Trajectory planning and following in urban environments

To reduce traffic accidents involving vulnerable road users

MSc Thesis

V.V.J. van der Drift

Delft University of Technology (TU Delft)

TU Delft

Trajectory planning and following in urban environments

To reduce traffic accidents involving vulnerable road users

by

V.V.J. van der Drift

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Tuesday July 18, 2023 at 01:30 PM.

Student number:	4482646	
Project duration:	September 13, 2022 – July 18, 2023	
Thesis committee:	Dr. L. Ferranti,	TU Delft, supervisor
	Ir. O. M. de Groot,	TU Delft, daily supervisor
	Dr. B. Shyrokau,	TU Delft

Cover: Generated by the author with Midjourney using prompt: "*cartoon style poster of autonomous modern futuristic car in busy urban environment at a crossing avoiding pedestrians and cyclists –ar 2:3*"

Abstract

This paper presents a comprehensive approach to integrating a trajectory planner and follower for autonomous vehicles (AVs) using model predictive contouring control (MPCC). The planner generates collision-free trajectories with a kinematic bicycle model, while the follower tracks them using a dynamic bicycle model with a smaller integration step size and higher update frequency. The hierarchical architecture allows for a long planning horizon and a fast control loop. Mismatches between the planner and follower can result in tracking errors and conservative trajectories. To address this, a feedback-hierarchical interface is proposed, feeding back the mismatch error from follower to planner. Obstacles are then inflated with this error, minimizing harmful deviations and collision risks. The paper validates the Local Motion Planner using a simulator with a dynamic bicycle model, testing different follower-solver settings in urban scenarios. The results show a reduction in collision rate of 23% compared to a single-layer MPCC planner, with similar levels of lateral error and task duration.

Acknowledgements

I would like to thank Oscar de Groot, my daily supervisor, for his guidance, help, advice, and patience during this project. He always took the time to take a good look at my problems and solutions, knew to ask the right questions, and was available to help when I got stuck. I could not have gotten this far without his support.

Also, I would like to thank Dr. Laura Ferranti for her support and feedback.

Last but not least, I would like to thank my parents, family, and friends for their support throughout my life as a student.

Rotterdam, 9 July 2023 Victor Vijay Jivan van der Drift

Contents

Nomenclature

Abbreviations

Abbreviation	Definition
3mE	Mechanical, Maritime and Materials Engineering
ABS	Anti-lock Braking Systems
ACC	Adaptive Cruise Control
ADAS	Advanced Driver-Assistance Systems
AEB	Autonomous Emergency Braking
AV	Autonomous Vehicle
CoR	Cognitive Robotics
ESC	Electronic Stability Control
LMP	Local Motion Planner
LQR	Linear Quadratic Regulator
MPC	Model Predictive Control
MPCC	Model Predictive Contouring Control
PID	Proportional Integral Derivative
R2CLab	Reliable Robot Control Lab
VRU	Vulnerable Road User

Symbols

Symbol	Definition	Unit
a_x	Longitudinal acceleration	[m/s^2]
$C_{\alpha f}$	Front lateral slip stiffness	[N/rad]
$C_{\alpha r}$	Rear lateral slip stiffness	[N/rad]
F_{lat}	Lateral tire forces	[N]
I_z	Moment of inertia	[kg m^2]
J	Objective function	[-]
K	Control gain	[-]
l_f	Distance from center of gravity to front axle	[m]
l_r	Distance from center of gravity to rear axle	[m]
m	Mass	[kg]
r_z	Yaw velocity	[rad/s]
v_x	Longitudinal velocity	[m/s]
v_y	Lateral velocity	[m/s]
X	X position	[m]
Y	Y position	[m]
α_f	Front slip angle	[rad]
α_r	Rear slip angle	[rad]
β	Side slip angle	[rad]
δ	Road wheel angle	[rad]
$\hat{\epsilon}^c$	Contouring error	[m]
Θ_{Vf}	Front velocity vector angle	[rad]
Θ_{Vr}	Rear velocity vector angle	[rad]
Ψ	Orientation	[rad]

List of Figures

1 Introduction

Between 20 and 50 million individuals are involved in traffic accidents annually [1]. Of these millions of people, about 1.3 million are the victims of fatal accidents. Road traffic injuries are the leading cause of death for children and young adults aged 5-29 years. Most of these accidents are the result of mistakes made by human errors. Driver error and irresponsibility may play a far smaller role in vehicle crashes if drivers were to use autonomous vehicles (AVs). Additionally, AVs will give a method of personal mobility for those who are physically or visually impaired and unable to drive [2]. Last but not least, autonomous cars would enable more efficient use of travel time, freeing up valuable hours for work or leisure and lessening the quantifiable negative consequences of driving stress.

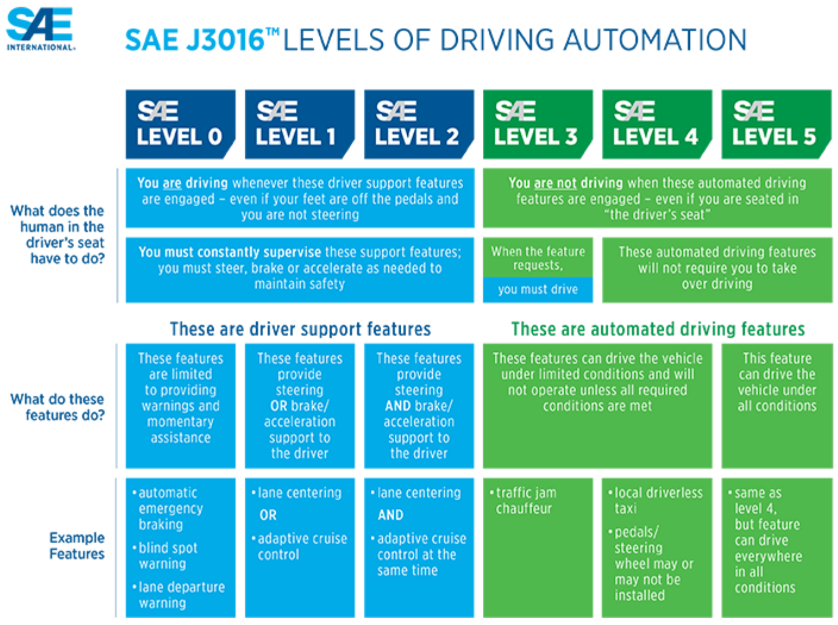
1.1. Advanced Driver-Assistance Systems (ADAS)

Self-driving automobiles have a lengthy history, which is not unexpected given the potential effects of this new technology. Although the concept has been around since the 1920s, autonomous automobiles did not appear to be a realistic option until the 1980s. The inaugural DARPA Grand Challenge, which took place in 2004, was the next significant development in autonomous car technology. It was a driverless car competition that sparked much interest in autonomous vehicles. Since then, several activities and significant tests of autonomous vehicle systems have been conducted in industry, but the pace of research has risen simultaneously in both academic and industrial settings.

One of the key benefits of autonomous cars in relation to pedestrians is their potential to reduce accidents caused by human error, which is a leading cause of road accidents. Pedestrians are often at risk due to drivers' failure to yield, distracted driving, or impaired judgment. By eliminating these factors, autonomous cars can provide a safer environment for pedestrians, potentially reducing the number of accidents and fatalities. Furthermore, the advanced sensors and algorithms utilized by autonomous vehicles can enhance their ability to detect and respond to pedestrians, even in low-visibility conditions, further increasing safety.

Over the past three decades, research into creating autonomous car technology has significantly increased in both academia and industry. Recent breakthroughs in computer processing and sensor technologies, as well as the potential disruptive effects on automobile transportation and the anticipated social benefit of reducing accidents, have all fuelled these innovations.

Autonomous vehicles (AVs) possess benefits in the field of safety, accessibility, and environmental impact. These include the potential of AVs in reducing the number of (fatal) road accidents, mobility for people with lesser physical abilities required for driving manually, and reducing environmental impact mainly by driving more energy-efficiently, together with shared mobility (Mobility As A Service / MaaS). Advanced Driving Assistance Systems (ADAS) are already being deployed in new cars to aid with improving driver safety and automated vehicles will in the future utilise extensions of ADAS functionalities which are currently being researched.



SAE J3016™ LEVELS OF DRIVING AUTOMATION

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

Figure 1.1: Levels of automation according to the SAE J3016 Standard. [4]

1.2. Levels of Driving Automation

A car's level of automation can range from totally human-operated to fully autonomous. A scale from 0 to 5 is included in the SAE J3016 standard [3] for rating vehicle automation. According to this standard, a vehicle at level 0 has a human driver handling every aspect of driving, which means the car has no autonomous features. Basic driving aids including Electronic Stability Control (ESC), Anti-lock Braking Systems (ABS), and Adaptive Cruise Control (ACC) are included in Level 1. Advanced assistance is included at level 2, such as longitudinal/lateral control or emergency braking. At level 3, the system can, in some circumstances, monitor its surroundings and engage in fully autonomous driving. However, if the driving job veers outside the autonomous system's operational area, the human operator must still take over. A vehicle with level 4 automation may drive completely autonomously under certain circumstances and will safely take over control if the operator refuses to do so when asked. All driving modes are completely automated for level 5 systems, meaning full autonomy and basically no need for a (backup) driver at all. This is visualized in figure 1.1. With this thesis, we aim to contribute toward level 4. With a follower, we remove the need for a human to take over in urban scenarios.

1.3. The Vulnerable Road User

So-called Vulnerable Road Users (VRUs), such as bicyclists and pedestrians, account for half of the 1.3 million annual casualties. To increase the safety of VRUs, active safety technologies such as Autonomous Emergency Braking (AEB) are increasingly being integrated with commercial cars seen on the road. Additionally, some cars already have automated steering functionalities.

In complicated urban areas with VRUs present, significant obstacles must be overcome to ensure safety and performance while driving. The self-driving car should be aware of the VRUs' existence and able to deduce their intent to adjust its course and prevent accidents. This is why motion planning techniques are necessary to deliver safe (collision-free) and system-compliant performance in challenging situations with moving and stationary obstacles.

1.4. Local Motion Planner Architecture

The design of a self-driving car's autonomous system is often divided into two key components: the perception system and the decision-making system [2]. The perception system could be broken down into several subsystems that are in charge of functions including autonomous vehicle localization, mapping of static obstacles, mapping of road infrastructure, detection and tracking of moving obstacles, detection and identification of traffic signalization, and more. Although this separation is somewhat vague

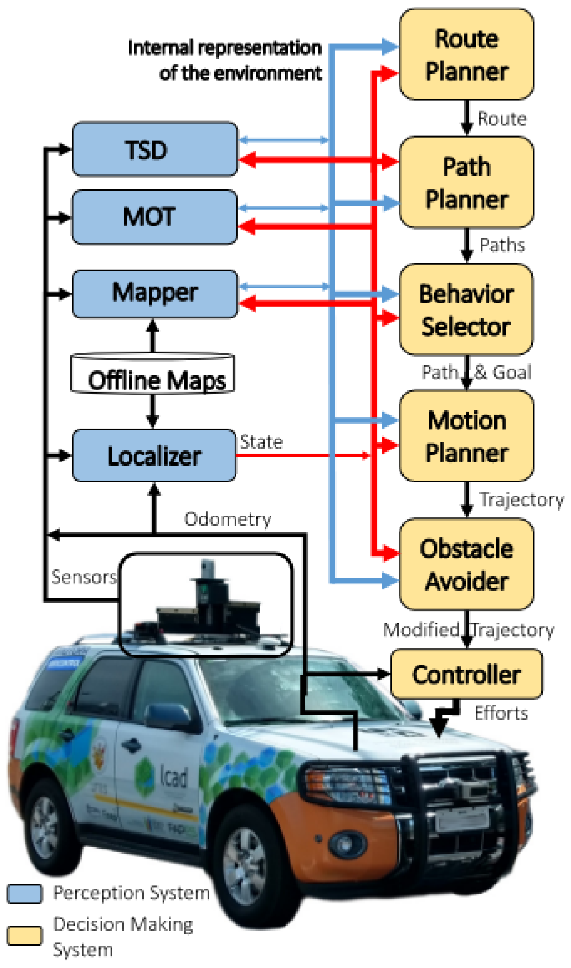


Figure 1.2: The autonomous vehicles hierarchy according to *Badue*: *"Overview of the typical architecture of the automation system of self-driving cars. TSD denotes Traffic Signalization Detection and MOT denotes Moving Objects Tracking."* [5]

and there are multiple distinct versions in the literature, the decision-making system is frequently divided into many subsystems responsible for activities including global route planning, behavior selection, local path planning, or trajectory planning including obstacle avoidance, and control, also called following or tracking. This hierarchy is visualized in the hierarchies presented in the survey by *Badue* [5] , in figure 1.2. We assume the route waypoints and obstacle trajectories as given and focus on trajectory planning around obstacles and trajectory following regarding local feedback control of the vehicle.

1.5. Thesis Objectives

This thesis aims to develop an implementation of a trajectory planner-follower architecture including a feedback-hierarchical interface between the two controllers. This way we split the responsibility of obstacle avoidance from vehicle stability and actuator control, while still maintaining a feedback cycle considering the mismatch between the controllers. The method extends the existing motion planner of the *SafeVRU* [6] platform, and builds on the work from [7].

A sensitivity analysis is done regarding the follower prediction horizon, integration step size, and update frequency. This analysis includes rigorous testing in simulation and a comparison between different settings and with the follower disabled.

2 Preliminaries

This chapter introduces other work related to the thesis. The ROS [8] packages used are maintained by the R2CLab [9].

2.1. Model Predictive Control

For the design of a planner and follower, Model Predictive Control (MPC) is used. Due to its ability to deal with Multi-Input Multi-Output systems, model-based control dates back to 1970 when it was used in the process industry. This control technique suits our purpose well due to its predictive capabilities which enable anticipatory corrections, easy extension to multi-variable non-linear processes, and handling of constraints representing things like the road or obstacles [10].

Techniques based on numerical optimization minimize or maximize a function with constrained variables. Function optimization and model-predictive approaches such as model predictive control (MPC), are the most often used numerical optimization-based strategies for self-driving car trajectory planning [11]. By minimizing a cost function that takes into account the limiting factors or constraints of the trajectory, such as position, velocity, acceleration, and jerk, function optimization methods determine a trajectory. These techniques may conveniently take into account the kinematic, dynamic, and environmental constraints of the vehicle and its surroundings. However, because each motion state is optimized separately and depends on global waypoints, these strategies have a large computational cost.

The foundation of MPC is optimal control [10]. The fundamental idea of MPC is to predict system behavior using a model of this system. The so-called Optimal Control Problem is solved for a finite horizon to obtain the optimal predicted control input variables of each point in the horizon. This is illustrated in figure 2.1. The control input variable(s) of the first point in the horizon will be selected as the control action to be executed. Then the other control actions are discarded and the process is repeated for the next time step [12]. More information about Model Predictive Control can be found in dedicated literature on the subject, such as by *Rawlings et al. (2020)* [10].

