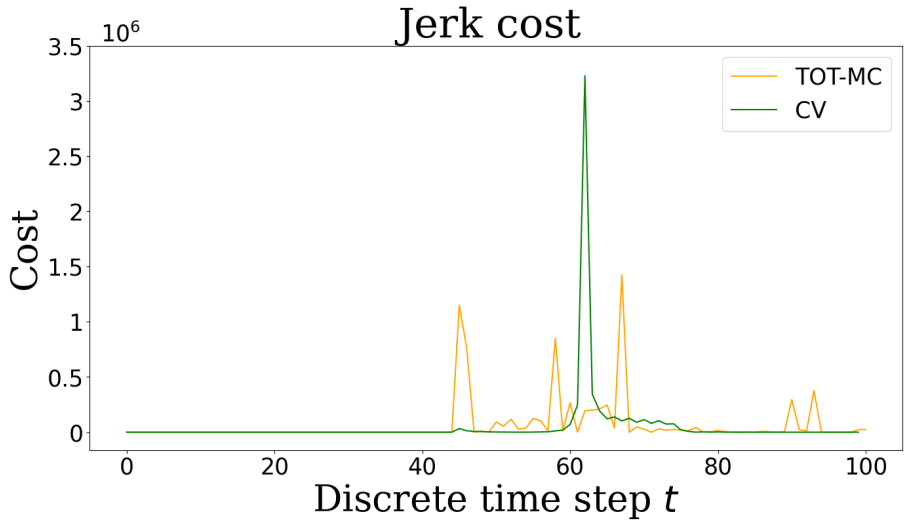


Figure 6.13: Jerk cost evaluation in the case study shows that using the constant velocity model as a predictor results in breaking later and harder, obtaining a higher peak in jerk compared to using the TOT-MC model.

In Figure 6.14 the slack cost from Table 6.4 is given for all discrete time steps t with a sampling time of 0.2 seconds of the simulation. Here the resulting action of breaking too late for the CV baseline model results in that it has to utilize slack, while the TOT-MC model does not. The need for using slack then clearly becomes apparent as a consequence of the lack of predictive capabilities in the baseline model.

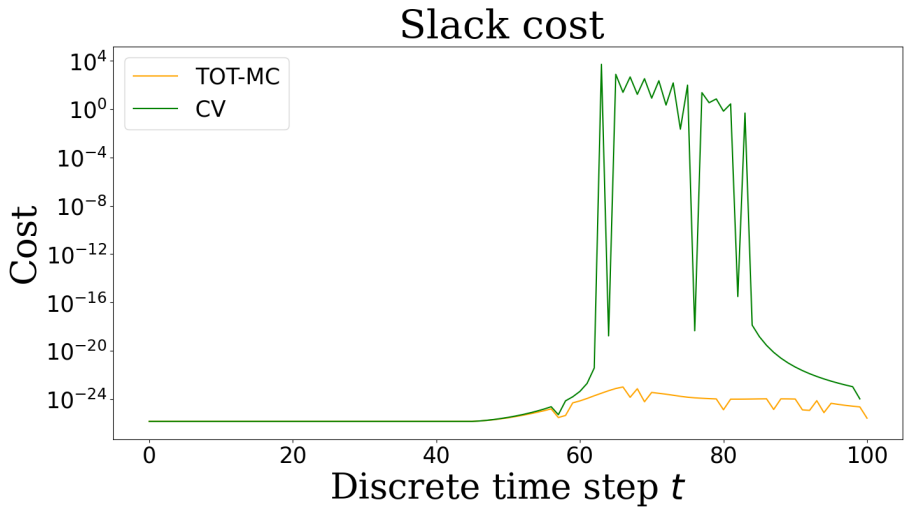


Figure 6.14: Slack cost evaluation in the case study shows that using the CV-model as a predictor results in the need to slack constraints in order to obtain feasible solutions, while no constraint violations are present while using the TOT-MC model as a predictor. The costs are plotted on a log-based scale.