A controller using MPC solves the following nonlinear model predictive control problem:

$$\begin{aligned}
\min_{\boldsymbol{u}} \quad & \sum_{k=0}^{N-1} J\left(\boldsymbol{\xi}_k, \boldsymbol{u}_k\right) \\
\text{s.t. :} \quad & \boldsymbol{\xi}_{k+1} = h\left(\boldsymbol{\xi}_k, \boldsymbol{u}_k\right), k = 0, \ldots, N-1 \\
& \boldsymbol{u}_{\text{min}} \leq \boldsymbol{u}_k \leq \boldsymbol{u}_{\text{max}}, \boldsymbol{\xi}_{\text{min}} \leq \boldsymbol{\xi}_k \leq \boldsymbol{\xi}_{\text{max}} \\
& \boldsymbol{\xi}_0 = \boldsymbol{\xi}_{\text{current}} \\
& \boldsymbol{u} := \left[\boldsymbol{u}_0^T, \ldots, \boldsymbol{u}_{N-1}^T\right]^T, \boldsymbol{\xi} := \left[\boldsymbol{\xi}_0^T, \ldots, \boldsymbol{\xi}_{N-1}^T\right]^T,
\end{aligned} \tag{2.1}$$

where $J\left(\boldsymbol{\xi}_k, \boldsymbol{u}_k\right)$ is the objective function. The predicted state and input of the vehicle at prediction step $k = 0, \ldots, N-1$ (N is the prediction horizon), are denoted by state variables $\boldsymbol{\xi}_k \in [\boldsymbol{\xi}_{\text{min}}, \boldsymbol{\xi}_{\text{max}}]$

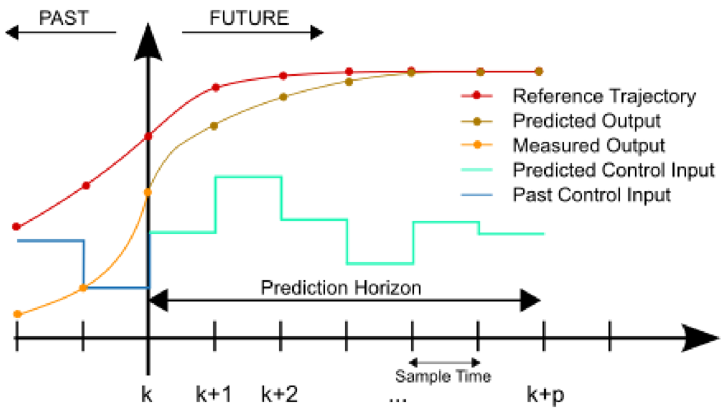


Figure 2.1: Basic MPC scheme by *Martin Behrendt* [13]

and input variables $u_k \in [u_{\text{min}}, u_{\text{max}}]$ respectively, and ξ_{current} is the measured state of the vehicle. The control problem in equation (2.1) minimizes an objective function with certain constraints, such as respecting the prediction model and state and input constraints.

The primary consideration when selecting a model for implementation is the trade-off between accuracy and computational cost. A highly intricate model, characterized by mathematical complexity, can effectively capture the future behavior of a system when exposed to known inputs. However, achieving this level of accuracy requires additional computational effort for performing calculations, as it involves a greater number of operations. The computational effort is further influenced by the analysis of the integration time step and horizon length. This may become problematic regarding embedded hardware limitations of the system.

Another consideration is that in practice every system faces limitations imposed by safety measures, environmental regulations, consumer requirements, and physical restrictions like power and operating speed. These constraints can be categorized into two main types:

- Hard constraints, which must not be violated under any circumstances, as they establish the feasible range of solutions.
- Soft constraints, which can be temporarily violated if necessary. They serve to address extreme initial conditions and ensure continuity between successive states.

2.1.1. Local Model Predictive Contouring Control

MPC is put into practice in the *lmpcc* package. This package uses Model Predictive Contouring Control (MPCC) [14] to perform trajectory planning by implementing a ForcesPro solver [15]. MPCC solves the optimization problem of Eq. (2) in Section II of the paper. ForcesPro generates an efficient solver that gives a fast and reliable numerical solution to the mathematical optimization problem.

A *lmpcc_follower* package was added to implement trajectory following, also performing MPCC, but with different requirements and thus a different ForcesPro solver. The follower package was built to integrate well with the existing planner functionalities. Therefore the prediction models, the modules responsible for adding constraints and objectives, and the interfaces for communicating with a simulator or real robot are defined in one common place and used by both controllers.

2.2. Simulation

Verification tests were done in a simulated environment. This environment consists of an ego-vehicle simulator, a roadmap, and a pedestrian simulator. These are visualized in figure 2.2.

2.2.1. Ego-Vehicle Simulator

The *simple_sim* package was used to simulate the ego-vehicle dynamic behavior. This package performs numerical integration using the *scipy.integrate* method [16]. The integration method uses the integration function RK45, which solves the initial value problem (IVP) [17] given by equation (2.2):

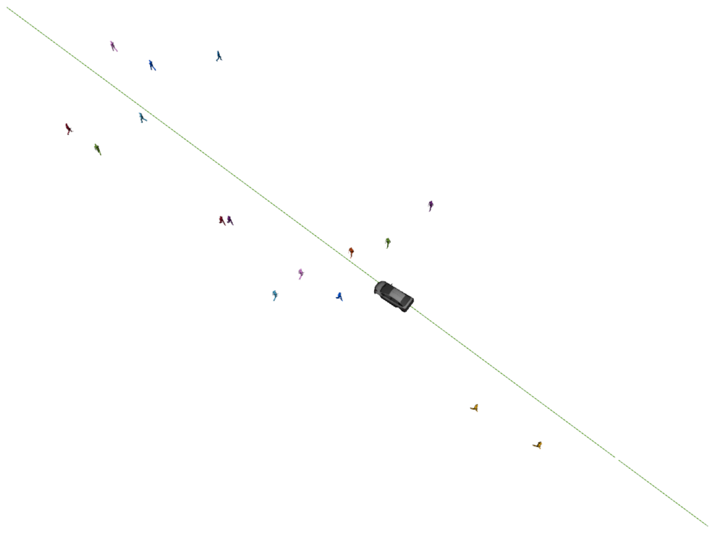


Figure 2.2: The simulation environment visualized in RViz. We see the ego-vehicle as a car model, pedestrians as human figures in different colors and the roadmap route reference as a green line.

$$\begin{aligned} \boldsymbol{\xi}(\boldsymbol{u})' &= f\left(\boldsymbol{u}, \boldsymbol{\xi}(\boldsymbol{u})\right) \\ \boldsymbol{\xi}(\boldsymbol{u}_0) &= \boldsymbol{\xi}_0 \end{aligned} \tag{2.2}$$

with initial state $\boldsymbol{\xi}_0$ known. A kinematic bicycle and dynamic bicycle model were added as vehicle models. The dynamic model implements either a linear or a Dugoff tire model. These models are given in Section III of the thesis paper. The simulator runs at and outputs the state $\boldsymbol{\xi}$ at 200 Hz. It updates the vehicle model inputs $\boldsymbol{u}$ at the rate of controller frequency. The controllers both get their states $\boldsymbol{\xi}$ directly from the simulator.

Model validation

The vehicle and tire models were validated by comparing their output with a 'back-of-the-envelope' calculation, using the point mass centripetal force formula equation (2.3):

$$F_c = \frac{mv^2}{r} \tag{2.3}$$

where F_c is the centripetal force, m is the mass, v is the longitudinal velocity and r is the radius. The vehicle drives in a constant-radius circle with constant longitudinal velocity to calculate lateral tire forces, which should equal the centripetal force exerted on a point mass according to the formula above. Results of these experiments are visualized in figure 2.3. Tests were conducted with various longitudinal velocities and radii.

2.2.2. Roadmap

To describe the route which the vehicle has to traverse, we make use of the *roadmap* package. This package takes a route consisting of waypoints (X, Y, Ψ) points and converts these waypoints to polylines. It takes the hardcoded waypoints and fits lanes through the waypoints via a Clothoid and/or Cubic spline fitting. A spline is generated over the received waypoints in two steps:

1. A Clothoid spline is fitted through waypoints and is sampled to obtain equally spaced waypoints at consistent spacial frequency.
2. Cubic splines are fitted through each segment of the Clothoid spline.

A path is then generated from these splines and used as a reference by the trajectory planner.

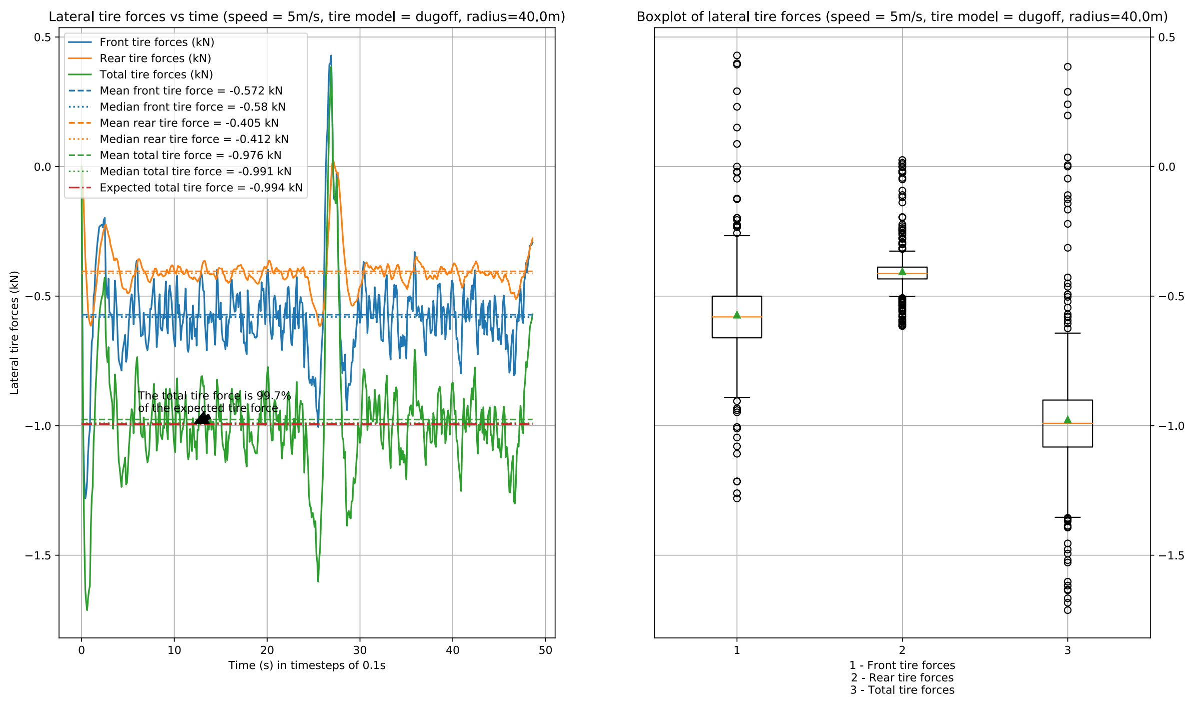


Figure 2.3: Result of vehicle and tire model validation test. On the left, we see the measured and expected tire forces and boxplots of those tire forces on the right.

2.2.3. Pedestrian Simulator

To simulate dynamic obstacles representing Vulnerable Road Users (VRUs), the *pedestrian_simulator* package was used. It simulates obstacles with positions (X, Y), orientation Ψ, and forward velocity v_x. The pedestrians spawn at a specified position and move towards a specified relative or absolute goal position with a specified forward velocity. These parameters may be set exactly or within a range. The pedestrians move independently of each other and the ego-vehicle.

3
Paper

The thesis is written in the form of a paper and is attached to the following pages.

Trajectory Planning and Following in Urban Environments

Victor van der Drift
Department of Cognitive Robotics
Delft University of Technology
Delft, The Netherlands

***Abstract*—This paper presents a comprehensive approach to integrating a trajectory planner and follower for autonomous vehicles (AVs) using model predictive contouring control (MPCC). The planner generates collision-free trajectories with a kinematic bicycle model, while the follower tracks them using a dynamic bicycle model with a smaller integration step size and higher update frequency. The hierarchical architecture allows for a long planning horizon and a fast control loop. Mismatches between the planner and follower can result in tracking errors and conservative trajectories. To address this, a feedback-hierarchical interface is proposed, feeding back the mismatch error from follower to planner. Obstacles are then inflated with this error, minimizing harmful deviations and collision risks. The paper validates the Local Motion Planner using a simulator with a dynamic bicycle model, testing different follower-solver settings in urban scenarios. The results show a reduction in collision rate of 23% compared to a single-layer MPCC planner, with similar levels of lateral error and task duration.**

***Index Terms*—Autonomous Vehicle Navigation, Motion and Path Planning, Trajectory Tracking, Collision Avoidance, Model Predictive Contouring Control**

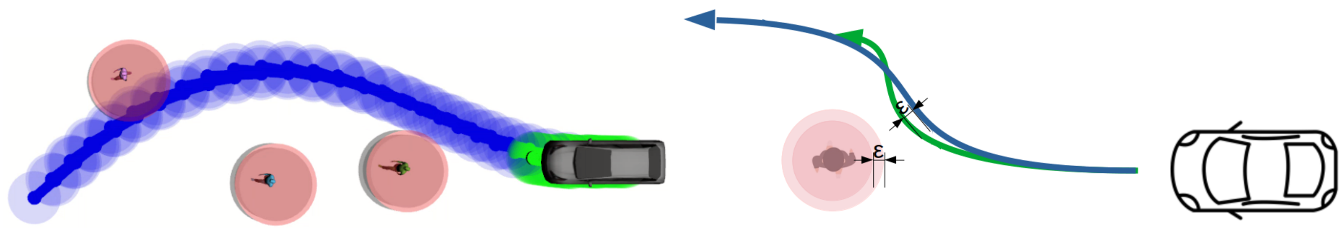
I. Introduction

Autonomous vehicles provide numerous advantages, including increased safety, accessibility, and energy efficiency. Much effort has gone into studies and development of Advanced Driver-Assistance Systems (ADAS) systems both by academics and industry since the middle 1980s [1]. To reach levels of high automation, the vehicle should be capable of performing all driving functions. However, ensuring the safety of Vulnerable Road Users (VRUs) such as pedestrians and cyclists is a major challenge. In this paper, a Local Motion Planner (LMP) is proposed that incorporates a two-layer planner-follower architecture, working towards a level of driving automation 4 by the SAE J3015 standard [2] to reduce the number of accidents caused by human driver error.

Motion planning techniques are important for delivering safe and system-compliant behavior in complex urban environments with VRUs. They highlight the importance of a clear interface between the perception and decision-making subsystems in the autonomous system, which should comprise global route planning, behavior selection, local trajectory planning, and actuator control.

Overall, this study attempts to provide insights into the challenges and potential related to the employment of ADAS in self-driving cars, particularly in terms of assuring the safety of VRUs. This study is a continuation of the work done by *Molteni et al. (2021)* [3], who researched and implemented a Local Motion Planner in MATLAB. The focus of this work is on the implementation of a trajectory follower and feedback-hierarchical interface within the LMPCC Model Predictive Contouring Control [4] ROS package, visualized in Fig. 1.

The LMP should predict a trajectory over a sufficiently long horizon to foresee possible collisions. However, it should also perform well regarding the car dynamics within its environment. Unforeseen deviations from the trajectory may result in collisions with obstacles, even if the planner provided a collision-free trajectory. Since the LMP uses an optimization-based solution, there exists a possibility of optimization failure. The LMP must be able to handle these situations robustly.



(a) Actual obstacle inflation in action, visualized in RViz

(b) Obstacle inflation illustration (exaggeration)

Fig. 1: Predicted planner trajectory in *blue* and predicted follower trajectory in *green*. The circles visualize the safe areas of the vehicle and obstacles. The feedback-hierarchical interface is visualized as safe radii of the obstacles are inflated by ϵ (light red area). *Note: It may look like the blue and red areas collide in Fig. 1a, but this is not the case, since the planner considers dynamic obstacle future trajectories.*

A. Related work

This section provides an overview of the related research on planning and following modules in autonomous driving systems. It focuses on developing planning algorithms to generate safe and collision-free trajectories. Additionally, various path-following control methods are discussed to ensure accurate trajectory execution, while respecting vehicle dynamics. The integration between the planning and following modules is discussed, including hierarchical architectures and feedback-hierarchical interfaces. The research in these areas aims to enhance the capabilities of autonomous vehicles and improve their effectiveness and reliability in autonomous driving in urban scenarios.