7

Discussion

In this chapter, critical discussions about the used methodology and the results obtained in this thesis are held. A discussion about the data generation method is raised in section 7.1, followed by a discussion about using a transformer base for trajectory predictions in section 7.2 and the selected method for uncertainty modeling in the robust model predictive controller in section 7.3. The result discussion mainly concerns how drivability and safety are balanced against each other and is treated in section 7.4. The chapter concludes with possible future work where opportunities for both the predictor and the planner are outlined in section 7.5

7.1 Consequences of data generation method

The selected methodology for model learning in this project has been to construct a supervised learning model. One aspect of supervised learning and in general for machine learning development is the question if the data is "good". Clearly, the data can be regarded as "good" in this project since it contains accurate measurements due to the simulation environment. However, it is interesting to lift the issue of "good/bad" data from the perspective of bias toward the baseline model.

The aim of the observer is to learn from the previous behavior of surrounding vehicles given historical scenes to produce a predicted future. It becomes apparent that the history of the scene has always to some degree been influenced by the ego vehicle's previous trajectory and intentions. This relation of influence for observed behavior follows into the future positions which are directly influenced by how the ego vehicle will plan its future trajectory and intentions. Clearly, it becomes impossible for the selected data generation method to generate data that is free from influence from the baseline model.

The problem becomes apparent when the combination of the developed observer and MPC is to be tested during new simulation scenarios. Are then the predicted trajectories of surrounding vehicles based on that the ego vehicle will act in a learned way? One could approach the problem through a different methodology which is to learn the surrounding vehicles' behavior without the influence of an ego vehicle. This could be achieved by removing the ego vehicle from the scene and observe the surrounding traffic. This would remove the influence of bias but may reduce the ego vehicle perspective too much. Removing the ego vehicle from the scene will put limitations on how surrounding vehicles will act conditioned on the specific ego

vehicle. Surrounding traffic will most likely act differently if there is a truck in the adjacent lane intending to do a lane change compared to a normal passenger car. This then motivates that an ego vehicle present data generation method may be preferable.

The selected data generation method could also result in a reduction of comparative evaluation to other algorithms. Preferably, trajectory prediction algorithms are trained and evaluated on datasets that are publicly available for usage and commonly used for evaluation. In this thesis, a self-made dataset has been used for both training and evaluation. This reduces the possibilities of comparisons to other algorithms, but there is always the possibility to implement and train other algorithms on our dataset. However, this approach may not be considered optimal since adaptations of other algorithms towards a self-made dataset is not always a plug-and-play operation. With this in mind, it is important to lift the aim of the thesis which is to conduct research and investigate the future potential in this relatively unexplored area of supervised machine learning and MPC combination which the method still provides.

7.2 Transformer for trajectory predictions

The developed observer was constructed with a quantized state space for both the embedding of positions and velocities surrounding the ego vehicle. In combination with this, the selected loss function during the training of the model is a cross-entropy loss function. Following this methodology it is important to highlight the fact that a model developed in this way can never be better than its quantized state space. Further breaking down this statement with an example is that even though the model performs optimal in terms of classification toward the generated quantized state space it can still never be better than the quantized state space. This problem was identified early in the selected method for the construction of the transformer-based model via an increase of resolution for the state space quantization. This increase in resolution gave better results for the predictions since it could predict more accurately. One problem with increasing resolution is that the classification problem can become more difficult to solve. However, transformer models are developed to be able to handle a large number of classification categories in NLP tasks thus making it possible to use a high resolution for the quantized state-space.

There exist other methods than the quantized state space classification approach presented in this thesis. One of these approaches is to reformulate the problem into a regression-based prediction strategy. Using a regression-based predictor facilitates a continuous prediction strategy and the problem of not being able to be better than the quantization can be removed. A regression strategy was however examined during the project together with a transformer-based structure without success. Why adopting a regression-based prediction strategy was unsuccessful could simply be because the field hasn't been fully explored or the transformer isn't originally designed for regression problems. Forcing a model towards something it wasn't originally intended for could be a difficult process and there is definitely an area of future interest for continued investigations.

Even though the transformer utilizes a classification-based approach to the problem the results show promising performance compared to the baseline model. Deep machine learning development is a process that constitutes of a lot of trial and error with creativity being a large part of the process to find what is functional. This thesis has presented one novel strategy for how the prediction problem can be approached. There may be other models or alterations of the developed models that achieve more promising results. The suggested methodology is meant to act as a base for further investigations and motivations toward a continued investigation into applying transformers for the trajectory prediction problem.

7.3 Implemented uncertainty consideration in the robust model predictive controller

As previously mentioned in this thesis, the MPC relies on accurate predictions of surrounding vehicle to form constraints that mitigate the chance of collisions. An extension in this project has also been to involve uncertainty in these predictions through applying a Monte Carlo dropout method. It becomes interesting to follow if the extensions of the constraints is representable for the differently predicted trajectories. The method to fuse these uncertainties from the observer with the trajectory planner has been to use a mode mean approach. The uncertainty is then expressed through a statistical fashion via standard deviations from the mean of the modes. One problem that can be identified here is the notion of representing different modes into one mode. The different modes showcase different properties and simply averaging them into one mode may be a somewhat questionable method. One example of this is when the TOT-MC model predicts a left lane change and a right lane change for a vehicle in front. Even though the model is predicting left and right changing trajectories a mode mean would result in a mean of corresponding to straight driving. However, one strength of the aforementioned method is that in this case, it also provides a large uncertainty to the prediction, meaning that the MPC is subjected to a smaller set of feasible states.

Another interesting note to look at is the used scaling factor. The selected scaling parameter used was kept constant during the entire prediction horizon, meaning that the MPC fully trusts the uncertainties equally the observer displays at every time step. This is however up to debate whether scaling the constraint in this way is accurate. One could argue that the constraints should not be extended equally as much for the uncertainty further away in the prediction horizon. The reason for this is that, even if the MPC obtains a faulty trajectory prediction far away into the prediction horizon, it would still be able to update its own trajectory in one of the next time steps, avoiding a potential collision. Since this is not the case for prediction at the beginning of the prediction horizon, reasonably, the MPC should heavily trust the uncertainty. This could in turn lead to a less conservative MPC which may obtain less jerk. However, it is also important to consider the weight of safety compared to a lesser jerk which will be discussed in the following section.

7.4 Drivability vs Safety

In terms of combining a transformer-based observer with a robust MPC, compared to the use of a constant velocity model, the outcome varies. In normal driving scenarios, i.e no heavy breaking or acceleration on a highway, the constant velocity in combination with a MPC performs better than the developed models in terms of drivability and total cost. The main cost component apparent in the total cost is the higher jerk cost while using the TOT model as a predictor. The reason for this discrepancy comes from the variations in the TOT-model predictions between time steps, while the CV-model predictions are smoother. Notably, it is interesting that a simple model like a CV model can perform better as a predictor for the developed MPC than a complex transformer-based observer. Although, as previously mentioned, during normal highway truck driving surrounding vehicles most likely keep their speed, hence CV is a very accurate baseline model and difficult to compete with. Observing the constraint violation in normal driving scenarios indicates a small improvement using the TOT and TOT-MC models, however, the cost of slack using the CV model is still marginal. The need for a robust MPC for a highway autopilot can thus be questioned in normal driving scenarios. This comes from the fact of the results indicating that the natural robustness using the baseline MPC might be sufficient for safe operation.