1) Planning: The cornerstone of autonomy in autonomous driving is critical decision-making. Planning is a decision-making process that occurs between vehicle perception and control and may output paths or trajectories. Path planning algorithms construct a geometric path from an initial to a final point, passing through pre-defined via-points in the joint space or the workspace of the robot. Trajectory planning algorithms augment a given geometric path with temporal information [5]. The basic goal of planning is to offer AVs a safe and collision-free trajectory to their destinations while taking into consideration vehicle motion, maneuverability around other road users, traffic regulations, and road boundaries [6]. Since this work aims to ensure VRU safety in urban environments, planning is an essential research and development area as it plays a vital part in dynamic obstacle evasion.

Numerous research on planning has been done in recent years. A clear overview of different state-of-the-art motion planning techniques including grid-based techniques, potential field approaches [7], sampling-based approaches, and numerical discrete optimization, such as model predictive control (MPC) [4], [8]–[10] is given by *Gonzalez et al. (2016)* [11]. Essential within planning for obstacle avoidance, is the correct handling of uncertain environments. Scenario-Based trajectory optimization in uncertain dynamic environments translates probabilistic constraints into deterministic ones [12]. Lagrange duality between autonomous agents and dynamic obstacles with uncertain predictions is used to obtain deterministic reformulations of the collision avoidance constraints [13].

The two levels of present planning algorithms are route planning and path or trajectory planning. A route planner plans the journey route from start to finish, carving the whole route up into several sections divided by waypoints. An example of this would be your go-to consumer navigation application, such as Google Maps. The path or trajectory planner makes sure these sections are well-traversed, adhering to necessary requirements. A digital map and localization system are used in the route planning step to identify routes and vehicle states. A path or trajectory can be constructed at the path or trajectory planning stage using a route and ambient information gathered from sensors like cameras, lidars, or radars [1].

2) Following: The following module (also known as tracking or control) receives the trajectory created by the planning module and computes control commands which are sent to the actuators of the steering wheel, throttle, and brakes of the AV. In this way, the follower executes the planned trajectory as best as the physical environment allows [1].

To minimize errors brought on mostly by the inaccuracies of the trajectory planner prediction model, tracking techniques stabilize tracking the trajectory generated by the trajectory planner. Numerous path-following control methods have been analyzed [14]. Model-free approaches such as Pure Pursuit [15], PID-Stanley [16], and Neural Network [17] do not require extensive resources regarding hardware and software, and are relatively simple to implement. Model-based path-following methods include sliding-mode control [18], LQR controllers [19] and MPC controllers [3], [20]–[22] or a H_∞-MPC disturbance rejection and trajectory following combination [23]. Regarding the models necessary, passivity-based Model Predictive Control (PB/MPC) is a method enabling the inclusion of high complexity models by stabilizing a system using a framework of energy shaping, adding to the constraints of the MPC problem [24], [25]. Also, modeling steering system delay [26] or road dynamics [27] and estimating model parameters such as cornering stiffness [26] and its uncertainty [28] could greatly improve effectiveness and reliability of a follower.

3) Interaction interfaces: The integrated architecture and interaction between planner and follower are not trivial. The simplest method of integrating separate systems is to design a one-way top-down hierarchical system, which feeds the path or trajectory from the planner into the follower directly [29]. Hierarchical planner-follower architectures benefit from both a large planner time horizon to effectively avoid obstacles and a fast follower control loop to ensure the stability and performance of the system.

However, if the mismatch of the planner and follower is too large, the target trajectory from the planner may be too conservative or impossible to track for the follower [30]. This can be overcome by integrating the architecture, for example by merging the planning and following into one optimization [30]–[32], or with a feedback-hierarchical interface between planner and follower [3]. If the follower becomes infeasible or the (soft) constraints are violated, the planner should re-plan a trajectory concerning the limits of the follower.

Another hierarchical control scheme for trajectory planning and following is using two Nonlinear Model Predictive Controllers, linking the terminal velocity of the follower to the corresponding value of the planner [21].

An adaption of a simultaneous local path planning and path following control framework, performing lane change and double-lane change maneuvers dynamically changes the lookahead distance with changing road shapes and curvatures, resulting in better performance regarding route error [33].

When using decoupled stabilization systems such as electronic stability control (ESC), collision avoidance may not function as intended due to the ESC stabilization actions leading to tracking errors. A single hierarchical MPC framework solves this by prioritizing collision avoidance above stabilization above path tracking by using soft constraints with different decreasing weights [34].

B. Contribution

In this work, an implementation of a trajectory planner-follower architecture with a feedback-hierarchical interface is presented. By adding a follower that also tracks velocity, the evasive performance is increased while keeping lateral deviation from the route and task duration similar as opposed to without a follower. A sensitivity study regarding the follower solver settings was done. The problem of planner-follower mismatch is addressed by using the lateral offset error to increase safety margins in planner obstacle avoidance. This feedback-hierarchical interface is easier to implement than fully integrated interfaces, while still taking into account planner inaccuracies and planner-follower mismatch. A PID-Stanley backup follower is implemented to handle MPC infeasibilities. The proposed planner-follower architecture is validated in simulation. The system is implemented in ROS/C++ and integrates within the *SafeVRU* [35] framework.

II. Preliminaries

A. Robot Description

We consider an ego-vehicle or robot as a nonlinear discrete-time system, whose dynamics can be represented by the following model:

$$\boldsymbol{\xi}_{k+1} = h\left(\boldsymbol{\xi}_k, \boldsymbol{u}_k\right) \tag{1}$$

where $\boldsymbol{\xi}_k \in \mathbb{R}^{n_\xi}$ and $\boldsymbol{u}_k \in \mathbb{R}^{n_u}$ are the state and input at discrete time instance k, and n_ξ and n_u are the state and input dimensions respectively. The robot moves in workspace $(X, Y, \Psi) \in \mathbb{R}^3 \subseteq \mathbb{R}^{n_\xi}$, which is a horizontal plane with orientation. Obstacles move in this same space.

The robot is given obstacle trajectories in (X, Y) by a pedestrian simulator, a reference path $(X, Y) \in \mathbb{R}^2 \subseteq \mathbb{R}^{n_\xi}$ by a route planner, and reference velocity along this path v_x as constant. The robot will track the reference path and velocity but can deviate from this reference to avoid collisions with obstacles. The robot is modeled as a non-holonomic system and its area is modeled as a set of circles of radius r_{ego}. The obstacles are modeled as moving circles of radius $r_{obstacle}$.

B. Problem Formulation

For the design of a planner and follower, Model Predictive Control (MPC) is used. This control technique suits our purpose well due to its predictive capabilities, enabling anticipatory corrections and handling of constraints such as the road or obstacles [36].

A controller using MPC solves the following nonlinear model predictive control problem:

$$\begin{aligned}
\min_{\boldsymbol{u}} \quad & \sum_{k=0}^{N-1} J\left(\boldsymbol{\xi}_k, \boldsymbol{u}_k\right) \\
\text{s.t. :} \quad & \boldsymbol{\xi}_{k+1} = h\left(\boldsymbol{\xi}_k, \boldsymbol{u}_k\right), k = 0, \ldots, N-1 \\
& \boldsymbol{u}_{\min} \leq \boldsymbol{u}_k \leq \boldsymbol{u}_{\max}, \boldsymbol{\xi}_{\min} \leq \boldsymbol{\xi}_k \leq \boldsymbol{\xi}_{\max} \\
& \boldsymbol{\xi}_0 = \boldsymbol{\xi}_{\text{current}} \\
& \boldsymbol{u} := \left[\boldsymbol{u}_0^T, \ldots, \boldsymbol{u}_{N-1}^T\right]^T, \boldsymbol{\xi} := \left[\boldsymbol{\xi}_0^T, \ldots, \boldsymbol{\xi}_{N-1}^T\right]^T \\
& \boldsymbol{u}_k \in \mathcal{U}, \boldsymbol{\xi}_k \in \mathcal{X}
\end{aligned} \tag{2}$$

where $J\left(\boldsymbol{\xi}_k, \boldsymbol{u}_k\right)$ is the objective function, which is different regarding the planner or follower and is given in their respective implementations in Section III-A and Section III-B. The predicted state and input of the vehicle at prediction step $k = 0, \ldots, N-1$ (N is the prediction horizon), are denoted by state variables $\boldsymbol{\xi}_k \in [\boldsymbol{\xi}_{\min}, \boldsymbol{\xi}_{\max}]$ and input variables $\boldsymbol{u}_k \in [\boldsymbol{u}_{\min}, \boldsymbol{u}_{\max}]$ respectively, and $\boldsymbol{\xi}_{\text{current}}$ is the measured state of the vehicle. $\mathcal{X}$ and $\mathcal{U}$ are the set of admissible states and inputs. Eq. (2) consists of minimizing an objective function with certain constraints, such as respecting the prediction model and state and input constraints. The prediction models and constraints for the planner and follower are different and elaborated upon in Section III.

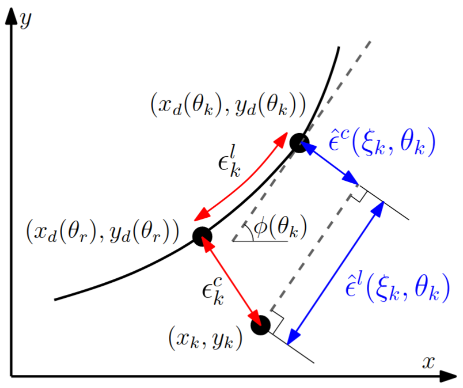
One downside of MPC is that it may get stuck in a local optimum. This could result in unsafe or slow motion, even when a better plan exists. A solution could be a planner layer above the MPC planner, to guide the local MPC planner [37], but this is not considered in this work.

1) Contouring Control: An extension of MPC is called Model Predictive Contouring Control (MPCC). Its goal is to reduce contouring error, which is defined as the normal deviation or shortest distance between the planned path and the current position [4], [8], [10], as seen in Fig. 2. As a result, it is an excellent indicator of how much the vehicle deviates from the reference path in the lateral direction. The contouring error $\hat{\epsilon}^c$ and objective $J_{\hat{\epsilon}^c}$ are defined in Eq. (3):

$$\begin{aligned}
\hat{\epsilon}^c\left(\boldsymbol{\xi}_k, \theta_k\right) =& \sin\phi\left(\theta_k\right)\left(x_k - x_d\left(\theta_k\right)\right) \\
& - \cos\phi\left(\theta_k\right)\left(y_k - y_d\left(\theta_k\right)\right) \\
\phi\left(\theta_k\right) =& \arctan\left(\frac{\nabla y_d\left(\theta_k\right)}{\nabla x_d\left(\theta_k\right)}\right) \\
\theta_{k+1} =& \theta_k + v_k, v_k \in [0, v_{max}], v_{max} > 0 \\
J_{\hat{\epsilon}^c}\left(\boldsymbol{\xi}_k, \theta_k\right) =& \hat{\epsilon}^c Q_{\hat{\epsilon}^c} \hat{\epsilon}^c
\end{aligned} \tag{3}$$

where (x_k, y_k) are the x- and y-position of the system respectively, $\left(x_d\left(\theta_k\right), y_d\left(\theta_k\right)\right)$ are the desired path coordinates along path parameter (θ_k) at time k and v_k and v_{max} are the current and maximum longitudinal velocities along the path. $Q_{\hat{\epsilon}^c}$ is a weight. This contouring error penalized in the objective along the MPC horizon in Eq. (2) [8].

In *Lam et al. (2013)* [9] the MPCC method subject to state and actuator constraints facilitates real-time implementation, by adjusting actuation speed automatically to maintain accuracy. The applicability of the design with more complex



y
$(x_d(\theta_k), y_d(\theta_k))$
ϵ_k^l
$\hat{\epsilon}^c(\xi_k, \theta_k)$
$(x_d(\theta_r), y_d(\theta_r))$
$\phi(\theta_k)$
ϵ_k^c
$\hat{\epsilon}^l(\xi_k, \theta_k)$
(x_k, y_k)
x

Fig. 2: Contouring error $\hat{\epsilon}^c$ by *Lam et al (2010)* [8]. The contouring error is the lateral distance between the path and the ego-vehicle.

robotic models is shown with the successful implementation of the model of an AV (modified Toyota Prius) [4], [35].

2) Collision avoidance: To perform obstacle collision avoidance, ellipsoidal constraints $c_k^{\text{obst},j}(\boldsymbol{\xi}_k) > 1$ are added to Eq. (2) [4]. The ellipsoidal constraint is given by Eq. (4):

$$\underbrace{\begin{bmatrix}\Delta x_k^j \\ \Delta y_k^j\end{bmatrix}^{\mathrm{T}} R(\psi)^{\mathrm{T}} \begin{bmatrix}\frac{1}{d^2} & 0 \\ 0 & \frac{1}{d^2}\end{bmatrix}^{\mathrm{T}} \begin{bmatrix}\Delta x_k^j \\ \Delta y_k^j\end{bmatrix}^{\mathrm{T}} R(\psi)}_{c_k^{\text{obst},j}(\boldsymbol{\xi}_k)} > 1 \tag{4}$$

We approximate the area of the ego-vehicle as a union of n_{discs} circles representing discs with radius r_{ego} and the area of obstacles with a circle of radius $r_{obstacle}$. Thus the distance d kept between the ego vehicle is $d = r_{ego} + r_{obstacle}$. The distance between an obstacle and ego-vehicle disc j is separated into its Δx^j and Δy^j components and $R(\psi)$ is a rotation matrix.

III. Method

For the design of the Local Motion Planner, we build on the existing MPCC planner from *Brito et al (2019)* [4], having a long horizon with a large integration step size and a simple kinematic prediction model. We extend the planner with a follower, another MPCC to track the planned trajectory. The follower uses a dynamic model with a smaller integration step size to better predict actual vehicle behavior. Control inputs should include dynamic behavior, resulting in longitudinal acceleration and steering rate. The follower will feed the planner-follower mismatch back to the planner to correct the deviation and has a PID controller as a backup follower if the MPCC does not produce feasible results. See Fig. 3 for a diagram of this architecture.

A. Planning

The trajectory planner produces a collision-free trajectory over a long planning horizon in the dynamic environment surrounding the vehicle. It is responsible for following the road and a reference velocity while avoiding obstacles. It uses

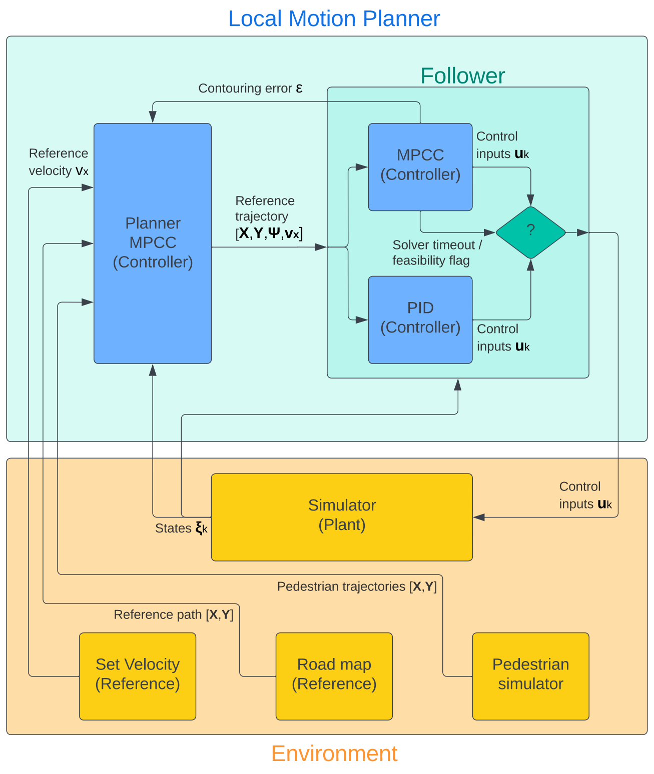


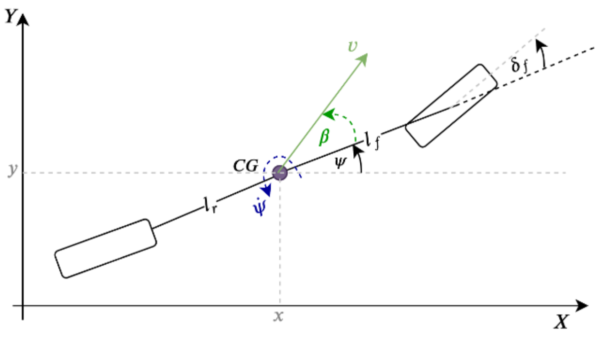
Fig. 3: A diagram of the architecture of the Local Motion Planner and the Environment. The output from either the MPCC or PID controller will be chosen, depending on the status of the MPCC controller. See Section III-B.