On the other hand, in safety-critical situations, the need for an observer that can capture interactions becomes apparent. Results show that a simple observer, like the CV model, fails to provide predictions that are good enough to keep the ego vehicle from violating the constraints. On the contrary, the TOT-MC provides predictions with uncertainty that allows for the controller to take action earlier, ensuring safe operation. Even if the TOT-MC does not provide an accurate prediction at a certain time step, including uncertainties in the predictions allows for a more conservative trajectory plan, which is preferable in these kinds of situations. One interesting factor to look at in the evaluated safety-critical situation using the CV model is that breaking is almost of the max amplitude of what is allowed by the constraints while using the TOT-MC model nearly halves that value. Not only might the lower level of braking lead to an improvement in the comfort for the driver in the truck, but it might also divert a potential crash if the truck was loaded heavier. This comes from the possibility to break earlier using the TOT-MC model, which would have been needed if the breaking capacity of the truck would be impaired.

Noted in the discussion above, the concept of safe operation does not always have a positive correlation to drivability, rather the opposite is true. Utilizing a complex transformer-based observer including uncertainty that has the possibility to capture driving interactions leads to less constraint violation. Although, this comes at the cost of persevering more jerk in the optimized solution compared to using a CV model. This behavior is similar to that obtained in [3], where safety came at the cost of higher maximum acceleration. In a perfect situation, the predictions from the TOT-model would correspond to the ground truth at every time instance in the prediction horizon. However, considering the stochastic nature of human behavior, strengthened by the results in this thesis, multiple outcomes will be viable in different

situations. Thus, the manner of altering predictions between time steps is hard to get rid of. Ultimately, there is a balance between obtaining an autopilot that is able to take safe decisions while still maintaining advantageous drivability.

7.5 Future Work

Two main areas have been recognized in the thesis as the next step for the robust controller to further conduct research within. What differentiates them is mainly that one focuses on the next step in the observer development, in section 7.5.1, and an identified development area in the trailing controller of the MPC in section 7.5.2.

7.5.1 Extensions towards the next interaction aware trajectory planner

What is accompanied in deep machine learning development is the question of: can the model be further improved? Initial approaches are to continue with increasing the amount of training data and more variations of training data, i.e. more scenarios that the model has not previously experienced. Other ways of improvement can be further tuning of hyperparameters or even an extension of the number of parameters in the model itself. However, even if all of these suggestions are interesting to apply there is one future work that should be emphasized: a transition from a non-interaction-aware observer to an interaction-aware observer.

What is referenced here is the involvement of interaction awareness in the model, simply put as "If I intend to do this what will you do?". This next step could be achieved by the addition of the planned ego vehicle trajectory into the observer input. One idea can be to use the planned trajectory from the previous time step and consider it in the observer for future predictions. The idea behind this is that interaction-awareness could play a crucial role in reducing jerk since different intentions from the ego vehicle could potentially reduce the difference between the multi-modal predictions for surrounding vehicles. The observer currently only considers how other vehicles will act excluding its own planned trajectory. The hypothesis is that if the ego vehicle can involve its own intentions in the prediction the predictions can further be conditioned on the whole scene with the ego vehicle in mind.

7.5.2 Future development for trailing controller to fully utilize trajectory predictions

Presented in the description of this section is the statement of being able to utilize the predictions fully. This statement is mainly based on the current trailing controller which does not consider a change of lead vehicle during the optimization horizon. The current trailing controller only considers one lead vehicle at the current time step. The problem with this formulation is that if an adjacent vehicle is to change lanes to a position of becoming a new lead vehicle during the optimization horizon, it will not be considered. A vehicle will therefore only be considered a lead vehicle, which the trailing controller considers, if it is in the same lane as the ego vehicle

in the current time step. Therefore, it becomes apparent that a scenario where a lane-changing maneuver prediction by an observer into a new lead vehicle during an optimization horizon might lead to problematic situations.

Important to note is that a predicted lane-changing maneuver into a new lead vehicle in front of the ego vehicle by an adjacent vehicle is always considered in the left and right lane change controller. Thus, a scenario could play out that a trailing decision based on a lead vehicle, which is to change during the horizon but not considered by the trailing controller, may result in a lower cost than a lane-changing decision. This lower cost for trailing could therefore give a bias towards continued trailing, even if perhaps the best decision would actually be to perform a lane change. Clearly, there are areas of improvement for fully utilizing prediction in the autopilot algorithm that could further enhance performance in safety-critical situations.