MPCC with a kinematic bicycle model [3], [38], such as in Fig. 4a, as a prediction model to control longitudinal acceleration a_x and steering rate ω simultaneously. The planner solves a nonlinear model predictive control problem, such as Eq. (2), including contouring penalty $J_{\hat{\epsilon}^c}$ and collision avoidance constraint $c_k^{\text{obst},j}$ explained in Section II. Its prediction horizon consists of $N = 25$ steps with $dt = 0.2s$, thus resulting in a horizon of 5 seconds. The planner updates at a frequency of $10Hz$. State and input constraints are given by Table I.

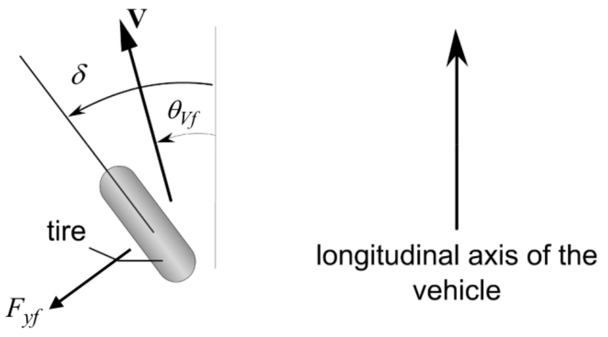
The kinematic bicycle model is given by Eq. (5):

TABLE I: Constraints on the planner and follower MPCC modules.

Symbol	Lower Bound	Upper Bound	Units
v_x	0.10	6.00	m/s
v_y	-1.00	1.00	m/s
r_z	-1.00	1.00	rad/s
δ	-0.45	0.45	rad
a_x	-6.00	2.00	m/s^2
ω	-0.20	0.20	rad/s



(a) Kinematic bicycle model [3]



(b) Front tire slip angle and lateral force [39]

Fig. 4: Bicycle and tire models. In these figures, v and V are the longitudinal velocity also denoted by v_x in Eqs. (5) and (7).

$$
\begin{aligned}
\dot{X} &= v_x \cos(\Psi + \beta) \\
\dot{Y} &= v_x \sin(\Psi + \beta) \\
\dot{\Psi} &= \frac{v_x}{l_r} \sin(\beta) \\
\dot{v_x} &= a_x \\
\dot{\delta} &= \omega \\
\beta &= \tan^{-1}\left(\frac{l_r}{l_f + l_r} \tan(\delta)\right)
\end{aligned} \tag{5}
$$

with states $\boldsymbol{\xi} = [X, Y, \Psi, v_x, \delta]$ and inputs $\boldsymbol{u} = [a_x, \omega]$. X, Y, and Ψ represent the vehicle's absolute position and orientation on the map, v_x and $\dot{\Psi}$ are the vehicle longitudinal velocity and the yaw rate respectively. The longitudinal acceleration a_x is directed parallel to the longitudinal axis of the vehicle. δ is the road wheel angle and ω the steering change rate. β is the sideslip angle and l_f and l_r are the distances of the front and rear axes to the center of gravity of the vehicle respectively. Values for these constants are given by Table II.

B. Following

The follower is used to track the collision-free trajectory that is calculated by the planner. This trajectory consists of the kinematic states X, Y, Ψ, v_x along the prediction horizon of the planner.

Since the sampling frequencies and integration step sizes of the planner and follower will not align, re-sampling is necessary. This was done by fitting and evaluating the kinematic states of the trajectory through cubic splines to obtain a continuous trajectory, consisting of segments $\boldsymbol{\tau}^i(t) = \left[\tau_x^i, \tau_y^i, \tau_\Psi^i, \tau_{v_x}^i\right]^T$, with τ_j^i given by Eq. (6):

$$
\tau_j^i(t) = a_j^i t^3 + b_j^i t^2 + c_j^i t + d_j \tag{6}
$$

where j could be one of states X, Y, Ψ, v_x. Together the cubic splines form a smooth trajectory $\boldsymbol{\tau} : [0, T] \to \mathbb{R}^4$ from the car state to the goal state. These splines are then sampled with the follower integrator step size to use as a reference for the follower controllers. More details about the cubic spline implementation are given by [40].

The follower consists of two controllers: an MPCC controller and a PID-Stanley controller. The output of the MPCC is always preferred, but the PID will be chosen if the MPCC takes too long or becomes infeasible. Both controllers are discussed below.

1) MPCC follower: To track the trajectory in an optimal manner and predict the future dynamic behavior of the car, an MPCC controller was used. The MPCC follower follows the collision-free trajectory provided by the planner while also considering the vehicle dynamics. Therefore the model chosen is a rewritten version from *Laurense (2019)* [20], also used by *Schwarting (2018)* [10]. It is a dynamic model with a linear tire model which is given by Eqs. (7) to (9):

$$
\begin{aligned}
\dot{X} &= v_x \cos(\Psi + \beta) \\
\dot{Y} &= v_x \sin(\Psi + \beta) \\
\dot{\Psi} &= r_z \\
\dot{v_x} &= r_z v_y - \frac{1}{m} F_{\text{lat},f} \sin(\delta) + a_x \cos(\delta) \\
\dot{v_y} &= -r_z v_x + \frac{1}{m}\left(F_{\text{lat},f} \cos(\delta) + F_{\text{lat},r}\right) + a_x \sin(\delta) \\
\dot{r_z} &= \frac{1}{I_z}\left(l_f F_{\text{lat},f} \cos(\delta) - l_r F_{\text{lat},r} + l_f a_x \sin(\delta)\right) \\
\dot{\delta} &= \omega \\
\beta &= \tan^{-1}\left(\frac{l_r}{l_f + l_r} \tan(\delta)\right)
\end{aligned} \tag{7}
$$

with states $\boldsymbol{\xi} = [X, Y, \Psi, v_x, v_y, r_z, \delta]$ and inputs $\boldsymbol{u} = [a_x, \omega]$. X, Y, and Ψ are global coordinates and orientation on the horizontal plane, v_x and v_y are the longitudinal and lateral velocities in the local frame of the vehicle, r_z is the angular velocity of the vehicle around the vertical axis or yaw rate, δ is the steering angle, β is the side slip angle, and a_x and ω represent control variables longitudinal acceleration and steering rate. Model constants l_f and l_r are the distances from the center of gravity to the front and rear axes respectively, as illustrated in Fig. 4a and m and I_z denote the vehicle mass and yaw inertia respectively. Values for these constants are given

TABLE II: Vehicle and tire model constants

Symbol	Value	Units
lf	1.123	m
lr	1.577	m
m	1590	kg
I_z	2830	$\text{kg} * \text{m}^2$
$C_{\alpha f}$	188990	N/rad
$C_{\alpha r}$	194370	N/rad

by Table II. The lateral tire forces $F_{\text{lat},i}$, in which i could be f for the front tire and r for the rear tire, are calculated with tire models explained below.

Regarding tire models, we first calculate the slip angle α_i with Eq. (8), where i may be replaced with f (front) or r (rear). The slip angle of a tire is defined as the angle between the orientation of the velocity vector of the wheel and the orientation of the tire. An illustration of the slip angle is given in Fig. 4b. Here Θ_{Vf} is the angle that the velocity vector at the front wheel makes with the longitudinal axis of the vehicle. Θ_{Vr} will be this angle for the rear tire.

$$\begin{aligned} \Theta_{Vf} &= \tan^{-1}\left(\frac{v_y + l_f r_z}{v_x}\right) \\ \Theta_{Vr} &= \tan^{-1}\left(\frac{v_y - l_r r_z}{v_x}\right) \\ \alpha_f &= \delta - \Theta_{Vf} \\ \alpha_r &= -\Theta_{Vr} \end{aligned} \tag{8}$$

The linear tire model is given by Eq. (9):

$$F_{\text{lat},i} = C_{\alpha i}\alpha_i \tag{9}$$

where $C_{\alpha i}$ is the lateral slip stiffness where i can again be replaced with f, r for front and rear tire respectively. The values used are given by Table II.

The controller solves the nonlinear model predictive control problem given by Eq. (2), and its objectives are:

- path tracking in a contouring control way, minimizing the contouring error of the car with respect to the X, Y trajectory calculated by the planner.
- velocity tracking by using the v_x trajectory calculated by the planner as reference velocity.
- minimizing control inputs.

Therefore the cost function is as follows:

$$J_{follower} := J_{\hat{\epsilon}^c} + J_{v_x} + J_{a_x} + J_\omega \tag{10}$$

where

$$J_{v_x} := \|v_x - v_{x,ref}\|^2_{W_{v_x}} \tag{11}$$

and the contouring objective $J_{\hat{\epsilon}^c}$ uses the contouring error from Eq. (3) and is further explained by *Brito et al. (2019)* [4]. $J_{a_x} + J_\omega$ are cost functions regarding the control inputs. State and input constraints are given in Table I.

2) PID-Stanley follower: This follower functions as a backup, when the MPCC follower becomes infeasible. The X, Y, Ψ states are used by two separate controllers for longitudinal and lateral movement. The longitudinal control is calculated by a double PID controller, which first calculates a velocity and then a control acceleration. To calculate a steering angle for lateral control, we use a steering angle control law from Stanley, the robot that won the DARPA Grand Challenge [16]. The control law is given by Eq. (12):

$$\delta = (\Psi_{ref} - \Psi) + \arctan\frac{K_S(Y_{ref} - Y)}{v_x} \tag{12}$$

where K_s denotes the Stanley gain and the other symbols mean the same as explained at Eq. (5). Subscript $_{ref}$ means reference.

The PID-Stanley follower has no regard for dynamics and performs no predictive evaluation. It does not provide any feedback to the planner and can not correct inputs and is thus only used as a backup.

C. Feedback-hierarchical interface

A feedback-hierarchical interface is implemented in the following way: the reference trajectory in X, Y coordinates, which is given as the prediction horizon from the planner, is interpolated using cubic splines. Then the contouring error between this interpolation and the prediction horizon from the follower is calculated with every follower control loop update. It gives an array of contouring errors $\hat{\epsilon}^c$ along the prediction horizon, using Eq. (3). The maximum value from this array ϵ is then sent to the planner and to inflate safe obstacle distances d, where $d = r_{ego} + r_{obstacle} + \epsilon$ from Eq. (4). In the next planner control loop, the planner now has to account for the predicted deviation from its previous predicted horizon, by planning around the obstacles with inflated radii.

This is illustrated in Fig. 1b, where the blue line indicates the planner horizon and the green line is the follower horizon. The red circle is the regular obstacle safe radius and the ring around this is the inflation of the safe radius ϵ. It can be seen in action in Fig. 1a.

D. Validation

1) Simulation: To validate the control method, the simulator *simple_sim* from the R2CLab was [41] used as a ROS node. It simulates a dynamic bicycle model with a linear or Dugoff tire model. The system was integrated with the SafeVRU framework through an interface for the LMPCC package.

This dynamic bicycle model is given by Eq. (7): In most normal driving scenarios in urban environments, slip angles are small. Thus lateral tire forces can be approximated well by the linear model, due to the tire force being proportional to slip ratio [39]. This means that a linear model will suffice in common urban driving scenarios. However, since evasions imply more dynamic maneuvers and to give the simulator a higher fidelity model than the controller, a Dugoff model [39] was implemented, given by Eqs. (8) and (13):

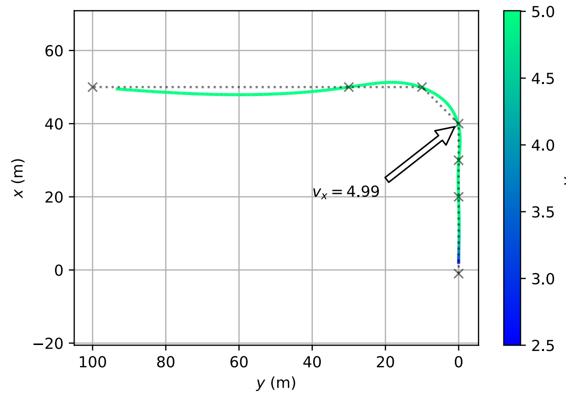


Fig. 5: The track (x, y) and speed v_x of experiment A with MPCC follower. The grey crosses $\times$ are the route via-points.

The Dugoff tire model is given by Eq. (13):

$$\begin{aligned}
\mu &= \mu_0\left(1 - e_r v_x\sqrt{\kappa^2 + \tan^2\alpha_i}\right)\\
\lambda &= \frac{\mu F_z(1-\kappa)}{2\sqrt{(C_{\kappa i}\kappa)^2 + (C_{\alpha i}\tan\alpha_i)^2}}\\
f(\lambda) &= \begin{cases}\lambda(2-\lambda) & \text{if } \lambda < 1\\ 1 & \text{if } \lambda \geq 1\end{cases}\\
F_{\text{lat },i} &= \frac{C_{\alpha i}\tan\alpha_i}{1-\kappa} f(\lambda)
\end{aligned} \tag{13}$$

in which μ is the tire-road friction coefficient, μ_0 represents the peak friction coefficient, e_r is the friction reduction coefficient, κ is the wheel slip, F_z is the normal force and $C_{\kappa i}$ is the longitudinal slip stiffness.

Validation of the dynamic model was done by measuring the total lateral tire forces in simulation while driving in constant-radius circles at constant longitudinal velocity. These forces were compared to steady-state lateral forces regarding centripetal forces on a point mass using the formula $F_y = \frac{mv_x^2}{r}$, where F_y is the total lateral force, v_x is the longitudinal velocity, m is the mass and r is the radius of the circle. Validation tests were done with different radii and velocities.

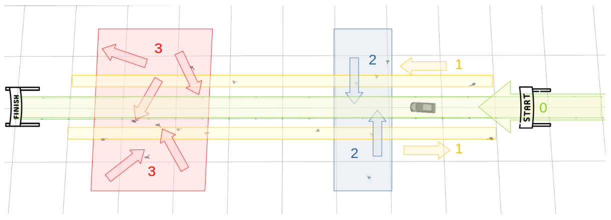


Fig. 7: Pedestrian evasion *Experiment B*.

IV. Experiments

We validate the local motion planner on a simulated self-driving vehicle. Two types of experiments are conducted, one with and one without pedestrians.

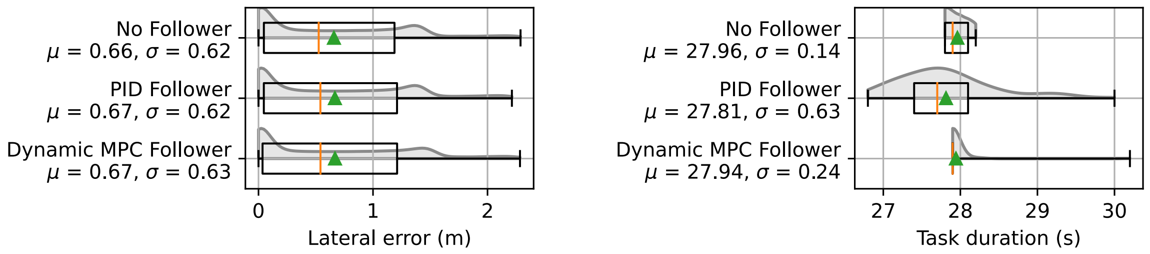
A. Lateral error through curve

The goal of the first experiment is to validate the local motion planner performance regarding lateral deviation from the path and experiment task duration. The lateral deviation is the contouring error between the planner and the reference path, and the experiment task duration is the time it takes from start to finish the experiment. In this experiment, the car will drive straight, make a turn, and then drive straight again, as seen in Fig. 5. No obstacles are present. Tests were done with the follower disabled, with only the PID follower enabled, and with the MPCC follower, using the dynamic model with settings prediction horizon length $N = 25$, integration step size $dt = 0.02s$, and controller clock frequency $cf = 100Hz$.

As seen in Fig. 6, adding a follower hardly affects the lateral error with respect to no follower. The task duration increased a bit in some runs with the follower, but the medians (orange line) and means (green triangle) are still similar in value.

B. Pedestrian evasion

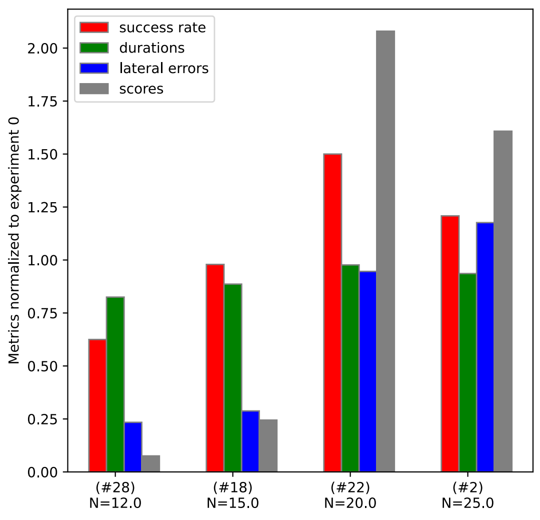
The second experiment tests the ability of the Local Motion Planner to effectively evade pedestrians while still tracking a reference path.



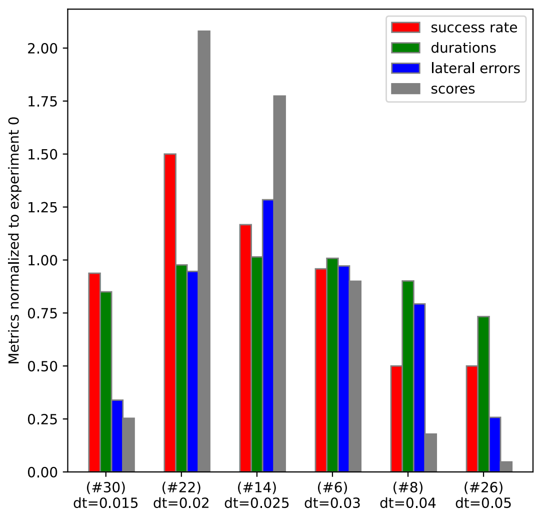
(a) Lateral deviation (m) from the path through the curve.

(b) Experiment task duration (s) through the curve.

Fig. 6: Experiment results of Section IV-A. The line in the middle of the box is the median, the triangle is the mean μ, and σ is the standard deviation. Since the PID follower is not velocity-controlled, a larger spread along its task duration is seen. However, the MPC follower behaves similarly to no follower, and there seems no difference in lateral error between the three.



(a) Horizon lengths N for:
integration stepsize=0.02s and clock frequency=100Hz



(b) Stepsizes dt (s) for:
horizon length N=20 and clock frequency=100Hz

Fig. 8: Results of experiments with settings adjacent to those of promising experiment *(#22)*.

This reference path is set up as a straight road of 100 meters, where 16 pedestrians are spawned in 3 locations, with random positions and orientations within these locations, and with a speed ranging from 0 m/s to 1 m/s. These pedestrians will walk with the following behavior:

1) Parallel to the road, simulating a sidewalk;
2) Orthogonal to the road, simulating a pedestrian crossing;
3) At random on and next to the road, simulating a pedestrian-car shared space.

These locations and behavior are visualized in Fig. 7. The green strip is the reference path or route of the car.

All experiments were done in sets of 100 test runs, to account for statistical factors considering the randomness in pedestrian spawning and the two solvers' optimization processes [42]. For each experiment, the follower horizon length, integration step size, and clock frequency could be set. The results of the experiments regarding different solvers with these settings are given in Table III.

From these results, we regard three metrics:

- Success := $\sum_1^{100}$(no time-out & no collisions)%
- Lateral error := mean of the contouring error
- Duration := mean of the duration until task completion

These metrics are visualized in Fig. 10. The horizontal and vertical axes represent the mean duration and mean lateral error respectively. These should be as low as possible. The success rate is indicated by color, where green means a high success rate and red low success rate. This should be as high as possible. A 1D success rate plot is given next to the scatter plot for extra readability and redundancy. Every dot represents an experiment with different settings. Experiment $\times$ represents an experiment without a follower, so any experiment with a higher success rate, lower duration, and lower lateral error can be considered as performing better.

As seen in Fig. 10, some settings produce worse results than without a follower, but some settings result in a significantly improved Local Motion Planner. Settings corresponding to experiment *#22* (green triangle $\triangle$) seem to have the most potential since it has the highest success rate and no significant reduction in lateral error and task duration. A run of this experiment is visualized in Appendix C.

Another metric is added to compare the settings:

- Score := $\frac{\text{success rate}^2}{\text{lateral error}*\text{duration}}$

This way all metrics are combined into one number to represent the best settings. Since evading obstacles is the main goal of this controller, the success rate metric is weighed heavier. Since experiment *#22* has the highest success rate, it will be compared to experiments with adjacent settings. In Fig. 8a all these metrics are combined into a bar graph. In this graph, the metrics are normalized with their corresponding results of experiment 0 (without a follower), where a higher number indicates more preferable behavior. A normalized success rate larger than 1, indicates a higher success rate compared to experiment 0. A normalized lateral error larger than 1, indicates a smaller lateral error compared to experiment 0. Therefore it is quickly deduced from Fig. 8a that with a

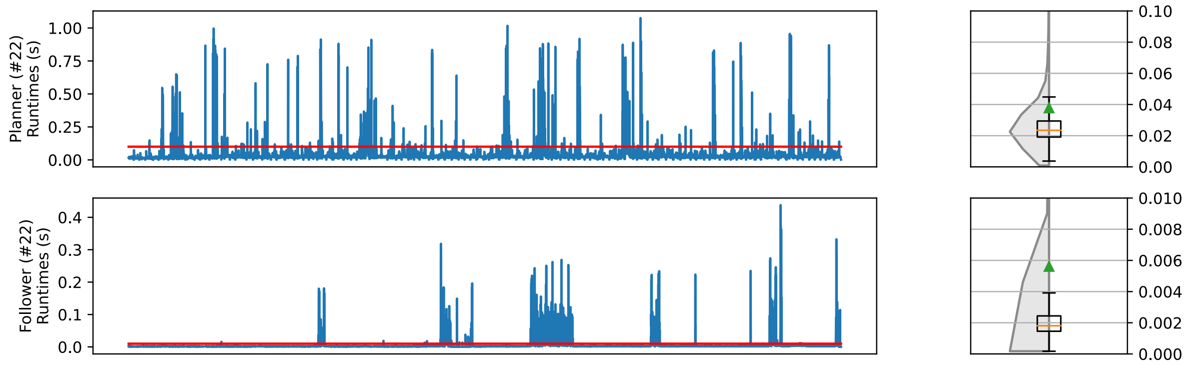


Fig. 9: The control loop runtimes of experiment *#22*. The red line is $1/\text{clock frequency}$. The planner and follower runtimes should be below 0.1 s and 0.01 s respectively. As seen in the plot, this is not always achieved. *Note the unequal axes.*

step size of 0.02 s and clock frequency of 100 Hz, the horizon length should optimally be N=20. Similarly, it can be deduced from Fig. 8b that for a horizon length of N=20, the integration step size should optimally be 0.02 s.

It should also be verified if the planner and follower actually run at their specified clock frequencies. Multithreading helps the parallel calculation of both controllers. For experiment *#22*, the clock frequencies are 10 Hz and 100 Hz for the planner and follower respectively. In Fig. 9 it can be seen that the desired clock frequencies are mostly achieved, but there is still room for improvement. The follower control loop exceeds 0.01 s in 9.8% of the time, triggering the PID-Stanley follower.

To validate the obstacle inflation the settings from experiment *#22* were also tested with obstacle inflation turned off. The results given by Fig. 11 confirm that enabling inflation yields better performance regarding obstacle avoidance.

The common settings of the best experiments were evaluated. It can be concluded from the results that a horizon size (horizon length * integration step size) of around 0.04s to 0.05 seconds with a small integration step size is preferable. A too-small horizon size will lead to the follower not being able to adequately predict the behavior of the car, leading to sudden braking and poor tracking performance. Boxplots from the lateral error Fig. 12 and task duration Fig. 13 show the performance of each experiment regarding those metrics, apart from only the mean of these values in other figures.

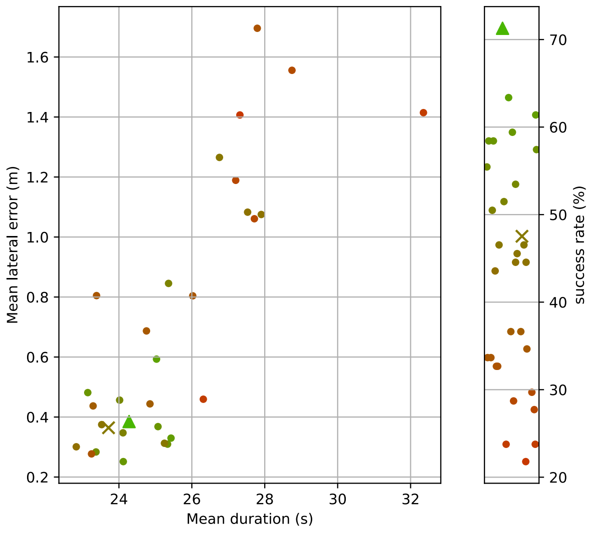


Fig. 10: Results of *scenario a*. Dots correspond to experiments with different settings. $\times$ is the run without a follower, and $\triangle$ is the best run with a follower *#22*.

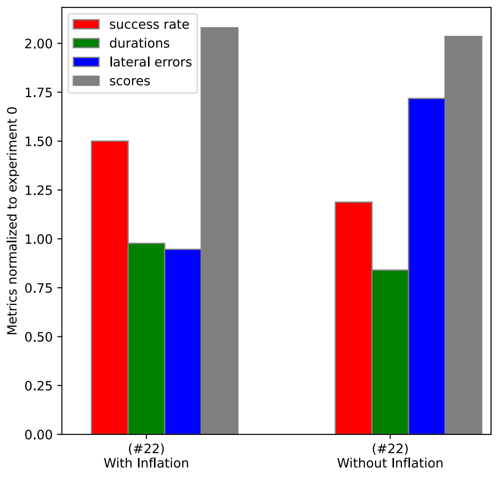


Fig. 11: Inflation validation results for N=20, integration stepsize=$0.02s$ and clock frequency=$100Hz$. Better results are seen with inflation enabled.

V. Conclusion and future work

The implemented Local Motion Planner architecture can increase obstacle avoidance performance in urban scenarios. The addition of a follower and feedback-hierarchical interface improves the success rate of the LMPCC framework by 23% while keeping lateral deviation from the path and task duration time at similar levels as without a follower. The dynamic model used by the follower adds higher certainty regarding pedestrian safety, due to its capabilities of correcting the control inputs of the planner regarding the dynamic behavior of the car. It also gives feedback to the planner if the planned trajectory is not dynamically desirable, in the form of inflation of obstacles by the amount of deviation from the initial plan.

A fail-safe design with a PID-Stanley controller as a backup follower assures the car will continue in a controlled manner, braking if the planner fails too, but still following the intentioned trajectory.

Further work should validate the Local Motion Planner experimentally using the SafeVRU platform. This could include changing road conditions, such as altering the tire-road friction coefficient μ in the simulator and implementing a μ estimator in the follower.

Also, a better representation of the vehicle dynamics could be implemented, by extending the dynamic bicycle model with a delay, actuator and steering dynamics, or 3D chassis dynamics incorporating the roll and pitch of the vehicle in high acceleration movement. For this to become worthwhile, the simulator model should first be upgraded too.

Expanding the dynamics to also include longitudinal jerk, lateral and yaw acceleration, and jerk and steering acceleration could also improve performance while keeping into account motion sickness.

Another addition could be to incorporate the human driver into this system as a form of shared control MPC or replace the PID-Stanley with another type of controller as a backup to increase redundancy. As the MPC planner might get stuck in a local optimum, a layer above the planner guiding it to a more optimal solution could help the planner get better results. And of course, better (auto)tuning may improve results further.

References

[1] C. Badue, R. Guidolini, R. V. Carneiro, *et al.*, "Self-driving cars: A survey," en, *Expert Systems with Applications*, vol. 165, p. 113 816, Mar. 2021, ISSN: 09574174. DOI: 10.1016/j.eswa.2020.113816. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S095741742030628X (visited on 10/24/2022).

[2] A. Serban, E. Poll, and J. Visser, "A Standard Driven Software Architecture for Fully Autonomous Vehicles," *Journal of Automotive Software Engineering*, vol. 1, Jan. 2020. DOI: 10.2991/jase.d.200212.001.

[3] C. Molteni, F. Braghin, and L. Ferranti, "Advanced NMPC-based Local Motion Planner for Autonomous Vehicles," M.S. thesis, POLITECNICO DI MILANO, Milan, Apr. 2021.

[4] B. Brito, B. Floor, L. Ferranti, and J. Alonso-Mora, "Model Predictive Contouring Control for Collision Avoidance in Unstructured Dynamic Environments," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4459–4466, Oct. 2019, Conference Name: IEEE Robotics and Automation Letters, ISSN: 2377-3766. DOI: 10.1109/LRA.2019.2929976.

[5] A. Gasparetto, P. Boscariol, A. Lanzutti, and R. Vidoni, "Path Planning and Trajectory Planning Algorithms: A General Overview," en, in *Motion and Operation Planning of Robotic Systems: Background and Practical Approaches*, ser. Mechanisms and Machine Science, G. Carbone and F. Gomez-Bravo, Eds., Cham: Springer International Publishing, 2015, pp. 3–27, ISBN: 978-3-319-14705-5. DOI: 10.1007/978-3-319-14705-5_1. [Online]. Available: https://doi.org/10.1007/978-3-319-14705-5_1 (visited on 11/23/2022).

[6] X. Hu, L. Chen, B. Tang, D. Cao, and H. He, "Dynamic path planning for autonomous driving on various roads with avoidance of static and moving obstacles," en, *Mechanical Systems and Signal Processing*, vol. 100, pp. 482–500, Feb. 2018, ISSN: 0888-3270. DOI: 10.1016/j.ymssp.2017.07.019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0888327017303825 (visited on 11/10/2022).

[7] C. van der Ploeg, R. Smit, A. Teerhuis, and E. Silvas, *Long Horizon Risk-Averse Motion Planning: A Model-Predictive Approach*, en, arXiv:2209.07093 [math], Sep. 2022. [Online]. Available: http://arxiv.org/abs/2209.07093 (visited on 12/07/2022).

[8] D. Lam, C. Manzie, and M. Good, "Model predictive contouring control," in *49th IEEE Conference on Decision and Control (CDC)*, ISSN: 0191-2216, Dec. 2010, pp. 6137–6142. DOI: 10.1109/CDC.2010.5717042.

[9] D. Lam, C. Manzie, and M. C. Good, "Model Predictive Contouring Control for Biaxial Systems," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 2, pp. 552–559, Mar. 2013, Conference Name: IEEE Transactions on Control Systems Technology, ISSN: 1558-0865. DOI: 10.1109/TCST.2012.2186299.

[10] W. Schwarting, J. Alonso-Mora, L. Paull, S. Karaman, and D. Rus, "Safe Nonlinear Trajectory Generation for Parallel Autonomy With a Dynamic Vehicle Model," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 9, pp. 2994–3008, Sep. 2018, ISSN: 1524-9050, 1558-0016. DOI: 10.1109/TITS.2017.2771351. [Online]. Available: https://ieeexplore.ieee.org/document/8207782/ (visited on 10/04/2022).

[11] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A Review of Motion Planning Techniques for Automated Vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1135–1145, Apr. 2016, Conference Name: IEEE Transactions on Intelligent Transportation Systems, ISSN: 1558-0016. DOI: 10.1109/TITS.2015.2498841.

[12] O. de Groot, B. Brito, L. Ferranti, D. Gavrila, and J. Alonso-Mora, *Scenario-Based Trajectory Optimization in Uncertain Dynamic Environments*, arXiv:2103.12517 [cs], Mar. 2021. [Online]. Available: http://arxiv.org/abs/2103.12517 (visited on 10/04/2022).

[13] S. H. Nair, E. H. Tseng, and F. Borrelli, *Collision Avoidance for Dynamic Obstacles with Uncertain Predictions using Model Predictive Control*, arXiv:2208.03529 [cs, math], Aug. 2022. DOI: 10.48550/arXiv.2208.03529. [Online]. Available: http://arxiv.org/abs/2208.03529 (visited on 11/10/2022).

[14] Z. Lu, B. Shyrokau, B. Boulkroune, S. van Aalst, and R. Happee, "Performance benchmark of state-of-the-art lateral path-following controllers," in *2018 IEEE 15th International Workshop on Advanced Motion Control (AMC)*, ISSN: 1943-6580, Mar. 2018, pp. 541–546. DOI: 10.1109/AMC.2019.8371151.

[15] M. Samuel, M. Hussein, and M. Binti, "A Review of some Pure-Pursuit based Path Tracking Techniques for Control of Autonomous Vehicle," en, *International Journal of Computer Applications*, vol. 135, no. 1, pp. 35–38, Feb. 2016, ISSN: 09758887. DOI: 10.5120/ijca2016908314. [Online]. Available: http://www.ijcaonline.org/research/volume135/number1/samuel-2016-ijca-908314.pdf (visited on 07/01/2023).

[16] S. Thrun, M. Montemerlo, H. Dahlkamp, *et al.*, "Stanley: The robot that won the DARPA Grand Challenge," en, *Journal of Field Robotics*, vol. 23, no. 9, pp. 661–692, Sep. 2006, ISSN: 15564959, 15564967. DOI: 10.1002/rob.20147. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/rob.20147 (visited on 06/28/2023).

[17] I. Rivals, L. Personnaz, G. Dreyfus, and D. Canas, "Real-time control of an autonomous vehicle: A neural network approach to the path following problem," en, 1993.

[18] C. Hu, R. Wang, and F. Yan, "Integral Sliding Mode-Based Composite Nonlinear Feedback Control for Path Following of Four-Wheel Independently Actuated Autonomous Vehicles," *IEEE Transactions on Transportation Electrification*, vol. 2, no. 2, pp. 221–230, Jun. 2016, Conference Name: IEEE Transactions on Transportation Electrification, ISSN: 2332-7782. DOI: 10.1109/TTE.2016.2537046.

[19] J. M. Snider, "Automatic Steering Methods for Autonomous Automobile Path Tracking," en, 2009.

[20] V. A. Laurense, "Integrated motion planning and control for automated vehicles up to the limits of handling," en, Ph.D. dissertation, Stanford University, 2019. [Online]. Available: https://purl.stanford.edu/ym171qz4644 (visited on 12/07/2022).

[21] T. Novi, A. Liniger, R. Capitani, and C. Annicchiarico, "Real-time control for at-limit handling driving on a predefined path," en, *Vehicle System Dynamics*, vol. 58, no. 7, pp. 1007–1036, Jul. 2020, ISSN: 0042-3114, 1744-5159. DOI: 10.1080/00423114.2019.1605081. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/00423114.2019.1605081 (visited on 12/05/2022).

[22] Y. Xie, C. Li, H. Jing, W. An, and J. Qin, "Integrated Control for Path Tracking and Stability Based on the Model Predictive Control for Four-Wheel Independently Driven Electric Vehicles," en, *Machines*, vol. 10, no. 10, p. 859, Sep. 2022, ISSN: 2075-1702. DOI: 10.3390/machines10100859. [Online]. Available: https://www.mdpi.com/2075-1702/10/10/859 (visited on 11/02/2022).

[23] C. Philippe, L. Adouane, B. Thuilot, A. Tsourdos, and H.-S. Shin, "Risk and comfort management for multi-vehicle navigation using a flexible and robust cascade control architecture," in *2017 European Conference on Mobile Robots (ECMR)*, Sep. 2017, pp. 1–7. DOI: 10.1109/ECMR.2017.8098703.

[24] A. Tahirovic and G. Magnani, "Passivity-based model predictive control for mobile robot navigation planning in rough terrains," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, ISSN: 2153-0866, Oct. 2010, pp. 307–312. DOI: 10.1109/IROS.2010.5650821.

[25] A. Tahirovic and G. Magnani, "General Framework for Mobile Robot Navigation Using Passivity-Based MPC," *IEEE Transactions on Automatic Control*, vol. 56, no. 1, pp. 184–190, Jan. 2011, Conference Name: IEEE Transactions on Automatic Control, ISSN: 1558-2523. DOI: 10.1109/TAC.2010.2089654.

[26] G. Chen, J. Yao, H. Hu, Z. Gao, L. He, and X. Zheng, "Design and experimental evaluation of an efficient MPC-based lateral motion controller considering path preview for autonomous vehicles," *Control Engineering Practice*, vol. 123, 2022. DOI: 10.1016/j.conengprac.2022.105164.

[27] Q. Liu, S. Song, H. Hu, T. Huang, C. Li, and Q. Zhu, "Extended model predictive control scheme for smooth path following of autonomous vehicles," *Frontiers of Mechanical Engineering*, vol. 17, no. 1, 2022. DOI: 10.1007/s11465-021-0660-4.

[28] X. Ji, X. He, C. Lv, Y. Liu, and J. Wu, "Adaptive-neural-network-based robust lateral motion control for autonomous vehicle at driving limits," en, *Control Engineering Practice*, vol. 76, pp. 41–53, Jul. 2018, ISSN: 0967-0661. DOI: 10.1016/j.conengprac.2018.04.007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0967066118300881 (visited on 10/14/2022).

[29] Y. Li, J. Fan, Y. Liu, and X. Wang, "Path Planning and Path Tracking for Autonomous Vehicle Based on MPC with Adaptive Dual-Horizon-Parameters," English, *International Journal of Automotive Technology*, vol. 23, no. 5, pp. 1239–1253, 2022, ISSN: 1229-9138. DOI: 10.1007/s12239-022-0109-8.

[30] L. Zhang, B. Li, Y. Hao, *et al.*, "A Novel Simultaneous Planning and Control Scheme of Automated

Lane Change on Slippery Roads," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 11, pp. 20 696–20 706, Nov. 2022, Conference Name: IEEE Transactions on Intelligent Transportation Systems, ISSN: 1558-0016. DOI: 10.1109/TITS.2022.3186429.

[31] V. A. Laurense and J. C. Gerdes, "Long-Horizon Vehicle Motion Planning and Control Through Serially Cascaded Model Complexity," *IEEE Transactions on Control Systems Technology*, vol. 30, no. 1, pp. 166–179, Jan. 2022, Conference Name: IEEE Transactions on Control Systems Technology, ISSN: 1558-0865. DOI: 10.1109/TCST.2021.3056315.

[32] H. Guo, C. Shen, H. Zhang, H. Chen, and R. Jia, "Simultaneous Trajectory Planning and Tracking Using an MPC Method for Cyber-Physical Systems: A Case Study of Obstacle Avoidance for an Intelligent Vehicle," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 9, pp. 4273–4283, Sep. 2018, Conference Name: IEEE Transactions on Industrial Informatics, ISSN: 1941-0050. DOI: 10.1109/TII.2018.2815531.

[33] M. A. Daoud, M. W. Mehrez, D. Rayside, and W. W. Melek, "Simultaneous Feasible Local Planning and Path-Following Control for Autonomous Driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 9, pp. 16 358–16 370, Sep. 2022, Conference Name: IEEE Transactions on Intelligent Transportation Systems, ISSN: 1558-0016. DOI: 10.1109/TITS.2022.3149986.

[34] J. Funke, M. Brown, S. M. Erlien, and J. C. Gerdes, "Collision Avoidance and Stabilization for Autonomous Vehicles in Emergency Scenarios," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 4, pp. 1204–1216, Jul. 2017, Conference Name: IEEE Transactions on Control Systems Technology, ISSN: 1558-0865. DOI: 10.1109/TCST.2016.2599783.

[35] L. Ferranti, B. Brito, E. Pool, *et al.*, "SafeVRU: A Research Platform for the Interaction of Self-Driving Vehicles with Vulnerable Road Users," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, Paris, France: IEEE, Jun. 2019, pp. 1660–1666, ISBN: 978-1-72810-560-4. DOI: 10.1109/IVS.2019.8813899. [Online]. Available: https://ieeexplore.ieee.org/document/8813899/ (visited on 09/19/2022).

[36] J. B. Rawlings, D. Q. Mayne, and M. M. Diehl, *Model Predictive Control: Theory, Computation, and Design*. 2020, vol. 2.

[37] O. de Groot, L. Ferranti, D. Gavrila, and J. Alonso-Mora, *Globally Guided Trajectory Planning in Dynamic Environments*, en, Mar. 2023. [Online]. Available: https://arxiv.org/abs/2303.07751v1 (visited on 06/23/2023).

[38] P. Polack, F. Altché, B. d'Andréa-Novel, and A. de La Fortelle, "The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?" In *2017 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2017, pp. 812–818. DOI: 10.1109/IVS.2017.7995816.

[39] R. Rajamani, *Vehicle Dynamics And Control*. Minneapolis: Springer, 2012, vol. 2, ISBN: 978-1-4614-1432-2.

[40] Tino Kluge, *Cubic splines*, Mar. 2021. [Online]. Available: https://kluge.in-chemnitz.de/opensource/spline/spline.pdf (visited on 06/27/2023).

[41] R. R. C. Lab. "Website." (2023), [Online]. Available: https://r2clab.com/ (visited on 07/08/2023).

[42] A. Zanelli, A. Domahidi, J. Jerez, and M. Morari, "FORCES NLP: An efficient implementation of interior-point methods for multistage nonlinear nonconvex programs," *International Journal of Control*, vol. 93, no. 1, pp. 13–29, May 2017, Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/00207179.2017.1316017, ISSN: 0020-7179. DOI: 10.1080/00207179.2017.1316017. [Online]. Available: https://doi.org/10.1080/00207179.2017.1316017 (visited on 07/03/2023).

[43] V. van der Drift. "Runs experiment 22 video." (2023), [Online]. Available: https://youtu.be/rfdZnYqeMxY?t=744 (visited on 07/11/2023).

Appendix A: Table of experiment results

In this appendix, Table III is given of the results of different follower solvers with their settings. Every experiment consists of 100 test runs in the environment described in subsection IV-B.

Experiment 0 corresponds to an experiment where the follower is disabled, as a benchmark. In total 6 experiments with disabled followers were conducted, of which the results with the highest success rate were chosen. (a) corresponds to the experiment of pedestrian evasion with 16 pedestrians, visualized in Fig. 7, (b) corresponds to the experiment of pedestrian evasion of 2 pedestrians.

The metrics are defined as follows:

- score := $\frac{\text{success}^2}{\text{lateral error}*\text{duration}}$
- success := $\sum_{1}^{\text{runs per experiment}}(\text{no time-out \& no pedestrians hit}) * 100\%$
- lateral error := mean of the contouring error
- duration := mean of the duration until task completion
- N := length of prediction horizon follower
- stepsize := integration stepsize of follower solver
- frequency := clock frequency of follower
- horizon := size of the horizon of follower, N*stepsize

TABLE III: Results regarding different follower solver settings.

experiment	score (a)	success (a)	lateral error (a)	duration (a)	score (b)	success (b)	lateral error (b)	duration (b)	N	stepsize	frequency	horizon
0	1	48 %	0.36 m	23.7 s	1	76 %	0.14 m	16.3 s	-	-	-	-
1	1.72	61 %	0.33 m	25.4 s	4.65	94 %	0.40 m	18.0 s	12.0	0.04 s	100.0 Hz	0.48 s
2	1.61	57 %	0.31 m	25.3 s	46.09	95 %	0.04 m	17.7 s	25.0	0.02 s	100.0 Hz	0.50 s
3	0.09	33 %	1.70 m	27.8 s	45.79	90 %	0.04 m	18.0 s	12.0	0.02 s	50.0 Hz	0.24 s
4	1	53 %	0.46 m	24.0 s	5.04	98 %	0.41 m	17.9 s	12.0	0.04 s	50.0 Hz	0.48 s
5	0.47	37 %	0.44 m	24.9 s	3.27	77 %	0.39 m	17.9 s	25.0	0.04 s	50.0 Hz	1.00 s
6	0.9	46 %	0.37 m	23.5 s	11.37	87 %	0.15 m	17.2 s	20.0	0.03 s	100.0 Hz	0.60 s
7	0.5	37 %	0.44 m	23.3 s	9.1	79 %	0.15 m	17.2 s	20.0	0.03 s	50.0 Hz	0.60 s
8	0.18	24 %	0.46 m	26.3 s	3.57	75 %	0.34 m	17.9 s	20.0	0.04 s	100.0 Hz	0.80 s
9	1.16	50 %	0.35 m	24.1 s	29.63	96 %	0.07 m	17.7 s	25.0	0.02 s	50.0 Hz	0.50 s
10	2.23	59 %	0.25 m	24.1 s	28.15	91 %	0.06 m	17.5 s	15.0	0.03 s	100.0 Hz	0.45 s
11	1.17	58 %	0.48 m	23.1 s	22.46	93 %	0.08 m	17.6 s	15.0	0.03 s	50.0 Hz	0.45 s
12	0.11	29 %	1.06 m	27.7 s	5.73	88 %	0.30 m	17.5 s	8.0	0.06 s	50.0 Hz	0.48 s
13	0.21	34 %	0.80 m	26.0 s	5.35	88 %	0.31 m	18.0 s	8.0	0.06 s	100.0 Hz	0.48 s
14	1.77	55 %	0.28 m	23.4 s	33.37	95 %	0.06 m	17.8 s	20.0	0.025 s	100.0 Hz	0.50 s
15	1.06	44 %	0.30 m	22.8 s	30.16	92 %	0.06 m	17.7 s	20.0	0.025 s	50.0 Hz	0.50 s
16	0.63	33 %	0.28 m	23.2 s	4.77	86 %	0.35 m	17.2 s	15.0	0.04 s	100.0 Hz	0.60 s
17	1.05	47 %	0.31 m	25.2 s	3.59	83 %	0.43 m	17.3 s	15.0	0.04 s	50.0 Hz	0.60 s
18	0.24	47 %	1.27 m	26.8 s	55.93	79 %	0.02 m	18.0 s	15.0	0.02 s	100.0 Hz	0.30 s
19	0.25	45 %	1.08 m	27.5 s	25.1	75 %	0.05 m	18.2 s	15.0	0.02 s	50.0 Hz	0.30 s
20	0.05	22 %	1.41 m	27.3 s	3.71	84 %	0.41 m	17.6 s	15.0	0.05 s	50.0 Hz	0.75 s
21	0.47	51 %	0.85 m	25.4 s	26.92	76 %	0.04 m	18.4 s	10.0	0.03 s	100.0 Hz	0.30 s
22	2.08	71 %	0.38 m	24.3 s	82.47	79 %	0.02 m	18.0 s	20.0	0.02 s	100.0 Hz	0.40 s
23	0.27	35 %	0.69 m	24.8 s	5.65	73 %	0.20 m	17.9 s	25.0	0.03 s	100.0 Hz	0.75 s
24	1.03	63 %	0.59 m	25.0 s	23.11	99 %	0.09 m	18.4 s	12.0	0.03 s	100.0 Hz	0.36 s
25	0.23	34 %	0.80 m	23.4 s	3.82	94 %	0.42 m	20.8 s	12.0	0.05 s	100.0 Hz	0.60 s
26	0.05	24 %	1.41 m	32.4 s	2.46	69 %	0.29 m	26.1 s	20.0	0.05 s	100.0 Hz	1.00 s
27	0.09	28 %	1.19 m	27.2 s	3.35	90 %	0.42 m	21.9 s	15.0	0.05 s	100.0 Hz	0.75 s
28	0.08	30 %	1.56 m	28.7 s	9.62	97 %	0.21 m	18.0 s	12.0	0.02 s	100.0 Hz	0.24 s
29	1.41	58 %	0.37 m	25.1 s	12.79	99 %	0.16 m	18.7 s	10.0	0.04 s	100.0 Hz	0.40 s
30	0.25	45 %	1.08 m	27.9 s	5.19	77 %	0.17 m	26.2 s	20.0	0.015 s	100.0 Hz	0.30 s

Appendix B: Lateral deviation and task duration results

In this appendix, Figs. 12 and 13 are given to visualize the consistency in lateral deviation from the route and task duration per experiment regarding the experiment visualized in Fig. 7. All experiments consisted of 100 runs and their performance is sorted on the mean of the corresponding results. The green triangle represents the mean, the orange line represents the median and the grey area above the boxplot represents the spread of the data.

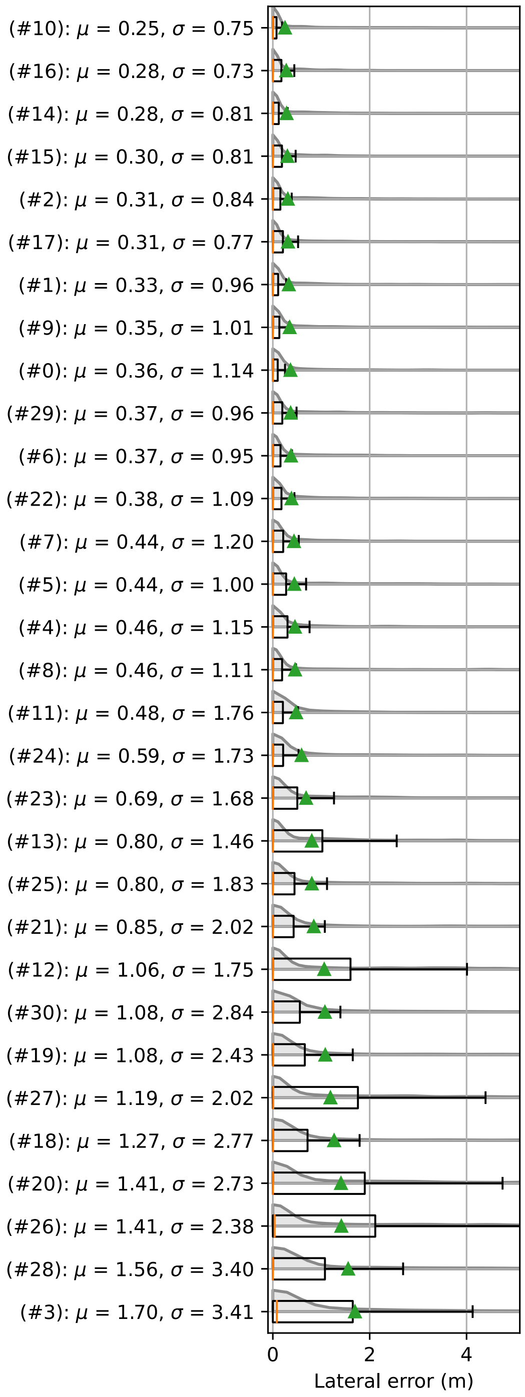


Fig. 12: Boxplots of lateral errors regarding the experiments.

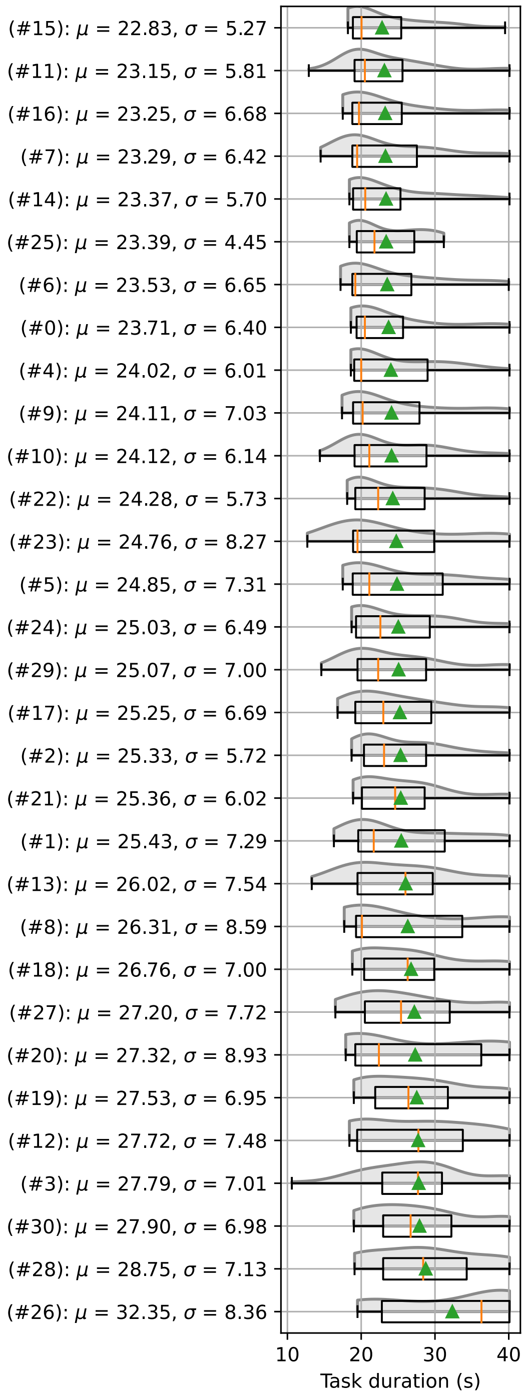


Fig. 13: Boxplots of durations regarding the experiments.

APPENDIX C: A RUN OF EXPERIMENT *#22*

In this appendix, Figs. 14, 15, 16, 17, 18, 19 show several key parts of a run of an experiment with settings *#22*. A video of this run can be watched via [43].

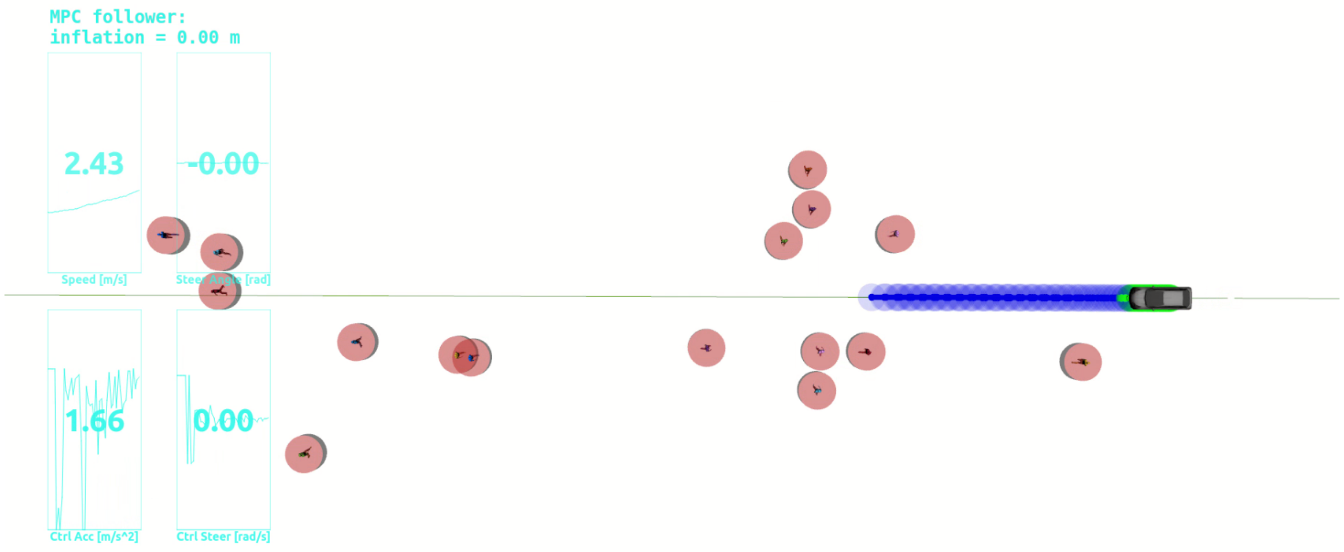


Fig. 14: $t = 0s$: *Start*. Here we see the ego-vehicle at the start of the experiment, accelerating to cruising speed.

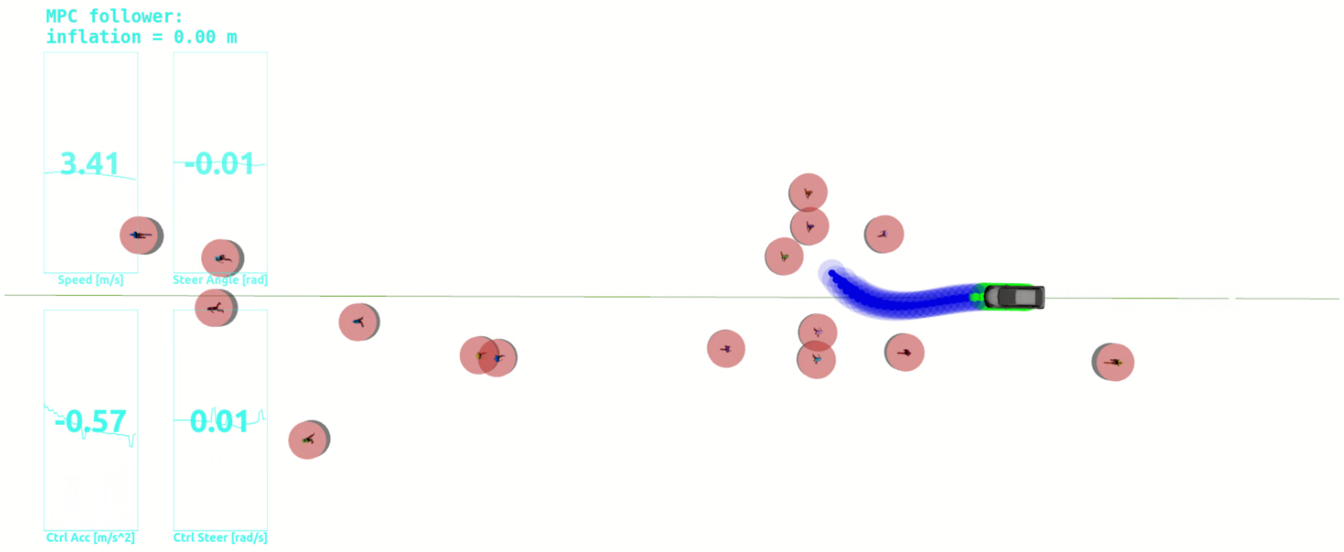


Fig. 15: $t = 4s$: *Brake*. Here we see the ego-vehicle braking to evade the pedestrians.

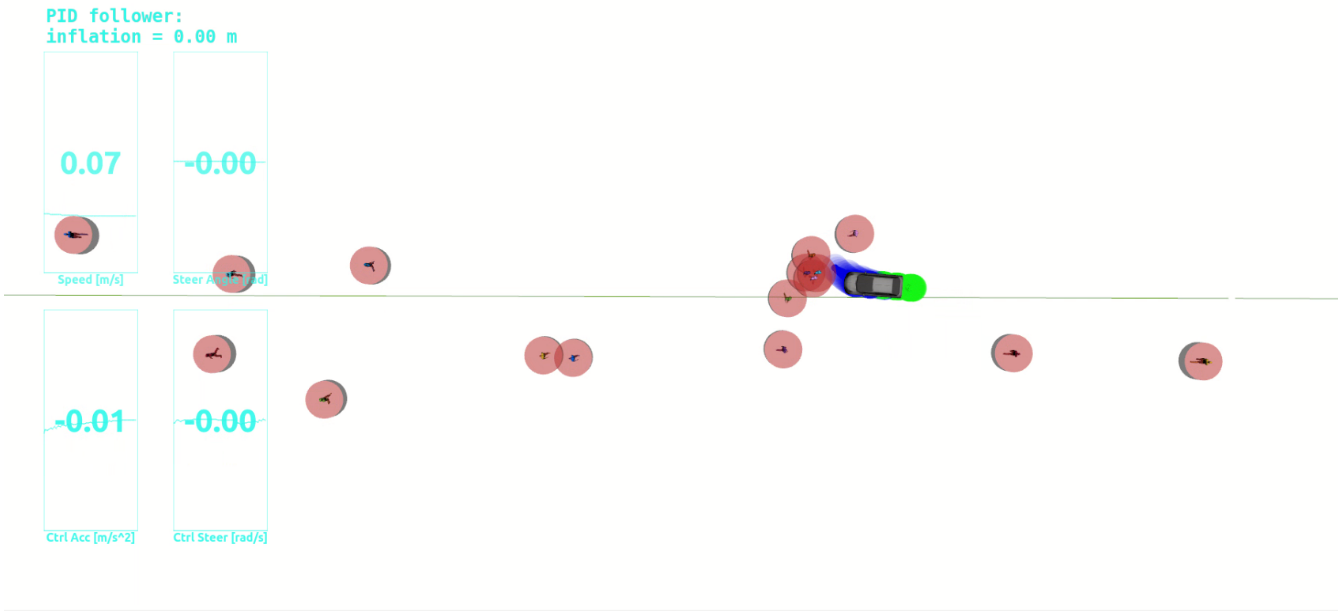


Fig. 16: $t = 12s$ *Wait*. Here we see the ego-vehicle waiting for the pedestrians to pass.

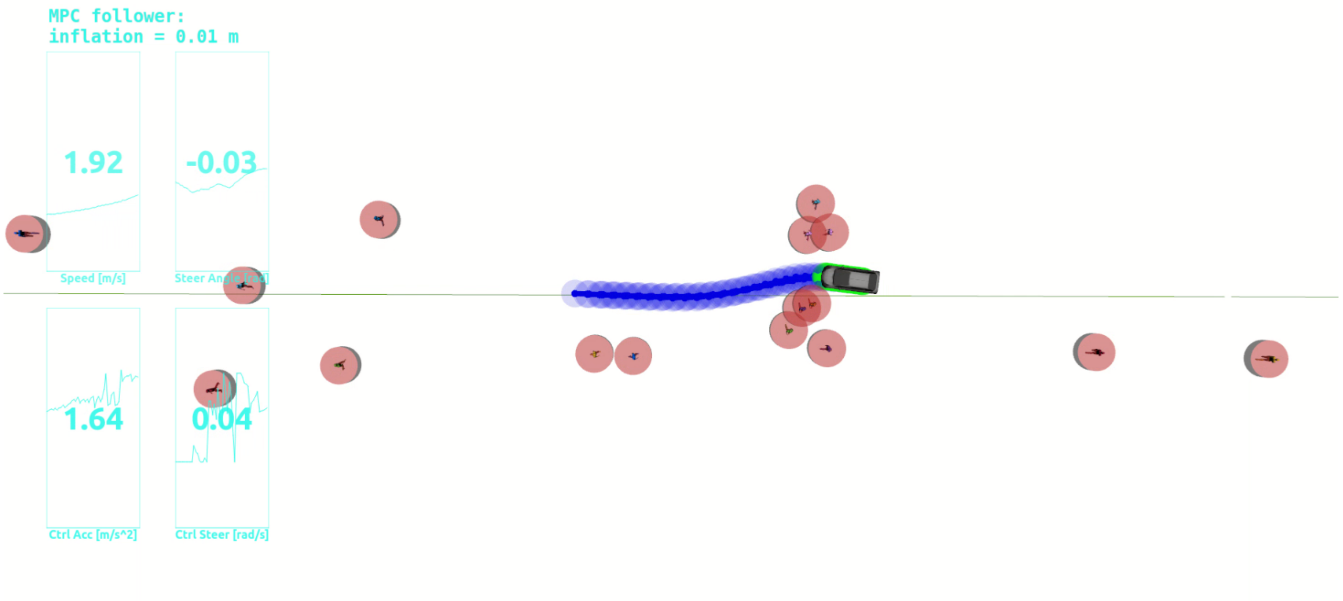


Fig. 17: $t = 19s$ *Continue*. Here we see the ego-vehicle accelerating to continue driving the route.

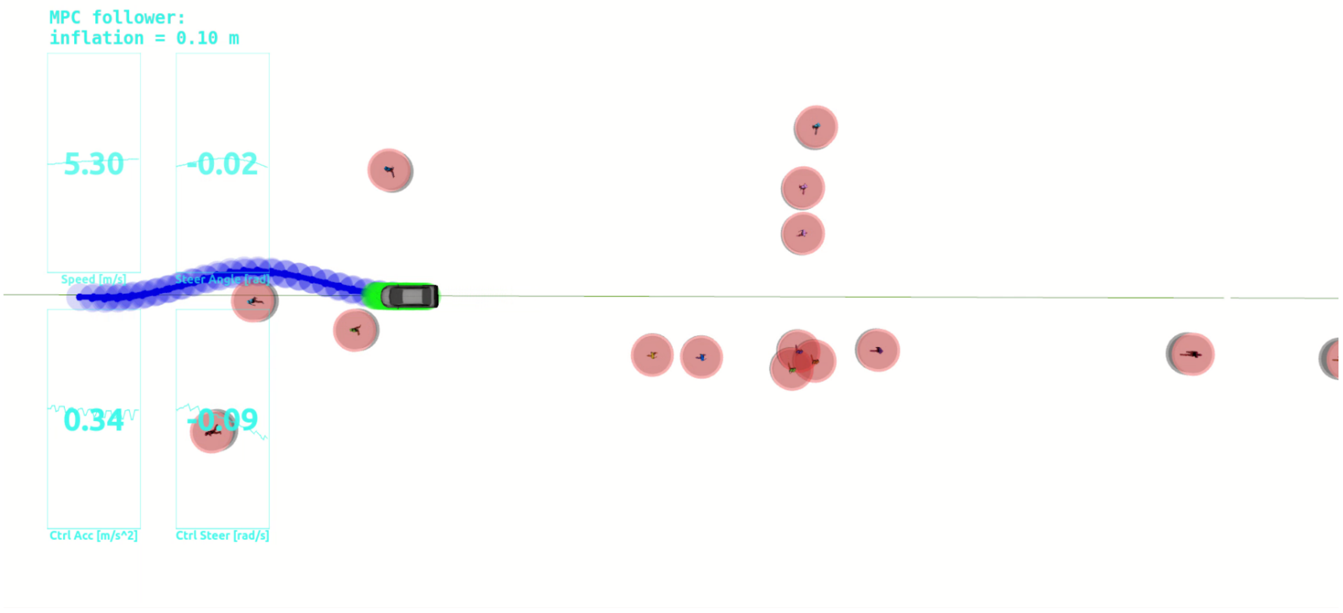


Fig. 18: $t = 27s$ *Inflation*. Here we see the ego-vehicle planning around a pedestrian. Due to planner-follower mismatch, the obstacles are inflated with $\epsilon = 0.1m$.

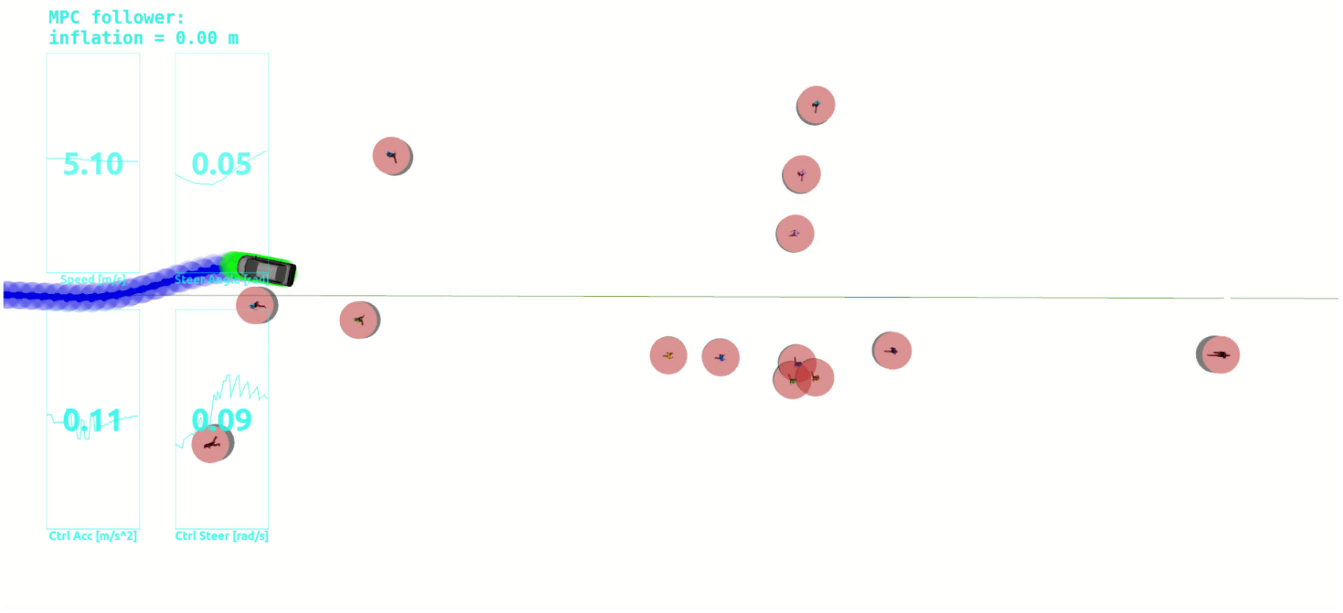


Fig. 19: $t = 29s$ *Evasion*. Here we see the ego-vehicle evading the pedestrian effectively, inflation has returned to $\epsilon = 0m$.

4 Discussion

Automated driving and autonomous cars have emerged as groundbreaking technological advancements that promise to revolutionize the transportation industry. The benefits of this innovation include improved road safety, increased efficiency, and enhanced mobility. This potential increase in safety not only saves lives, but also reduces the strain on healthcare systems and decreases the economic burden associated with traffic accidents.

However, there are concerns that must be addressed to ensure the well-being of pedestrians in an automated driving landscape. Firstly, there is a need to consider the human aspect of pedestrian-vehicle interactions. Pedestrians often rely on visual and auditory cues from drivers to anticipate their behavior and make informed decisions while crossing the road. Autonomous vehicles lack the human characteristics that pedestrians are accustomed to, making it challenging for pedestrians to accurately predict their movements. This unpredictability can deteriorate confidence in the technology and lead to a decrease in pedestrian comfort and safety. Addressing this issue requires the development of clear and intuitive communication methods, such as visual indicators or audible signals.

Another primary concern revolves around the issue of liability and responsibility. In the event of an accident involving an autonomous vehicle, determining who should be held accountable can be complex. Should it be the manufacturer of the technology, the vehicle owner, or a combination of both? Also, if the autonomous car had to choose between saving the life of a child on the road, or its passenger, what should it do? These legal and ethical dilemmas pose significant challenges that need to be addressed to ensure a fair and just system.

Additionally, vulnerable road users, such as children, elderly individuals, or individuals with disabilities, require special attention in the context of autonomous cars. These groups may face unique challenges when navigating the road, and it is crucial to ensure that the technology accounts for their specific needs. For instance, children may behave unpredictably, and individuals with visual impairments may rely heavily on auditory cues. Considering the diverse range of pedestrian behaviors and abilities is essential in designing autonomous systems that prioritize the safety and inclusion of all road users.

Lastly, the reliance on complex technology also introduces cybersecurity risks. As autonomous cars become increasingly connected, they become potential targets for cyberattacks. A breach in the system could have severe consequences, ranging from unauthorized access to personal data to potential physical harm if control of the vehicle is compromised. The development of robust cybersecurity measures and protocols is crucial to safeguard the integrity and security of these autonomous systems.

In conclusion, while automated driving and autonomous cars offer significant potential benefits for pedestrian safety, there are specific concerns that need to be addressed to ensure the well-being of vulnerable road users. By developing technology that accurately interprets and responds to pedestrian cues, establishing clear and intuitive communication methods, and accounting for the diverse needs of pedestrians, a transportation system that prioritizes the safety and inclusivity of all road users can be created.

References

[1] World Health Organization. *Road traffic injuries*. 2022. URL: `https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries` (visited on 11/15/2022).

[2] Brian Paden et al. “A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles”. In: *IEEE Transactions on Intelligent Vehicles* 1.1 (Mar. 2016). Conference Name: IEEE Transactions on Intelligent Vehicles, pp. 33–55. ISSN: 2379-8904. DOI: `10.1109/TIV.2016.2578706`.

[3] Alexandru Constantin Serban, Erik Poll, and Joost Visser. “A Standard Driven Software Architecture for Fully Autonomous Vehicles”. In: *2018 IEEE International Conference on Software Architecture Companion (ICSA-C)*. 2018 IEEE International Conference on Software Architecture Companion (ICSA-C). Seattle, WA: IEEE, Apr. 2018, pp. 120–127. ISBN: 978-1-5386-6585-5. DOI: `10.1109/ICSA-C.2018.00040`. URL: `https://ieeexplore.ieee.org/document/8432195/` (visited on 01/08/2023).

[4] SAE International. *SAE International Releases Updated Visual Chart for Its “Levels of Driving Automation” Standard for Self-Driving Vehicles*. 2018. URL: `https://www.sae.org/news/press-room/2018/12/sae-international-releases-updated-visual-chart-for-its-%E2%80%9Clevels-of-driving-automation%E2%80%9D-standard-for-self-driving-vehicles` (visited on 07/08/2023).

[5] Claudine Badue et al. “Self-driving cars: A survey”. In: *Expert Systems with Applications* 165 (Mar. 2021), p. 113816. ISSN: 09574174. DOI: `10.1016/j.eswa.2020.113816`. URL: `https://linkinghub.elsevier.com/retrieve/pii/S095741742030628X` (visited on 10/24/2022).

[6] L. Ferranti et al. “SafeVRU: A Research Platform for the Interaction of Self-Driving Vehicles with Vulnerable Road Users”. In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. 2019 IEEE Intelligent Vehicles Symposium (IV). Paris, France: IEEE, June 2019, pp. 1660–1666. ISBN: 978-1-72810-560-4. DOI: `10.1109/IVS.2019.8813899`. URL: `https://ieeexplore.ieee.org/document/8813899/` (visited on 09/19/2022).

[7] Claudio Molteni, Francesco Braghin, and Laura Ferranti. “Advanced NMPC-based Local Motion Planner for Autonomous Vehicles”. Master Thesis. Milan: POLITECNICO DI MILANO, Apr. 5, 2021.

[8] Morgan Quigley et al. “ROS: An open-source robot operating system”. In: *Workshops at the IEEE International Conference on Robotics and Automation*. 2009.

[9] Reliable Robot Control Lab. *Website*. 2023. URL: `https://r2clab.com/` (visited on 07/08/2023).

[10] James B. Rawlings, David Q. Mayne, and Moritz M. Diehl. *Model Predictive Control: Theory, Computation, and Design*. Vol. 2. 2020.

[11] David González et al. “A Review of Motion Planning Techniques for Automated Vehicles”. In: *IEEE Transactions on Intelligent Transportation Systems* 17.4 (Apr. 2016). Conference Name: IEEE Transactions on Intelligent Transportation Systems, pp. 1135–1145. ISSN: 1558-0016. DOI: `10.1109/TITS.2015.2498841`.

[12] Jacob Mattingley, Yang Wang, and Stephen Boyd. “Receding Horizon Control”. In: *IEEE Control Systems Magazine* 31.3 (June 2011). Conference Name: IEEE Control Systems Magazine, pp. 52–65. ISSN: 1941-000X. DOI: `10.1109/MCS.2011.940571`.

[13] Martin Behrendt Wikipedia contributors. *File:MPC scheme basic.svg - Wikipedia*. Oct. 2009. URL: `https://en.wikipedia.org/wiki/File:MPC_scheme_basic.svg` (visited on 12/20/2022).

[14] Bruno Brito et al. “Model Predictive Contouring Control for Collision Avoidance in Unstructured Dynamic Environments”. In: *IEEE Robotics and Automation Letters* 4.4 (Oct. 2019). Conference Name: IEEE Robotics and Automation Letters, pp. 4459–4466. ISSN: 2377-3766. DOI: `10.1109/LRA.2019.2929976`.

[15] A. Zanelli et al. “FORCES NLP: an efficient implementation of interior-point methods for multi-stage nonlinear nonconvex programs”. In: *International Journal of Control* 93.1 (May 22, 2017). Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/00207179.2017.1316017, pp. 13–29. ISSN: 0020-7179. DOI: `10.1080/00207179.2017.1316017`. URL: `https://doi.org/10.1080/00207179.2017.1316017` (visited on 07/03/2023).

[16] Scipy documentation. *Integration (scipy.integrate)*. 2023. URL: `https://docs.scipy.org/doc/scipy/tutorial/integrate.html` (visited on 07/08/2023).

[17] Lawrence F Shampine. “Some Practical Runge-Kutta Formulas”. In: *MATHEMATICS OF COMPUTATION* 46.173 (1986), pp. 135–150.