8

Conclusion

The purpose of the thesis was to investigate and research the combinations of a supervised learning algorithm and a model predictive controller. This has been performed by the development of a transformer-based observer for generation of trajectory predictions of surrounding vehicles with uncertainty estimation. The predicted trajectories together with uncertainties have been combined with the ego vehicle's MPC for increased robustness. The identified research questions from the aim of the thesis read:

- Is the developed observer able to capture the behavior of surrounding vehicles in a highway scenario and to what extent?
- How can a learning-based observer and MPC be combined to increase robustness in a trajectory planning algorithm for an autonomous truck in a highway scenario?
- What are the main benefits that can be gained by using the aforementioned combination from the perspective of drivability and safety?

From the perspective of capturing behaviour of surrounding vehicles in the trajectory prediction it can be concluded that the observer is capable of producing accurate predictions for the highway scenario. Furthermore, it is also able to produce predictions to the extent of reasonable suggestions in a multi-modal fashion of future surrounding vehicle trajectories to enhance safe trajectory planning.

The combination of the observer and MPC was performed through prediction uncertainty estimation involvement in the constraint formulation of the trajectory planner. Robustness in this combination was proven to be successful, especially in a safety-critical situation where predictions combined with uncertainty estimation produced a more safe planning.

From a drivability perspective measured in jerk the developed observer and trajectory planning algorithm produced a worse result than the baseline autopilot. However, this result was to be expected due natural properties of predicting for safe trajectories with implied compensation. The main benefits comes from the fact that the autopilot now can predict a safer trajectory by considering multi-modal prediction of surrounding vehicles and avoid dangerous situations.

It can be concluded that the combination of advanced machine learning algorithms for situation analysis together with an MPC can produce properties for extending the robustness of a highway autopilot. Further research will need to be produced before

inclusion into real-world scenarios, but the results presented in this thesis provide a good foundation for further exploration in the field of MPC in combination with a transformer-based observer.

Bibliography

[1] Geneva: World Health Organization, "Global status report on road safety 2018," 2018.

[2] Volvo Group, *Volvo Trucks Safety Report 2017*, 2017.

[3] S. Bae, D. Isele, A. Nakhaei, P. Xu, A. M. Añon, C. Choi, K. Fujimura, and S. Moura, "Lane-change in dense traffic with model predictive control and neural networks," *IEEE Transactions on Control Systems Technology*, vol. 31, no. 2, pp. 646–659, 2023.

[4] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social gan: Socially acceptable trajectories with generative adversarial networks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2255–2264, 2018.

[5] M. Everett, Y. F. Chen, and J. P. How, "Collision avoidance in pedestrian-rich environments with deep reinforcement learning," *IEEE Access*, vol. 9, pp. 10357–10377, 2021.

[6] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[7] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in *NIPS 2014 Workshop on Deep Learning, December 2014*, 2014.

[8] N. Nikhil and B. Tran Morris, "Convolutional neural network for trajectory prediction," in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pp. 0–0, 2018.

[9] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE transactions on neural networks*, vol. 20, no. 1, pp. 61–80, 2008.

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[11] C.-J. E. Hoel, *Decision-Making in Autonomous Driving Using Reinforcement Learning*. PhD thesis, Chalmers Tekniska Hogskola (Sweden), 2021.

[12] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd,

R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic traffic simulation using sumo," in *The 21st IEEE International Conference on Intelligent Transportation Systems*, IEEE, 2018.

[13] E. Börve, "Autonomous truck sim." `https://github.com/BorveErik/Autonomous-Truck-Sim`, 2023.

[14] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical review E*, vol. 62, no. 2, p. 1805, 2000.

[15] A. Kesting, M. Treiber, and D. Helbing, "General lane-changing model mobil for car-following models," *Transportation Research Record*, vol. 1999, no. 1, pp. 86–94, 2007.

[16] D. D. Salvucci and R. Gray, "A two-point visual control model of steering," *Perception*, vol. 33, no. 10, pp. 1233–1248, 2004.

[17] D. Q. Mayne, "Model predictive control: Recent developments and future promise," *Automatica*, vol. 50, no. 12, pp. 2967–2986, 2014.

[18] M. Morari and J. H. Lee, "Model predictive control: past, present and future," *Computers & Chemical Engineering*, vol. 23, no. 4-5, pp. 667–682, 1999.

[19] C. E. Beal and J. C. Gerdes, "Model predictive control for vehicle stabilization at the limits of handling," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 4, pp. 1258–1269, 2013.

[20] A. Carvalho, Y. Gao, A. Gray, H. E. Tseng, and F. Borrelli, "Predictive control of an autonomous ground vehicle using an iterative linearization approach," in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, pp. 2335–2340, 2013.

[21] G. de Nicolao, L. Magni, and R. Scattolini, "On the robustness of receding-horizon control with terminal constraints," *IEEE Transactions on Automatic Control*, vol. 41, no. 3, pp. 451–453, 1996.

[22] J. Karlsson, N. Murgovski, and J. Sjöberg, "Optimal trajectory planning and decision making in lane change maneuvers near a highway exit," in *2019 18th European Control Conference (ECC)*, pp. 3254–3260, 2019.

[23] N. Murgovski and J. Sjöberg, "Predictive cruise control with autonomous overtaking," in *2015 54th IEEE Conference on Decision and Control (CDC)*, pp. 644–649, 2015.

[24] J. Rawlings, D. Mayne, and M. Diehl, *Model Predictive Control: Theory, Computation, and Design 2nd edition*. Nob Hill Publishing, 2017.

[25] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "Casadi: a software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, pp. 1–36, 2019.

[26] P. J. Campo and M. Morari, "Robust model predictive control," in *1987 American control conference*, pp. 1021–1026, IEEE, 1987.

[27] D. M. Raimondo, D. Limon, M. Lazar, L. Magni, and E. F. ndez Camacho, "Min-max model predictive control of nonlinear systems: A unifying overview on stability," *European Journal of Control*, vol. 15, no. 1, pp. 5–21, 2009.

[28] T. Brüdigam, F. D. Luzio, L. Pallottino, D. Wollherr, and M. Leibold, "Grid-based stochastic model predictive control for trajectory planning in uncertain environments," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1–8, 2020.

[29] T. Brüdigam, M. Olbrich, D. Wollherr, and M. Leibold, "Stochastic model predictive control with a safety guarantee for automated driving," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 1, pp. 22–36, 2023.

[30] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[31] Q. Wen, T. Zhou, C. Zhang, W. Chen, Z. Ma, J. Yan, and L. Sun, "Transformers in time series: A survey," *32nd International Joint Conference on Artificial Intelligence (IJCAI 2023)*, 2023.

[32] R. Korbmacher and A. Tordeux, "Review of pedestrian trajectory prediction methods: Comparing deep learning and knowledge-based approaches," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 12, pp. 24126–24144, 2022.

[33] F. Altché and A. de La Fortelle, "An lstm network for highway trajectory prediction," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 353–359, 2017.

[34] B. Kim, C. M. Kang, J. Kim, S. H. Lee, C. C. Chung, and J. W. Choi, "Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 399–404, 2017.

[35] L. R. Medsker and L. Jain, "Recurrent neural networks," *Design and Applications*, vol. 5, pp. 64–67, 2001.

[36] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr, "Conditional random fields as recurrent neural networks," in *Proceedings of the IEEE International conference on computer vision*, pp. 1529–1537, 2015.

[37] K. Antoniadou-Plytaria, L. Eriksson, J. Johansson, R. Johnsson, L. Kötz, J. Lamm, E. Lundblad, D. Steen, L. A. Tuan, and O. Carlson, "Effect of short-term and high-resolution load forecasting errors on microgrid operation costs," in *2022 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, pp. 1–5, 2022.

[38] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 961–971, 2016.

[39] V. Kosaraju, A. Sadeghian, R. Martín-Martín, I. Reid, H. Rezatofighi, and S. Savarese, "Social-bigat: Multimodal trajectory forecasting using bicycle-gan and graph attention networks," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[40] E. Jo, M. Sunwoo, and M. Lee, "Vehicle trajectory prediction using hierarchical graph neural network for considering interaction among multimodal maneuvers," *Sensors*, vol. 21, no. 16, p. 5354, 2021.

[41] F. Giuliari, I. Hasan, M. Cristani, and F. Galasso, "Transformer networks for trajectory forecasting," in *2020 25th International conference on pattern recognition (ICPR)*, pp. 10335–10342, IEEE, 2021.

[42] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, (Minneapolis, Minnesota), pp. 4171–4186, Association for Computational Linguistics, June 2019.

[43] A. Sadeghian, V. Kosaraju, A. Gupta, S. Savarese, and A. Alahi, "Trajnet: Towards a benchmark for human trajectory prediction," *arXiv preprint*, 2018.

[44] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, "Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data," 2021.

[45] A. Quintanar, D. Fernández-Llorca, I. Parra, R. Izquierdo, and M. A. Sotelo, "Predicting vehicles trajectories in urban scenarios with transformer networks and augmented information," in *2021 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1051–1056, 2021.

[46] E. Amirloo, A. Rasouli, P. Lakner, M. Rohani, and J. Luo, "Latentformer: Multi-agent transformer-based interaction modeling and trajectory prediction," 2022.

[47] Y. Yuan, X. Weng, Y. Ou, and K. Kitani, "Agentformer: Agent-aware transformers for socio-temporal multi-agent forecasting," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9793–9803, 2021.

[48] Y. Liu, J. Zhang, L. Fang, Q. Jiang, and B. Zhou, "Multimodal motion prediction with stacked transformers," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7573–7582, 2021.

[49] M.-F. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, and J. Hays, "Argoverse: 3d tracking and forecasting with rich maps," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8740–8749, 2019.

[50] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal, "Explaining explanations: An overview of interpretability of machine learning," in *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, pp. 80–89, 2018.

[51] E. Hüllermeier and W. Waegeman, "Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods," *Machine Learning*, vol. 110, pp. 457–506, 2021.

[52] S. Shafaei, S. Kugele, M. H. Osman, and A. Knoll, "Uncertainty in machine learning: A safety perspective on autonomous driving," in *Computer Safety, Reliability, and Security: SAFECOMP 2018 Workshops, ASSURE, DECSoS, SASSUR, STRIVE, and WAISE, Västerås, Sweden, September 18, 2018, Proceedings 37*, pp. 458–464, Springer, 2018.

[53] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural network," in *International conference on machine learning*, pp. 1613–1622, PMLR, 2015.

[54] Y. Gal, *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, 2016.

[55] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *International conference on machine learning*, pp. 1050–1059, PMLR, 2016.

[56] S. Wager, S. Wang, and P. S. Liang, "Dropout training as adaptive regularization," *Advances in neural information processing systems*, vol. 26, 2013.

[57] J. Ngiam, V. Vasudevan, B. Caine, Z. Zhang, H.-T. L. Chiang, J. Ling, R. Roelofs, A. Bewley, C. Liu, A. Venugopal, *et al.*, "Scene transformer: A unified architecture for predicting future trajectories of multiple agents," in *International Conference on Learning Representations*, 2022.

[58] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.

[59] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Computation*, vol. 1, no. 2, pp. 270–280, 1989.

[60] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference for Learning Representations*, 2015.

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY