



**Politecnico
di Torino**

ScuDo
Scuola di Dottorato - Doctoral School
WHAT YOU ARE, TAKES YOU FAR

Doctoral Dissertation

Doctoral Program in Mechanical Engineering (37th cycle)

Laboratory car testbed for single and multi-agent control: Testing of autonomous driving control algorithms using scaled vehicles

By

Eugenio Tramacere, M.Sc.

Supervisor(s):

Prof. Nicola Amati

PhD. Luis Miguel Castellanos Molina

Doctoral Examination Committee:

Prof. Umberto Montanaro (University of Surrey, Guildford, UK)

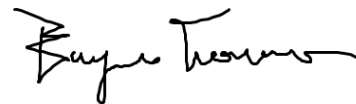
Prof. Sanjarbek Ruzimov (Kimyo International University in Toshkent, Toshkent, Uzbekistan)

Politecnico di Torino

9th July 2025

The thesis is licensed under Creative Common License, Attribution - Noncommercial - NoDerivative Works 4.0 International: see www.creativecommons.org. The text may be reproduced for non-commercial purposes, provided that credit is given to the original author.

I hereby declare that, the contents and organization of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

A handwritten signature in black ink, appearing to read 'Eugenio Tramacere', with a stylized, cursive script.

Eugenio Tramacere, M.Sc.
Turin, 9th July 2025

* This dissertation is presented in partial fulfillment of the requirements for **Ph.D. degree** in the Graduate School of Politecnico di Torino (ScuDo).

Abstract

This dissertation presents the design, implementation, and experimental validation of a novel scaled-vehicle testbed tailored for the development and real-time testing of autonomous driving algorithms. The framework is based on 1:10 scale remote-controlled (RC) cars, enhanced with custom embedded electronics, wireless communication, and a vision-based indoor localization system. The testbed is engineered to closely replicate the dynamic behavior of full-scale vehicles, ensuring that controllers developed in laboratory-scaled environments can be reliably validated under real-time constraints.

A modular system architecture is introduced. The platform is fully compatible with MATLAB/Simulink workflows, enabling automatic code generation and deployment on real-time Electronic Control Units (ECUs) via Controller Area Network (CAN) communication. This facilitates rapid prototyping and evaluation of advanced control algorithms with minimal hardware integration effort.

A key contribution of this work is the integration of nonlinear model predictive control (NMPC) strategies using the open-source Acados solver. This allows real-time execution of optimization-based controllers, which are evaluated in various path-following and collision avoidance (CA) scenarios. Three distinct CA strategies are developed and benchmarked: (1) a heuristic sigmoid-based reference generator, (2) a nonlinear constraint approach based on a circular safety region, and (3) a probabilistic approach employing a stage-varying Gaussian confidence bound. The probabilistic method, which dynamically adjusts the safety margins during the prediction horizon, consistently demonstrated superior performance in terms of trajectory smoothness, dynamic feasibility, and safety margins, particularly in dynamic obstacle scenarios with respect to the other considered CA strategies.

Extensive experiments validate the effectiveness of the proposed architecture, showcasing its ability to bridge the gap between theoretical controller design and

practical deployment. The testbed supports both single-vehicle and multi-agent experiments, making it suitable for research on cooperative and distributed driving strategies.

This thesis bridges the gap between theoretical control formulations and practical deployment by offering a reproducible, scalable, and industry-compliant methodology for testing autonomous driving solutions in controlled environments. The insights gained contribute toward the broader objective of safe and reliable autonomous mobility, particularly in constrained or cost-sensitive development phases.

*To my family,
to my friends*

Contents

List of Figures	x
List of Tables	xv
Nomenclature	xviii
1 Introduction	1
1.1 Objectives	3
1.1.1 Scientific Contributions	3
1.1.2 Practical Contributions	3
1.2 Thesis outline	5
2 Preliminaries and related works	6
2.1 Introduction	6
2.2 Small-scale tesbeds	7
2.3 Characteristics of testbeds	10
2.3.1 General characteristics	10
2.3.2 Sense-driven characteristics	12
2.3.3 Plan-driven characteristics	13
2.3.4 Act-driven characteristics	15
2.4 CARS facilities	16

3	CARS Autonomous Scaled-Vehicle Testbed	22
3.1	Testing scenarios	22
3.2	Miniature RC cars	23
3.3	Vision based localization system	25
3.4	Controller PC	25
3.5	Overview	27
4	Vehicle model	30
4.1	System Identification	30
4.1.1	Static Parameters	31
4.2	Longitudinal dynamics	33
4.3	Lateral dynamics	40
4.4	Dynamics Similarity	46
5	Car motion control problem	48
5.1	Problem formulation	49
5.1.1	Racing-oriented optimal control formulation	56
5.1.2	Passenger-car-oriented optimal control formulation	60
5.2	Collision Avoidance	63
5.2.1	Reference lateral deviation generator logic	65
5.2.2	Obstacle nonlinear constraint with circle-safe region	67
5.2.3	Obstacle nonlinear constraint with probabilistic region	68
5.3	Real-time implementation	72
5.3.1	Acados	72
5.3.2	Implementation	74
6	Experimental tests	76
6.1	Sensor	76

6.1.1	Localization systems	77
6.1.2	Inertial Measurement Unit	78
6.2	State estimators	78
6.2.1	Velocity estimator	79
6.2.2	Kinematic Kalman Filter	79
6.2.3	Outlier rejection	81
6.3	Control	84
6.3.1	TEST control	86
6.3.2	STANLEY control	86
6.3.3	PURE PURSUIT control	87
6.3.4	MPC control	89
6.4	Low-level control	89
6.4.1	Speed-based control	91
6.4.2	Acceleration-based control	92
6.5	Plant	94
7	Experimental results and Discussion	97
7.1	Reference lateral deviation generator logic	98
7.2	Obstacle nonlinear constraint with circle-safe region	108
7.3	Obstacle nonlinear constraint with probabilistic region	116
8	Conclusions	125
8.1	Outlook	127
	References	129
	Appendix A Survey of small testbed	137

List of Figures

2.1	Examples of existent small-scale tesbeds available in the present literature.	9
2.2	Costs associated to early, mid and final development stage of autonomous vehicle control alogrithms.	17
2.3	V-cycle for autonomous vehicle control algorithms.	18
2.4	CARS facilities	19
2.5	Progressive screening of the scenario to test: from virtual simulation to test on the field.	19
3.1	CARS Autonomous Scaled-Vehicle Testbed	23
3.2	1:10 scale vehicle with custom modifications	24
3.3	Schematic drawing of the two different marker patterns used to identify the scaled cars	26
3.4	Cascade time delays throughout the entire communication pipeline .	27
3.5	Schematic overview of the CARS Autonomous Scaled-Vehicle Testbed	27
4.1	Scaled-vehicle vehicle distribution test	31
4.2	Moment of inertia measurement	32
4.3	Longitudinal vehicle model	34
4.4	Modeling vehicle longitudinal dynamics response	35
4.5	Modeling vehicle longitudinal dynamics response	40

4.6	Dynamic single-track model	41
4.7	Experimental lateral forces vs Pacejka simplified model	43
4.8	Parameter identification of the simplified lateral dynamics model . .	45
5.1	General control scheme of the scaled-vehicle.	48
5.2	Curvilinear prediction model	51
5.3	Kamm's friction circle	54
5.4	States and control commands evolution along the circuit	59
5.5	Racing-oriented motion control behaviour	60
5.6	States and control commands evolution along the circuit	62
5.7	Passenger-car-oriented motion control behaviour	63
5.8	Reference lateral deviation generator logic	65
5.9	ISO 3888-2 layout for scaled vehicle.	66
5.10	Obstacle constraint using circle safe-region	67
5.11	Obstacle constraint using probabilistic region	70
5.12	Obstacle constraint using probabilistic region with dynamic con- straint upper bound definition	71
6.1	Control pipeline for experimental tests. The proposed framework can be used for both simulations and hardware experiments.	77
6.2	KF with covariance threshold outlier rejection method	82
6.3	KF with Chi-squared rejection method	84
6.4	Vehicle control state machine	85
6.5	Stanley controller kinematic model	86
6.6	Pure Pursuit controller model	88
6.7	Low level control state machine	90
6.8	FB velocity control	92
6.9	Acceleration-based control scheme	92

6.10	Acceleration control with FB and GSFF control	94
6.11	Implementation of the feedback linearization logic.	95
7.1	Experimental layout of the circuit	98
7.2	States evolution along the maneuver. Collision avoidance with static obstacle using Reference lateral deviation generator logic	99
7.3	Commands, vehicle accelerations and poses evolution along the maneuver. Collision avoidance with Reference lateral deviation generator logic behaviour with static obstacle	101
7.4	States evolution along the maneuver. Collision avoidance with dynamic obstacle using Reference lateral deviation generator logic . .	102
7.5	Commands, vehicle accelerations and poses evolution along the maneuver. Collision avoidance with Reference lateral deviation generator logic behaviour with moving obstacle	103
7.6	States evolution along the maneuver. Collision avoidance with static obstacle using Reference lateral deviation generator logic. Experimental results are reported using travelled distance on the x-axis. . .	104
7.7	Commands control actions evolution along the maneuver. CA with static obstacle using Reference lateral deviation generator logic. Experimental results are reported using travelled distance on the x-axis.	105
7.8	States evolution along the maneuver. Collision avoidance with dynamic obstacle using Reference lateral deviation generator logic. Experimental results are reported using travelled distance on the x-axis.	106
7.9	Commands control actions evolution along the maneuver. CA with dynamic obstacle using Reference lateral deviation generator logic. Experimental results are reported using travelled distance on the x-axis.	107
7.10	States evolution along the maneuver. Collision avoidance with static obstacle using circle safe region	108
7.11	Commands, vehicle accelerations and poses evolution along the maneuver. Collision avoidance with circle safe region logic behaviour with static obstacle	109

7.12	States evolution along the maneuver. Collision avoidance with dynamic obstacle using circle safe region	110
7.13	Commands, vehicle accelerations and poses evolution along the maneuver. Collision avoidance with circle safe region logic behaviour with moving obstacle	111
7.14	States evolution along the maneuver. Collision avoidance with static obstacle using circle safe region logic. Experimental results are reported using travelled distance on the x-axis.	112
7.15	Commands control actions evolution along the maneuver. CA with static obstacle using circle safe region logic. Experimental results are reported using travelled distance on the x-axis.	113
7.16	States evolution along the maneuver. Collision avoidance with dynamic obstacle using circle safe region logic. Experimental results are reported using travelled distance on the x-axis.	114
7.17	Commands control actions evolution along the maneuver. CA with dynamic obstacle using circle safe region logic. Experimental results are reported using travelled distance on the x-axis.	115
7.18	States evolution along the maneuver. Collision avoidance with static obstacle using Gaussian safe region	116
7.19	Commands, vehicle accelerations and poses evolution along the maneuver. Collision avoidance with Gaussian safe region logic behaviour with static obstacle	117
7.20	States evolution along the maneuver. Collision avoidance with dynamic obstacle using Gaussian safe region	118
7.21	Commands, vehicle accelerations and poses evolution along the maneuver. Collision avoidance with Gaussian safe region logic behaviour with dynamic obstacle	119
7.22	States evolution along the maneuver. Collision avoidance with static obstacle using Gaussian safe region logic. Experimental results are reported using travelled distance on the x-axis.	120

- 7.23 Commands control actions evolution along the maneuver. CA with static obstacle using Gaussian safe region logic. Experimental results are reported using travelled distance on the x-axis. 121
- 7.24 States evolution along the maneuver. Collision avoidance with dynamic obstacle using Gaussian safe region logic. Experimental results are reported using travelled distance on the x-axis. 122
- 7.25 Commands control actions evolution along the maneuver. CA with dynamic obstacle using Gaussian safe region logic. Experimental results are reported using travelled distance on the x-axis. 123

List of Tables

3.1	Comparison of laboratory scaled-car testbed based on selected characteristics.	29
4.1	Scaled-vehicle weight distribution	31
4.2	Yaw moment of inertia I_z for different string heights h	32
4.3	Summary of static parameters of the scale-car	33
4.4	Parameters defining the longitudinal dynamics of the scaled car using non-linear grey-box identification method	35
4.5	Parameters defining the longitudinal dynamics of the scaled car using non-linear grey-box identification method	40
4.6	Identified Pacejka parameters for front and rear tires	44
4.7	Parameters defining the single-track vehicle model and their dimensions.	46
5.1	ISO 3888-2 Double lane change maneuver. Comparison between full scale and 1:10 scale	66
6.1	Performance indicators for proposed outlier rejection methods . . .	84
7.1	CA strategy comparison for static obstacle scenario	124
7.2	CA strategy comparison for dynamic obstacle scenario	124
A.1	CARS Autonomous Scaled-Vehicle Testbed: General, Sense and Plan characteristics	138

A.2	CARS Autonomous Scaled-Vehicle Testbed: Act characteristics . .	139
A.3	CPM Lab Testbed: General, Sense and Plan characteristics	140
A.4	CPM Lab Testbed: Act characteristics	141
A.5	FITENTH: General, Sense and Plan characteristics	142
A.6	FITENTH: Act characteristics	143
A.7	Robotarium: General, Sense and Plan characteristics	144
A.8	Robotarium: Act characteristics	145
A.9	IDS3C(UDSSC): General, Sense and Plan characteristics	146
A.10	IDS3C(UDSSC): Act characteristics	147
A.11	Cambridge Minicar: General, Sense and Plan characteristics	148
A.12	Cambridge Minicar: Act characteristics	149
A.13	Duckietown: General, Sense and Plan characteristics	150
A.14	Duckietown: Act characteristics	151
A.15	CHARTOPOLIS: General, Sense and Plan characteristics	152
A.16	CHARTOPOLIS: Act characteristics	153
A.17	Cambridge RoboMaster: General, Sense and Plan characteristics . .	154
A.18	Cambridge RoboMaster: Act characteristics	155
A.19	Go-Chart: General, Sense and Plan characteristics	156
A.20	Go-Chart: Act characteristics	157
A.21	CRS: General, Sense and Plan characteristics	158
A.22	CRS: Act characteristics	159
A.23	MiniCity: General, Sense and Plan characteristics	160
A.24	MiniCity: Act characteristics	161
A.25	Optimal RC racing (ORCA): General, Sense and Plan characteristics	162
A.26	Optimal RC racing (ORCA): Act characteristics	163
A.27	Miniature Autonomy: General, Sense and Plan characteristics . . .	164

A.28 Miniature Autonomy: Act characteristics	165
A.29 Self-Driving Car Studio: General, Sense and Plan characteristics . .	166
A.30 Self-Driving Car Studio: Act characteristics	167

Nomenclature

Greek Symbols

β_{obst} Confidence bound of Gaussian distribution with probabilistic safe region

δ Steering angle

$\Delta\delta$ Variation of steering angle command

δ_{kin} Kinematic steering angle

δ_{limit} Maximum steering angle

δ_{max} Maximum steering angle

δ_{min} Minimum steering angle

$\dot{\phi}_I$ Roll angle velocity measured by the IMU

$\dot{\psi}_I$ Yaw angle velocity measured by the IMU

$\dot{\theta}_I$ Pitch angle velocity measured by the IMU

$\hat{\omega}$ Estimated yaw velocity

$\hat{\Psi}$ Estimated yaw angle

$\hat{v}_x^w$ Estimated longitudinal velocity

$\hat{v}_y^w$ Estimated lateral velocity

$\hat{X}$ Estimated X position

$\hat{Y}$ Estimated Y position

κ_r	Curvature at reference path arc length Frenet coordinate
ω	Motor speed
ω	Yaw rate
ψ	Yaw angle
ψ_e	Relative heading angle between vehicle and reference path
Ψ_G	Yaw angle measured by the motion system
ψ_r	Reference path heading angle
ψ_v	Scaled car heading angle
ψ_{dev}	Deviation from reference heading angle
σ_n	Standard deviation along the lateral deviation Frenet coordinate
σ_q	Process noise standard deviation
σ_s	Standard deviation along the progress Frenet coordinate
σ_X	Measurement noise standard deviation for position x
σ_Y	Measurement noise standard deviation for position y
σ_Ψ	Measurement noise standard deviation for orientation
τ_i	Integral time constant for the Proportional-Integral control
τ_m	DC motor torque
τ_{brake}	Braking torque
τ_{load}	Load torque
X	Global position along x direction
X_G	X coordinate measured by localization system
x_{ref}	Reference state trajectory
Y	Global position along y direction

Y_G Y coordinate measured by localization system

Relaxation factor of probabilistic dynamic upper obstacle constraint bound

Other Symbols

$\overline{h_{a_{x,y}}}$ Upper bound on non-linear acceleration constraint

$\overline{h_{a_{x,y}}}$ Upper bound on non-linear obstacle constraint

$\overline{h_{fin}}$ confidence bound of Gaussian probabilistic safe region at final prediction step

$\overline{h_{init}}$ confidence bound of Gaussian probabilistic safe region at initial prediction step

$\overline{c_x}$ Upper bound on state constraints

$\underline{h_{a_{x,y}}}$ Lower bound on non-linear acceleration constraint

$\underline{h_{obst}}$ Lower bound on non-linear obstacle constraint

$\underline{c_x}$ Lower bound on state constraints

A_σ Desired lateral deviation amplitude

A_{obst} Amplitude of the obstacle probabilistic safe region

$a_{x,target}$ Longitudinal requested acceleration

$a_{x,I}$ Longitudinal acceleration measured by the IMU

$a_{x,max}$ Maximum longitudinal acceleration

$a_{y,I}$ Lateral acceleration measured by the IMU

$a_{y,max}$ Maximum lateral acceleration

$a_{z,I}$ Vertical acceleration measured by the IMU

b Distance between the strings for bifilar test

b_f Damping coefficient

$B_{f,r}$ Front/Rear stiffness factor for simplified Paejka

$C_{\alpha f,r}$	Front/Rear tire cornering stiffness
$C_{f,r}$	Front/Rear shape factor for simplified Paejka
CMD_{δ}	Steering servo command output from vehicle control state machine
CMD_{EM}	Electric motor command output from vehicle control state machine
$D_{f,r}$	Front/Rear peak factor for simplified Paejka
$Duty_{FC}$	Duty output value from feedback control
$Duty_{GSFF}$	Duty output value from Gain-scheduling feedforward control
$Duty_{linear}$	Duty command computed assuming linear actuator
$Duty_{out}$	Duty output value from the acceleration-based controller
e	Crosstrack error in the Stanley controller
F_x	Longitudinal tire force
F_{fy}	Lateral front tire force
F_{rx}	Longitudinal rear tire force
F_{ry}	Lateral rear tire force
$F_{x,target}$	Target longitudinal force
g	Gravitational acceleration
h	Length of the strings for bifilar test
i	DC motor current
I_z	Moment of inertia
k	Stepness parameter for logistic collision avoidance function
k_b	DC motor braking constant
K_e	Back-EMF force constant
K_i	Integral gain of the Proportional-Integral control

K_p	Proportional gain of the Proportional-Integral control
K_t	DC motor torque constant
L	DC motor inductance
l_f	Distance CoG to front axle
l_r	Distance CoG to rear axle
l_{ahead}	Lookahead value in the Pure Pursuit controller
L_{obst}	Length of the obstacle
m	Scaled-car weight
n_r	Orthogonal distance in Frenet coordinates
n_{max}	Maximum orthogonal distance from the path in Frenet coordinates
n_{min}	Minimum orthogonal distance from the path in Frenet coordinates
n_{obst}	Lateral deviation of the obstacle
$n_{ref,obst}$	Lateral deviation reference for CA algorithm
N_{steps}	Number of steps
Q	Weight matrix for state tracking over the horizon
Q^e	Weight matrix of terminal cost for state tracking
R	Weight matrix for control effort
R_m	DC motor resistance
R_s	Shaft radius
R_{safe}	Radius of circle-based safe region for nonlinear constraint
R_{total}	Resistances of DC motor and H-bridge circuit
s	Arc length in Frenet coordinate
s_r	Reference path arc length in Frenet coordinate

s_{dev}	Deviation from reference arch length in Frenet coordinate
s_{obst}	Progression of the obstacle along the path
s_{ref}	Reference arc length in Frenet frame
s_{ref}^e	Terminal reference arc length in Frenet coordinate
s_{traj}	Predicted progress
s_{vect}	Progress vector
T	Time horizon
T_s	Sampling time
v	Velocity
V_a	Applied voltage to the DC motor
v_{ch}	Characteristic speed
V_{emf}	Back electromotive force
V_{max}	Maximum DC motor voltage
v_{max}	Maximum vehicle velocity
v_{min}	Minimum vehicle velocity
v_{ref}	Reference velocity
v_{target}	Vehicle velocity reference
v_x	Longitudinal velocity
v_y	Lateral velocity
w_{lane}	Lane width
x_{rel}	Relative longitudinal distance between vehicle and obstacle
y_{rel}	Relative lateral distance between vehicle and obstacle
H	Observation matrix

K	Kalman gain
P	Error covariance matrix
Q	Process noise covariance matrix
R	Measurement noise covariance matrix
S	Innovation covariance matrix
F	State transition matrix
D	Duty cycle of the DC motor
J	DC motor rotational inertia
l	Wheelbase
R	Radius of curvature
T	Time

Acronyms / Abbreviations

4WD	Four Wheel Drive
I2C	Inter-Integrated Circuit communication
CA	Collision Avoidance
CAN	Controller Area Network
CAN	Controller Area Network
CARS	Center for Automotive Research and Sustainable Mobility
CAVs	Connected and Automated Vehicles
CCD	Charge-Coupled Device
CMOS	Complementary Metal-Oxide-Semiconductor
CoG	Vehicle center of gravity
ECU	Electronic Control Unit

EKF	Extended Kalman Filter
ESC	Electronic Speed Controller
FB	Feedback control
FF	Feedforward control
GNSS	Global Navigation Satellite Systems
HIL	Hardware-in-the-Loop
HPIPM	High performance interior point method
IMU	Inertial Measurement Unit
KF	Kalman Filter
LIDAR	Laser Imaging Detection and Ranging
LUT	Lookup Table
MAE	Mean Absolute Error
MAUTO	Museo Nazionale Automobili di Torino
MIL	Model-in-the-Loop
MOST	Centro Nazionale per la Mobilità Sostenibile
MPC	Model Predictive Control
MPCC	Model Predictive Contouring Controller
NLP	Non-linear programming
NMPC	Non-linear Model Predictive Control
OCP	Optimal control problem
PI	Proportional-Integral control
PP	Path Planning
PTC	Path Tracking Control

PWM Pulse Width Modulation

QP Quadratic program

RMSE Root Mean Square Error

RMSE Root Mean Square Error

RRT Rapidly Random Exploring Tree algorithm

RTOS Real-Time Operating System

RTTM Real-Time Target Machine

RWD Rear Wheel Drive

RWD Rear Wheel Drive

SPI Serial Pheripheral Interface

SQP Sequential quadratic programming

SQP-RTI Sequential quadratic programming with real-time iteration

UDP User Datagram Protocol

VIL Vehicle-in-the-Loop

Chapter 1

Introduction

In the rapidly evolving field of Connected and Automated Vehicles (CAVs), research and development have typically relied on either simulations or empirical experiments. Each possesses distinct strengths and weaknesses, creating a multifaceted description of the testing landscape. Simulation environments are crucial for testing and validation; however, they frequently encounter difficulties in accurately replicating real-world experiments [1]. Concerning the difficulties of computational modeling, these simulations fail to accurately represent variables such as fluctuating weather patterns or erratic pedestrian behavior. As they pursue enhanced precision, particularly regarding the complexities of physics, the requirements for computing power and hardware increase considerably. Conversely, real-world experiments, while directly applicable, entail substantial costs and overhead, often rendering them prohibitive for iterative testing and development [2]. One promising way to close the gap between these two methods is to use small-scale testbeds. A controlled setting that strikes a balance between the advantages of simulation and actual experimentation is offered by these testbeds. They provide reproducibility, an essential element of scientific experimentation, while decreasing development and maintenance expenses relative to full-scale testing. Moreover, small-scale testbeds are customized for particular use cases, optimizing the testing process and enhancing its efficiency. The scaled infrastructure presented in this dissertation serves as a testbed for exploring the potential of CAVs. Within this facility, we have 2 cars, each built to a 1:10 scale. They operate in a designated area of $6 \times 5 \text{ m}^2$. Although compact, these vehicles are designed to replicate authentic driving dynamics, particularly utilizing the Ackermann steering mechanism [3]. The scaled vehicles utilize an indoor positioning

system featuring a ceiling-mounted camera as a substitute for conventional Global Navigation Satellite Systems (GNSS) for navigation. The scaled car framework infrastructure accommodates both centralized and distributed computing systems. Additionally, users can create and execute their own test scenarios with programming tools like MATLAB, which can produce the C code needed to deploy the controller on the Real-Time Electronic Control Unit (ECU). Nonetheless, small-scale testbeds present inherent limitations. A significant limitation is the scope of their feature sets. They typically concentrate on particular use cases and accommodate only select programming languages, resulting in deficiencies in their domain coverage. Several proposed solutions include testbeds lacking a vision-based system functioning as GNSS for the vehicle. In these instances, vehicles depend on integrated sensors and control systems to detect their environment. Nevertheless, a significant disadvantage of these solutions is the restriction on the maximum velocity of the scaled vehicles. This is mainly attributable to the environmental data sampling rate, which is generally approximately 10 Hz, thereby limiting the overall functionality of the scaled vehicle. The additional weight from these perception components influences the vehicle's sprung mass, elevates its center of gravity, and ultimately affects its dynamic performance. Moreover, one of the main challenges in transitioning from simulation to real-world applications is achieving real-time performance. While many platforms and solutions focus on developing and testing algorithms, they often lack the capability to ensure real-time execution, which is crucial for real-world deployment. This dissertation addresses this gap by introducing a scaled vehicle framework for both single and multi-agent control, featuring distinct characteristics. At first, our vehicle incorporates customized electronics into its framework without modifying its dynamic characteristics. Our second contribution is an approach that allows for the integration of SIMULINK-based controllers into real-time electronic control units (ECUs). The control unit guarantees real-time performance and interfaces with the scaled vehicle through CAN communication. This enables controllers designed for a 1:10 scale vehicle to be effortlessly adapted for a 1:1 full-scale vehicle. This methodology connects academic research with practical applications, guaranteeing that controllers developed in a simulated context can be directly evaluated and verified on hardware within real-time constraints.

1.1 Objectives

The main objective of this research is to develop a cost-effective, modular, and scalable testbed using 1:10 scale vehicles for testing and validating autonomous driving algorithms. The framework is intended to bridge the gap between simulation environments and full-scale vehicle experimentation, while maintaining compatibility with industry-standard tools and practices. The work aims to support both single- and multi-agent control strategies and to ensure real-time execution of advanced model-based controllers.

1.1.1 Scientific Contributions

This thesis offers several scientific contributions to the field of autonomous vehicle control and experimental validation frameworks:

- *Real-time NMPC implementation:* The integration of nonlinear model predictive control (NMPC) using the Acados framework [4] allows for efficient real-time execution on embedded systems.
- *Novel obstacle avoidance strategies:* The work introduces and benchmarks three distinct collision avoidance (CA) strategies — a heuristic sigmoid-based generator, a circular safety region constraint, and a probabilistic dynamic constraint using Gaussian bounds.
- *Comparative evaluation under scaled realistic conditions:* The testbed enables systematic comparison of different controllers and CA approaches under both static and dynamic obstacle scenarios, providing experimental validation of their effectiveness.
- *Control-oriented modeling:* A detailed vehicle modeling process ensures that both longitudinal and lateral dynamics are preserved, contributing to more accurate control design and validation.

1.1.2 Practical Contributions

The developed platform also provides valuable practical benefits for engineering applications, education, and industry:

- *Low-cost hardware architecture*: A fully functional 1:10 scale car platform with custom electronics (IMU, ESP32 [5]) that replicates real vehicle dynamics while remaining affordable and easy to assemble.
- *Preserved Vehicle Dynamics*: The original vehicle dynamics are preserved by integrating only non-intrusive sensors and electronic components, ensuring that the scaled car's dynamic behavior remains unaltered. In contrast, the use of intrusive sensors and electronics, such as LiDAR, cameras, or onboard GPUs, not only increases the overall mass but also modifies the vehicle's geometry and physics, i.e. vehicle moment of inertia or position shift of the center of mass. These changes compromise the dynamic similarity between the full-scale vehicle and its 1:10 scaled counterpart, potentially undermining the validity of experimental results at the laboratory scale.
- *Real-time compatible control pipeline*: The system supports controller development in MATLAB/Simulink with automatic code generation and deployment to real-time ECUs, enabling fast prototyping and closed-loop testing.
- *Scalability and flexibility*: The modular framework supports centralized and distributed computation, making it suitable for single-vehicle as well as cooperative multi-agent scenarios.
- *Bridging the gap between function development and testing on scaled vehicles*: By leveraging real-time hardware and communication protocols commonly used in the automotive industry, the platform enables seamless function development and testing in a scaled environment. This approach ensures compliance with industry standards while bridging the gap between academic research and real-world applications.
- *User Friendliness*: The platform is user-friendly for individuals with basic electronics knowledge and intermediate programming skills. This is essential for it to become widely adopted in research and educational projects.
- *Teaching autonomous driving concepts*: The proposed framework, tailored for MATLAB/Simulink environment, enables students to effortlessly test their algorithms on a real system. This method allows them to acquire a thorough understanding of the essential principles of autonomous driving systems by empirically evaluating the developed algorithms in scaled real-world conditions.

1.2 Thesis outline

This thesis is structured as follows: Chapter 2 offers an exhaustive analysis of existing small-scale testbeds, emphasizing their significance in Connected and Autonomous Vehicle (CAV) development and highlighting their specific characteristics. Additionally, the Autonomous-vehicle Testlab is introduced, a component of a broader initiative aimed at developing methodologies for validating and testing autonomous driving controllers at the Center for Automotive Research and Sustainable Mobility (CARS) at Politecnico di Torino. Chapter 3 provides a detailed description of the scaled framework, outlining its design principles and capabilities. Chapter 4 presents the vehicle model developed for path planning and decision-making in autonomous vehicle control. In Chapter 5, this model is integrated into a control structure designed for both racing and passenger-car autonomous vehicle applications, where both path planning and tracking are managed using Model Predictive Control (MPC). Three different approaches of Collision Avoidance are presented, namely the Reference lateral deviation generator logic, the obstacle constraint using circle safe region and the obstacle constraint using probabilistic safe region. Chapter 6 presents the modular architecture for testing autonomous vehicle algorithms developed in MATLAB/Simulink environment. The control pipeline includes a plant, which can represent either a simulated vehicle dynamic model or real hardware, such as the scaled vehicle. Sensor data is used both to monitor and to estimate system state. Thus, the controller computes the appropriate control inputs. Each component of the control loop is described in detail, highlighting the specific models and algorithms used. Chapter 7 presents a benchmark analysis of the previous presented obstacle avoidance techniques applied to static and dynamic obstacle avoidance scenarios. This comparative study underscores the advantages of a scaled architecture, which facilitates efficient testing of advanced obstacle avoidance strategies while reducing both time and costs. Chapter 8 conclude the dissertation.

Chapter 2

Preliminaries and related works

2.1 Introduction

In robotics and control development, experimentation is crucial for assessing and validating algorithm performance. Recently, there has been a notable increase in the development and adoption of small-scale testbeds in the domain of CAVs [6]. These specialized testbeds play a vital role in research and experimentation, advancing fields such as robotics, artificial intelligence, and control systems. A common characteristic among these diverse platforms is their adherence to the *sense-plan-act* paradigm, which defines the fundamental functions of an autonomous vehicle. This Chapter provides a comprehensive analysis of existing small-scale testbeds. The proposed review is structured around the *sense-plan-act* paradigm, ensuring a consistent and systematic evaluation. Car-like robots serve as excellent platforms for this purpose, combining agility with dynamic complexity. Additionally, fleets of scaled vehicles can be leveraged to simulate driving scenarios and explore multi-robot coordination strategies. This review is intended to provide deeper insights into the architecture and operational principles of the testbeds documented in the literature.

2.2 Small-scale testbeds

This comparison is intended to be purely analytical, aiming to highlight the differences and similarities between testbeds without making evaluative judgments regarding their quality or value. The goal is to offer an unbiased and comprehensive overview of existing small-scale testbeds. Such an approach supports a clearer understanding of both unique and common features, serving as a useful reference for researchers in selecting appropriate testbeds for specific applications or identifying areas for potential enhancement of current platforms.

This section presents the proposed approach for studying and comparing small-scale testbeds available in the literature. The analysis focuses on identifying distinct characteristics within the three fundamental domains—*sense*, *plan*, and *act*. These characteristics serve as key criteria that encapsulate essential functional aspects of the testbeds. In the *sense* domain, parameters such as the types of sensors used are considered, with a distinction made between onboard sensors, which are directly mounted on the vehicle, and those integrated into the testbed's infrastructure. Similarly, specific parameters are defined for the *plan* and *act* domains. Firstly, robots are categorized into two types: unicycle-type and bicycle-type vehicles. The former typically consists of a vehicle with two parallel-driven wheels, each positioned on either side of the center. The latter includes car-like robots with four wheels, where only the front two are steerable, typically using an Ackermann steering mechanism. Khepera [7] was the first small-scale unicycle to be successful. The first generation of this technology was developed at EPFL in 1991 and gained significant traction in both research [8] and education. Another widely successful platform was the LEGO Mindstorms RCX [9], LEGO's first robotic system, which became popular for low-budget middle and high school education [5] as well as research projects [6], thanks to its integration with LEGO bricks. Another notable unicycle robot, also developed at EPFL, is the e-puck [7], designed as a more affordable alternative to the larger and more expensive Khepera III. Over time, more advanced and expandable unicycle robots emerged, including iRobot Create [10], Pololu 3pi [11], Swarmrobot [12], AERobot [13], GRITSBot [14], Pheeno [15], Duckiebots [16], and Wheel-bot [17]. In recent years, remotely accessible swarm robotics platforms have been introduced, such as Mobile Emulab [18], the HoTDeC testbed [19] and the Robotarium [20]. A comprehensive list of multi-robot testbeds is available in [21]. While these platforms are well-suited for trajectory planning applications, they are limited by

their lack of agility and low maximum speed, making them less suitable for high-performance controller development. This limitation has led to the increased use of car-like vehicles in research and education. The most common platforms in this category range from medium- to full-scale vehicles, allowing for the integration of powerful computing units and various sensors, including IMUs, CMOS/CCD motion sensors, LIDARs, and wheel encoders. Notable full-scale prototypes include the Stanford Shelley [22], Stanley [23], and Marty cars [24], as well as the Formula Driverless custom cars [25], which were designed for the Formula Student Driverless competition. More recently, autonomous racing platforms like Roborace DevBot 2.0 [26] and Dallara AV-21 [27] have emerged. Among medium-scale vehicles, which typically range from 1:10 to 1:18 scale, prominent platforms include Autorally [28], Donkey Car [29], F1-Tenth cars [30] and the AWS Deep Racer [31]. These platforms primarily focus on single-agent autonomous driving and racing rather than multi-agent systems, as the latter requires large-scale infrastructure and higher costs, making multi-agent experiments less feasible. Some small-scale platforms, such as those presented in [32], [33], and [34], could support both single- and multi-agent applications. However, they are either only partially or not at all open-source, making them difficult to replicate. On the software side, only a few toolboxes are currently available for control tasks. Among them are Drake [35], the Control Toolbox [36]—which is no longer maintained—and OCS2 [37], which provides controllers but lacks a complete control pipeline, including state estimators and safety filters.

Caleffi et al. [6] conducted a systematic review summarizing existing literature on CAV testbeds. While they identified key trends—such as the recent surge in publications and the predominant emphasis on software—they did not perform a comparative analysis. Although previous surveys on testbeds exist, the most recent comparative study dates back to 2024 [38]. Many recent publications include their own related work sections, often accompanied by comparative tables highlighting selected testbeds. These tables, however, vary significantly in scope and level of detail. One of the more comprehensive examples can be found in the work of Samak et al. [39].

To address the fragmentation in existing literature, this information is consolidated into comprehensive, up-to-date tables that include the relevant testbeds found in the literature. Figure 2.1 illustrates some testbeds selected from the presented literature review.



(a) IDS Scaled Smart City (IDS3C), Cornell University, NY, USA



(b) MiniCity, Massachusetts Institute of Technology, Cambridge, USA



(c) Optimal RC Racing (ORCA Project), ETH, Zurich, CH



(d) Cambridge Minicar, University of Cambridge, Cambridge, UK

Fig. 2.1 Examples of existent small-scale tesbeds available in the present literature.

2.3 Characteristics of testbeds

This section outlines and examines the principal identified characteristics. It begins with the presentation of general characteristics that cannot be distinctly attributed to the *sense*, *plan*, or *act* domains. This is followed by an exploration of the distinctive characteristics inherent to each domain, providing a more detailed analysis of their specific attributes. In Appendix A, multiple tables are provided, listing our small-scale testbed, i.e. CARS Autonomous Scaled-Vehicle Testbed, and the existing ones in the available literature.

2.3.1 General characteristics

The main design objective of most testbeds is to facilitate the testing of algorithms within a scaled-down environment. Although individual testbeds may serve different specific purposes, they exhibit several common features. This section outlines the general characteristics by emphasizing these shared features.

Focus

The focus characteristic provides a brief, high-level overview of a testbed, typically presented in its introduction. It outlines the specific use cases the testbed is intended to address, offering a concise summary of its primary objectives. In this analysis, the focus characteristic is used to describe the investigated use cases, the methodologies applied, and the insights derived from experiments conducted within the testbed. This facilitates an understanding of the testbed's overall purpose. Additionally, testbeds may concentrate on particular problems within a broader area of focus. For instance, within the domain of multi-vehicle coordination, emphasis may be placed on aspects such as traffic management, communication, or human-autonomy interaction.

Software

The variation in software languages, architectures, frameworks, and computation models across testbeds highlights their adaptability to different needs and contexts. The choice of software components often depends on factors such as the intended application domain, the availability of relevant libraries and tools, and the preferences

or expertise of the development team. For example, some testbeds employ Python due to its simplicity and rich ecosystem of scientific computing libraries, while others may adopt performance-oriented languages or MATLAB for its domain-specific capabilities.

An additional aspect of the software characteristic involves the selection of software architecture, which defines how components are structured and interact. Common architectures include service-oriented, client-server, peer-to-peer, and event-driven models. The chosen architecture influences key properties of the testbed, such as scalability, fault tolerance, modularity, and interoperability.

Frameworks play an important role in shaping the software characteristics of testbeds by offering reusable components, libraries, and tools that facilitate software development. Testbeds often utilize existing frameworks designed for specific functions, such as simulation, Internet of Things (IoT), or machine learning applications.

Another key element of the software characteristic is the computation model. Some testbeds adopt deterministic models, which produce predictable and reproducible outcomes based on defined input conditions. These models are well-suited for experiments requiring high precision and consistency. In contrast, probabilistic computation models incorporate randomness and uncertainty, enabling the exploration of scenarios influenced by stochastic factors. Hybrid models, which combine deterministic and probabilistic elements, offer a balanced approach that supports both predictability and variability in experimental outcomes.

Scenario

A scenario denotes a defined set of conditions, events, and interactions executed within a testbed environment. Scenarios typically involve the use of various maps and the inclusion of different actors, such as pedestrians, obstacles, and other agents, that enhance the complexity and realism of the experimental setting. The selection of specific scenarios is guided by the research objectives, as scenarios are tailored to investigate particular questions. Consequently, the choice of a scenario is closely aligned with the focus of the research.

2.3.2 Sense-driven characteristics

Within the *sense* domain of the *sense-plan-act* paradigm, the primary focus is on perceiving the environment and gathering relevant information using sensors. The following section outlines the characteristics identified to describe this domain.

Sensors

Sensors constitute the core component for environmental perception within a testbed. Various types of sensors, such as cameras, LiDAR, radar, and ultrasonic sensors, offer complementary information about surrounding objects and their characteristics. The type, availability, and quality of these sensors significantly influence the accuracy and richness of the collected perceptual data. Assessing the sensor suite and its capabilities allows researchers to determine whether a testbed sufficiently meets their sensing requirements.

Positioning System

Accurate and reliable positioning information is essential for many applications. A testbed may incorporate a precise positioning system, such as an indoor positioning system, to enable accurate tracking of vehicle position and orientation.

Accuracy

Sensors within the testbed provide perceptual data, and the accuracy of this data is an important consideration. It reflects how closely the sensed information corresponds to the actual state of the environment. Evaluating the accuracy of individual sensors, as well as the overall perception system, is essential for determining whether the testbed aligns with specific accuracy requirements.

Traffic management infrastructure

This evaluation encompasses elements such as road networks, traffic signs, traffic lights, and other components that replicate real-world traffic infrastructure. A testbed equipped with a detailed traffic management setup allows researchers to analyze and

assess the behavior and effectiveness of their algorithms and systems under realistic traffic scenarios.

Surroundings

The surrounding environment of a testbed plays a significant role in determining its suitability for specific experiments. Factors such as whether the setting is urban, suburban, or rural can affect the overall complexity of the testbed. For example, an urban environment may introduce challenges like dense traffic and frequent interactions, whereas a suburban setting may present different types of obstacles and road conditions.

2.3.3 Plan-driven characteristics

The *plan* domain is essential for enabling CAV/RS to formulate suitable actions based on environmental data. This section identifies key characteristics of the *plan* domain that support the evaluation and comparison of testbeds.

Testbed architecture

The architecture of a testbed pertains to the overall design and organization of the experimental setup. This includes the physical layout, the inclusion of relevant infrastructure components, and the integration of hardware and software systems. In the context of the *plan* domain, it is important to assess whether the architecture adequately supports the implementation and evaluation of planning algorithms and strategies. To ensure reproducibility, deterministic timing across testbed components is necessary, i.e. meaning that all computing and sensing devices must be properly synchronized. A well-designed architecture can help ensure that experiments are both deterministic and reproducible.

Distributed computations

In many real-world applications, CAVs/RSs function within distributed environments, with processes running across multiple nodes. Evaluating a testbed's ability to support distributed computation is crucial, as it indicates how effectively the system

can handle coordination and communication among vehicles during the planning stage.

Computation power

The computational power of a testbed plays a key role in determining the complexity and scalability of the planning algorithms it can support. Greater processing power enables the use of more sophisticated techniques, such as optimization-based approaches. Researchers should take into account factors like the processing performance of the testbed's hardware and the availability of parallel computing resources.

Computation schemes

The computation scheme describes the approach or algorithms used for planning within the testbed. Distributed planning algorithms can adopt various schemes, such as sequential, parallel, or hybrid processing. Hybrid schemes integrate both sequential and parallel methods to enhance the efficiency of planning operations within the testbed.

Human-robot interaction

In numerous real-world situations, CAVs/RSs operate alongside and interact with humans in shared spaces. Assessing a testbed's features related to human-robot interaction is crucial for determining its effectiveness in mixed traffic scenarios. Key aspects to consider include the presence of realistic human models, the ability to simulate human behaviors, and interfaces that allow human interaction with the testbed.

Different kinds of agents

Evaluating a testbed's ability to accommodate heterogeneous vehicles is crucial when studying CAVs/RSs. Vehicles with varying sensing and actuation capabilities may impose different planning demands. Therefore, the testbed should enable the integration of diverse vehicle models, allowing researchers to test planning algorithms in scenarios that involve a mix of vehicle types, such as cars and trucks.

Agent count

The number of vehicles supported by a testbed is a key factor, especially in transportation and traffic management scenarios. Assessing this characteristic involves examining the testbed's ability to manage multiple vehicles simultaneously, how well planning algorithms scale with an increasing number of vehicles, and how overall planning performance is affected as vehicle count grows.

2.3.4 Act-driven characteristics

This subsection compiles information related to executing a plan, commonly referred to as trajectory tracking. In a hierarchical architecture like the *sense-plan-act* framework, trajectory following operates as a distinct layer from planning [40, 41]. The connection between these layers is established through a vehicle model, which encapsulates the vehicle's dynamic constraints. Effective collision avoidance in the planning layer also depends on accurate knowledge of the vehicle's dimensions

Dynamics

Modeling system dynamics is a fundamental step in classical control design. When planning accounts for the physical constraints of the vehicle, trajectory tracking becomes more accurate. Conversely, if the model used for planning deviates significantly from the actual vehicle behavior, a trajectory that appears collision-free in theory may result in a collision when executed in practice. Therefore, it is essential to validate trajectory planning models through experiments with real vehicles. Vehicles can be broadly categorized based on their steering geometry into differential-drive and Ackermann-steering types. For differential-drive vehicles, common models include the point-mass model with double-integrator dynamics and the unicycle model, which distinguishes between translational and rotational control inputs [42, 20]. For Ackermann-steering vehicles, typical models range from the point-mass and kinematic bicycle models to the kinetic bicycle model and more complex dynamic representations. A comprehensive overview of these models is provided in [43]. The selection of a particular model dictates the required vehicle parameters. In general, speed and acceleration limits constrain the longitudinal dynamics, while steering angle and steering rate limits influence the lateral dynamics.

Geometry

The vehicle dynamics models mentioned above describe the motion of the vehicle's center of gravity. However, the vehicle's physical structure extends around this central point. To ensure collision-free motion, it's necessary to verify that none of the vehicle's occupied regions intersect with obstacles. This requires knowledge of the vehicle's geometry in order to accurately compute its occupied area.

2.4 CARS facilities

The validation of control algorithms for autonomous driving has become an increasingly prominent research focus in recent years. This is driven by the need to shorten testing phases while maintaining realistic, cost-effective, and time-efficient evaluation processes.

Despite significant advancements, several challenges remain in the development and deployment of autonomous vehicles. These challenges can be grouped into public, legislative, and technical domains. On the public side, factors such as societal perception, political reactions to accidents, customer trust, cybersecurity, and ethical issues heavily influence the acceptance of these technologies [44]. From a legal standpoint, matters like certification procedures and liability in the event of an accident are still unresolved [45]. Technically, guaranteeing functional safety and reliable performance across a wide range of traffic scenarios, weather conditions, and road types remains a complex and resource-intensive task, especially given the sheer number of test cases required [46]. These challenges underline the importance of a structured and comprehensive validation methodology. One of the key questions is: what metrics should be used to approve autonomous systems? With validation costs expected to match or even exceed development costs, a solid approach is crucial. Industry estimates suggest that demonstrating a sufficient level of system reliability could require up to 250 million test kilometers, highlighting the urgency of developing more efficient validation strategies [47].

Currently, validation primarily relies on virtual simulations and on-road testing at dedicated proving grounds. While simulation costs remain low in the initial phases, the development costs associated to new autonomous features tend to rise as the process moves forward, as illustrated in Fig. 2.2.

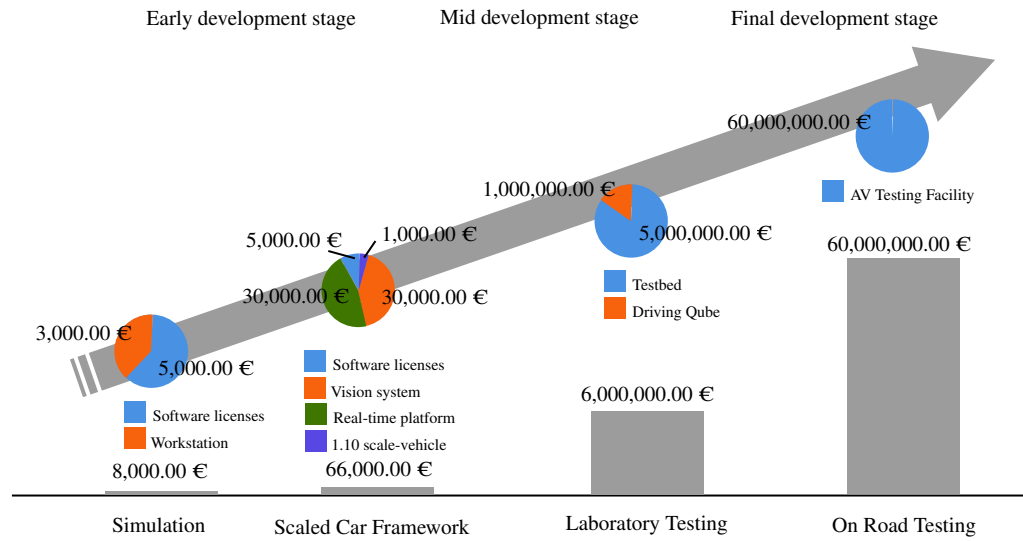


Fig. 2.2 Costs associated to early, mid and final development stage of autonomous vehicle control algorithms.

In response to the need for a more integrated approach, the Center for Automotive Research and Sustainable Mobility (CARS) at Politecnico di Torino has proposed a multi-layered methodology that combines both virtual and physical testing. This process begins with micro- and macro-level simulations, which allow for extensive scenario testing, incorporating traffic dynamics, vehicle behaviors, and assisted/autonomous vehicles controller functionalities. These simulations are essential for identifying edge cases and potential safety-critical situations. The methodology then progresses step by step, from purely virtual environments to small-scale physical tests, and finally to full-scale real-world testing, ensuring an iterative, flexible, and cost-effective validation process. The cost estimates shown in Fig. 2.2 related to the simulation, scaled car, and laboratory testing phases are drawn from the CARS group's direct investments in building and equipping the necessary infrastructure. Conversely, the costs linked to the on-road testing phase are derived from a feasibility study conducted by CARS for the development of a dedicated facility for testing autonomous, electric, and connected vehicles, envisioned for the site of the former thermoelectric power plant in Trino Vercellese [48]. It is important to note that the values presented in the figure reflect only the costs required to establish the infrastructure for each development stage. Operating and maintenance costs arising from the actual use of these infrastructures, such as energy consumption, personnel, and regular maintenance, are not included in this analysis.

In the early development stage, the simulation phase incurs a modest cost of € 8,000, primarily allocated to software licenses (€ 5,000) and workstation hardware (€ 3,000). Progressing to the scaled car framework, total expenditures rise to € 66,000, reflecting investments in software licenses (€ 5,000), a vision system (€ 30,000), a real-time platform (€ 30,000), and a 1:10 scale vehicle (€ 1,000). The cost associated with the 1:10 scale vehicle includes approximately € 170 for the physical model, € 30 for additional electronic components, and around € 800 in labor, calculated as two full working days (16 hours at € 50/hour) required for mechanical modifications and software integration. The mid development stage is marked by laboratory testing, where costs increase substantially to € 6,000,000, distributed between a testbed (€ 5,000,000) and an additional test environment to support virtual and physical validation of autonomous driving function, i.e. the Drivig Qube (€ 1,000,000). The final development stage, on-road testing, exhibits the highest expenditure at € 60,000,000, entirely attributed to the autonomous vehicle testing facility.

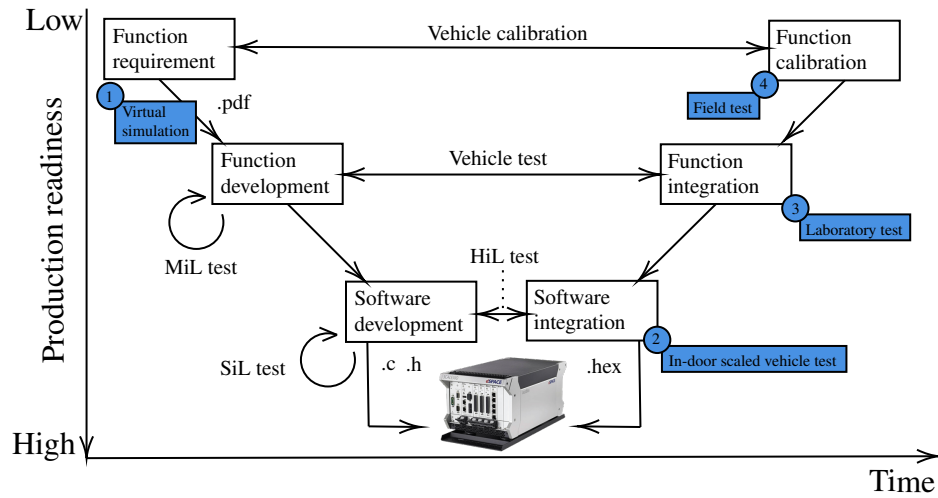
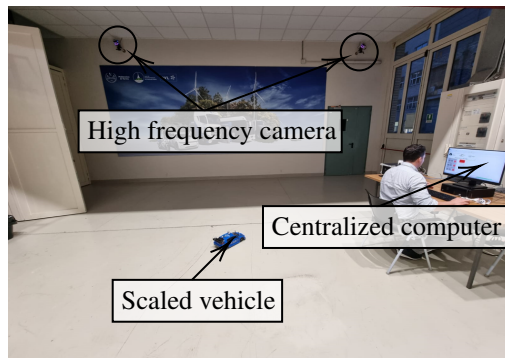
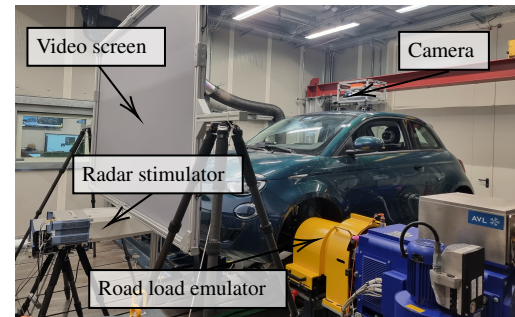


Fig. 2.3 V-cycle for autonomous vehicle control algorithms.

Figure 2.3 illustrates the V-cycle approach used for validating control algorithms for autonomous driving. Thanks to the infrastructures available at the CARS research center, the process starts with virtual simulations to define functional requirements, advances to software development and integration using Hardware-in-the-Loop (HIL) testing with scaled vehicles, and concludes with full-scale testing in the powertrain testbed, as shown in Fig. 2.4a and 2.4b.



(a) CARS Autonomous Scaled-Vehicle Testbed



(b) CARS Powertrain Testbed

Fig. 2.4 CARS facilities

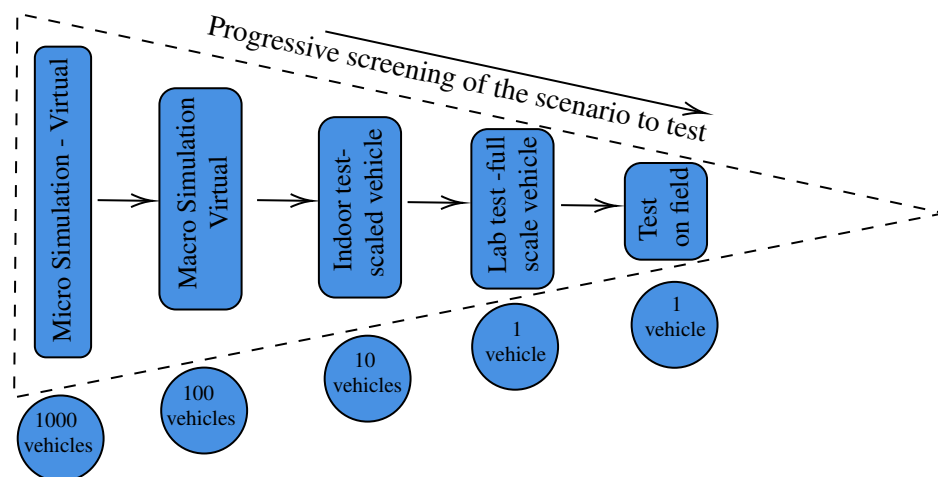


Fig. 2.5 Progressive screening of the scenario to test: from virtual simulation to test on the field.

Figure 2.5 shows a progressive screening approach for testing an autonomous vehicle scenario, starting from large-scale virtual simulations and gradually narrowing down to real-world testing. The process starts with micro simulations usually involving thousands of virtual vehicles to analyze the autonomous vehicle control system's behavior on a broad scale. Road networks and traffic behaviours are usually modeled using software tool like OpenDRIVE [49] and SUMO [50]. These simulations generate outputs such as vehicle trajectories, collisions, emergency braking incidents, and lane changes, which help define the test scenarios. It then moves to macro simulations with hundreds of vehicles, refining the model with more detailed analysis, by adding elements such as sound and visual modules, interactive 3D views and driver models. The simulation tool validates scenarios obtained from micro-simulations while ensuring high fidelity in control system validation. To bridge the gap between simulation and reality, in-door scaled-vehicle testing is introduced. Thus, in-door tests using several scaled vehicles provide controlled experimental validation in a lab environment. These tests involve scaled-down 1:10 vehicles equipped with sensors and controllers operating in a controlled environment. High-precision infrared strobe systems and real-time computing hardware facilitate accurate data acquisition and real-time control execution. The Buckingham Pi-theorem is used to analyze scaling effects and determine whether similarity principles hold between scaled and full-size vehicles. A more in-depth discussion of these aspects will be provided in the upcoming sections. This is followed by lab tests with a full-scale vehicle, allowing for real-world insights while still maintaining controlled conditions. Real vehicle-in-the-loop (VIL) tests are performed using virtual scenarios while engaging real sensors and actuators. This setup enables repeatability, allows for safe testing of edge cases, and supports intensive validation campaigns. Sensors are stimulated using radar and camera stimuli to generate realistic driving conditions, enhancing the reliability of test outcomes. It integrates tools such as AVL Testbed.CONNECT [51] for scenario generation and simulation management, AVL PUMA [52] for testbed automation and control, and a real-time target machine for the real-time execution. This setup facilitates remote monitoring, vehicle control, and vehicle controller tuning. Finally, a single vehicle undergoes field testing to assess system performance in real-world scenarios. This staged methodology ensures early identification of potential issues before advancing to costly real-world tests.

The proposed approach transitions from Model-in-the-Loop (MIL) simulations to real-world testing, ensuring that all levels of validation are covered. In summary, this

methodology proposes a multi-tiered validation strategy that combines simulation, scaled-vehicle testing, and real-vehicle laboratory tests to address the challenges of autonomous vehicle validation. By leveraging both virtual and physical testing environments, it aims to create a more efficient, scalable, and reliable approach to autonomous driving system validation.

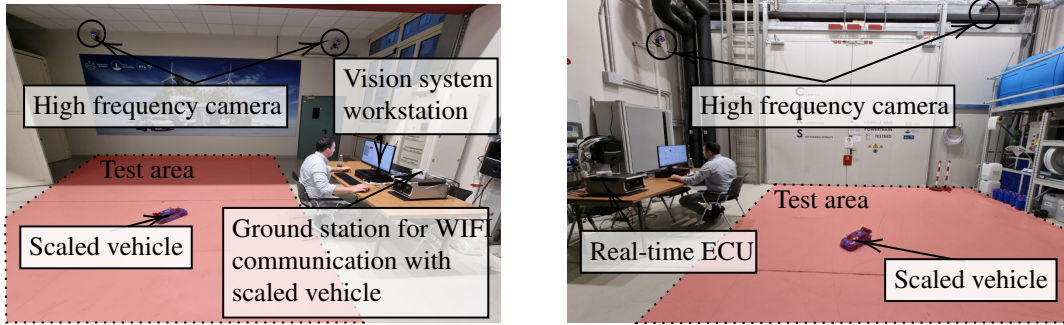
Chapter 3

CARS Autonomous Scaled-Vehicle Testbed

This Chapter outlines the experimental setup and architecture of the proposed novel autonomous scaled-vehicle testbed, detailing its framework, i.e. the vision system, the embedded 1:10 scale cars and the real-time hardware. Moreover, the communication system connecting all components and the autonomous software architecture is presented.

3.1 Testing scenarios

The experimental tests for the scaled autonomous vehicles are conducted in a dedicated space with dimensions of about 5 m x 6 m as shown in Fig. 3.1a and 3.1b. Urban driving scenarios, such as road intersections, or small circuits can be constructed within this setup. The test area is not physically bounded, thus enabling rapid modification of the scenario to be tested. Enhancing the traction of the model vehicles with additional surface materials is not feasible, due to limitations that prevent the installation of alternative floor coverings in the test area. Due to physical constraints imposed by the vehicle's maximum steering angle, the tracks are designed with a minimum turning radius of 0.7 m. This ensures that the layouts remain within the physical limits of the scaled model.



(a) Testing area: 2 high frequency cameras, the vision system workstation, the ground station

(b) Testing area: 2 high frequency cameras, the real-time ECU, the scaled vehicle

Fig. 3.1 CARS Autonomous Scaled-Vehicle Testbed

3.2 Miniature RC cars

The used miniature race cars are *Tamiya TT-02* 1:10 scale cars, which are about 0.416 m x 0.190 m in size and weight 1.6 kg. The cars features a bathtub-type frame, four double wishbone suspension with friction dampers. The cars are rear wheel driven (RWD) by a DC brushless electric motor, featuring one-speed gearbox and open differential transmission. The front wheel are steerable with a proportional front wheel steering mechanism. The cars do not have separated brakes, but the motor can break the rear axle. It is therefore important to account for this by constraining the vehicle's maximum braking force, so as to avoid the onset of vehicle dynamics instability [53].

The original electronics of the cars are replaced with a custom electronics that allow a custom made communication protocol and onboard sensing. The onboard electronics replace the conventional radio control system board. The Electronic Speed Controller (ESC) and steering servo are regulated through Pulse-Width Modulation (PWM) signals produced by the ESP32-WROOM-32D [5] microcontroller, which provides integrated Wi-Fi connectivity, Serial Pheripheral Interface (SPI) and Inter-Integrated Circuit communication (I2C) support for effortless peripheral integration. An Inertial Measurement Unit (IMU), specifically the MPU-9250 [54], is positioned near the vehicle's center of gravity to guarantee precise motion detection. The IMU interfaces with the microcontroller via I2C, delivering real-time data on acceleration and angular velocity across all three axes.

All hardware components are integrated seamlessly to preserve the vehicle's original dynamic characteristics while ensuring dependable autonomous functionality. Meticulous selection and positioning of each component guarantee that the alterations do not disrupt the vehicle's inherent handling attributes.

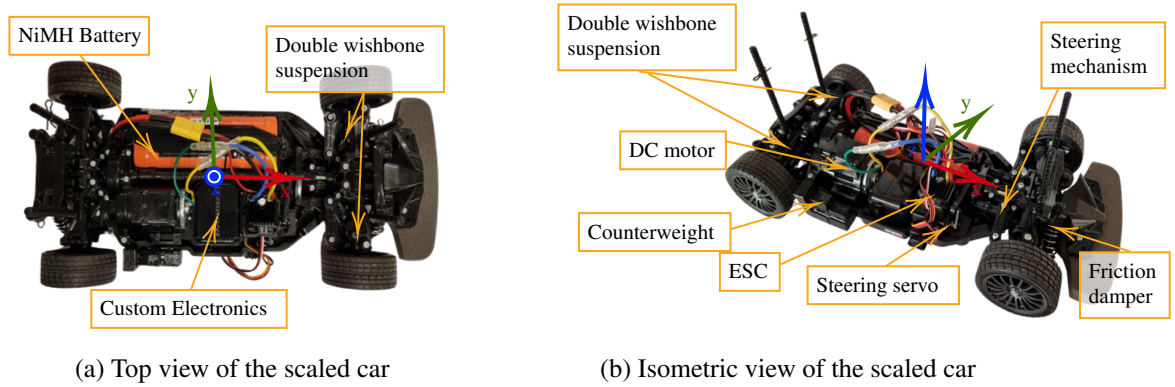


Fig. 3.2 1:10 scale vehicle with custom modifications

Figure 3.2a and 3.2b show the embedded scale car. Counterweights were added to the right rear corner of the vehicle to compensate for the imbalance caused by the battery placement, which significantly increased the load on the left-side wheels. This weight distribution asymmetry became particularly noticeable at high speeds: the vehicle exhibited oversteer during right turns (positive steering angles) and understeer during left turns (negative steering angles) [53]. By introducing the counterweights, the vehicle's dynamic behavior was restored to a more symmetrical response along the longitudinal x axis of the scaled vehicle.

Communication system

The system uses a master-slave communication architecture, where a central ground control unit (master) manages data exchange with multiple peripheral devices (slaves), i.e. the scaled cars. Each scaled vehicle communicates via Wi-Fi communication with a custom ground-based control unit. The ground station is equipped with an ESP32-WROOM-32D microcontroller [5], identical to the one installed on the vehicle. The Wi-Fi communication is capable of exchange information at maximum distance of 50 m. Moreover, it introduces a maximum transmission delay of approximately 50 ms.

3.3 Vision based localization system

Vehicle localization is achieved through a vision-based system consisting of four infrared cameras. These cameras are mounted on the ceiling, one in each corner of the room, and oriented downward to cover the entire test area.

Camera

The cameras used are VICON Vero v2.2 models [55], operating at 200 fps (i.e. 200 Hz). Each camera is equipped with a wide-angle lens to capture at least half of the test area in a single frame, ensuring sufficient overlap between the fields of view of different cameras. The infrared LEDs arranged around each camera assist in detecting the reflective markers placed on top of the scaled vehicles, enabling accurate motion tracking within the test environment.

Vision Software

The goal of the vision software is to extract the position and orientation of all the cars on the test area and send this information to the controller PC. The cameras are connected via UDP/IP to a 10 Gb/s PoE Ethernet switch, which in turn communicates with a workstation running the proprietary VICON Tracker software [56] through a UDP/IP connection. The relative positioning of the markers on the vehicle body must be unique and asymmetrical with respect to the vehicle's axes to ensure that the vision system can unambiguously track the movement of the object. The marker configuration defines the object to be tracked, which is then assigned its own reference frame expressed in global coordinates, relative to the origin of the test area's coordinate system. Figure 3.3 shows the two different marker patterns used to identify the scaled vehicles.

3.4 Controller PC

The controller PC utilizes a Linux-based real-time operating system (RTOS) . The sampling time is restricted to a multiple of 5 ms due to the update of the camera system.

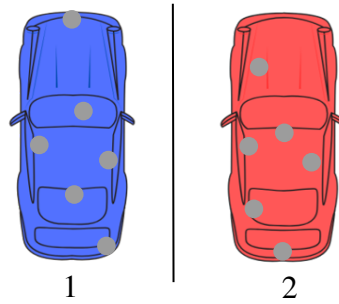


Fig. 3.3 Schematic drawing of the two different marker patterns used to identify the scaled cars

Control loop and system delays

The resulting control loop can be summarized as follows; the controller PC sends commands to the car via CAN communication to the ground station. The ground station sends commands to the scaled car via WIFI communication, which drives along the track. The cameras then take new images which are transmitted via Ethernet UDP/IP to the vision PC. The images are processed by the proprietary vision software, and the new position of the cars is detected. This information is then sent via Ethernet UDP/IP to the controller PC which computes new commands for the car.

A fundamental challenge associated with complex feedback loops is the presence of time delays. In the system under consideration, such delays originate from both communication latencies across all the links and computational delays within the vision processing and control modules. Specifically, the vision system has an average latency of approximately 4 ms, while the UDP/IP Ethernet transmission between the vision PC and the controller PC introduces an additional delay of roughly 5 ms. The controller updates the computed control commands, i.e. throttle, brake, and steering, at a maximum interval of 50 ms. Reference setpoints for the electric motor and steering duties are transmitted via the CAN bus at 10 ms intervals. These signals are received by the ground station with the ESP32-WROOM-32D microcontroller through the integrated TWAI peripheral. Upon reception, which is interrupt-driven at a 10 ms rate, the ESP32 microcontroller processes the incoming data and transmits it wirelessly via ESP-NOW to the onboard ESP32 units installed onboard the scaled vehicles. This architecture effectively bridges the wired CAN communication with the wireless ESP-NOW protocol, facilitating

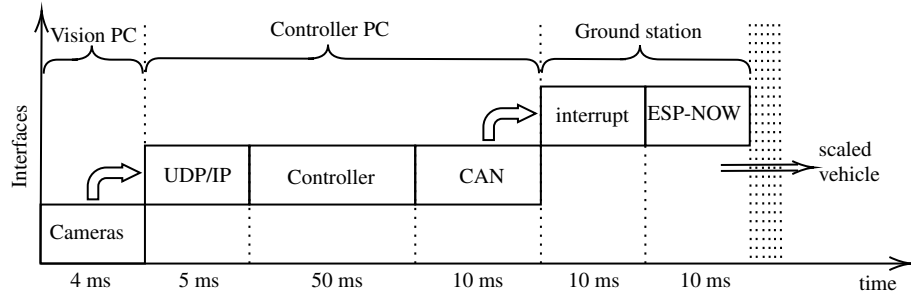


Fig. 3.4 Cascade time delays throughout the entire communication pipeline

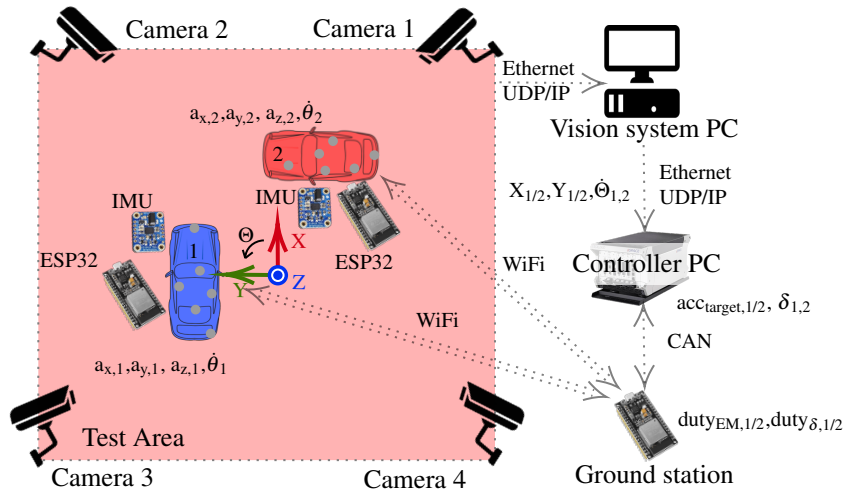


Fig. 3.5 Schematic overview of the CARS Autonomous Scaled-Vehicle Testbed

real-time transmission of control messages. Importantly, the delays introduced by the communication architecture do not compromise the performance or stability of the control system. Figure 3.4 illustrates the cascaded time delays throughout the entire communication pipeline, from the vision system to the actuators on the scaled vehicles, with a total time delay of about 89 ms.

3.5 Overview

Figure 3.5 provides a schematic overview of the proposed scaled infrastructure. The vision system, composed of four cameras, tracks the motion of the scaled vehicles and transmits their poses expressed in global Cartesian coordinates, i.e. $X_{1/2}$, $Y_{1/2}$, $Z_{1/2}$, from the Vision System PC to the Controller PC. The Controller PC filters the incoming data and computes the control actions, namely the target acceleration

($acc_{target,1/2}$) and the steering angle ($\delta_{1/2}$). These high-level control commands are then mapped by a low-level controller into duty cycle references for the electric motor and the steering servo, respectively. The resulting commands are sent via CAN bus to the ground station, which subsequently transmits them to the scaled vehicles using ESP-NOW wireless communication. In addition, the ground station receives inertial measurements, such as acceleration along the three axes and yaw rate, from the IMU sensor of each vehicle, transmitted via ESP-NOW. These sensor measurements are then forwarded back to the Controller PC through the CAN interface.

Table 3.1 summarizes how our infrastructure compares to those presented in the literature, based on four key aspects:

- *Planning and control*: The ability to implement planning and trajectory tracking algorithms;
- *Computer vision*: The ability to develop perception algorithms using onboard sensors and dedicated electronics;
- *Autonomous racing*: The capability to enable vehicle interaction, supporting the development of obstacle avoidance strategies with the goal of competitive autonomous racing;
- *Human-robot interaction*: The possibility to control the scaled vehicle through human input, e.g., teleoperation.

Table 3.1 Comparison of laboratory scaled-car testbed based on selected characteristics.

	Planning and Control	Computer vision	Collective behaviour	Autonomous racing	Human-robot interaction
CARS Autonomous Scaled-Vehicle Testbed	✓		✓	✓	✓
CPM Lab	✓				✓
F1TENTH	✓	✓		✓	
Robotarium	✓		✓		
IDS3C(UDSSC)	✓				✓
Cambridge Minicar	✓				✓
Duckietown	✓	✓			
CHARTOPOLIS	✓	✓			✓
Cambridge RoboMaster	✓	✓			
Go-Chart		✓			✓
CRS	✓			✓	
MiniCity	✓	✓			
Optimal RC racing (ORCA)	✓			✓	
Miniature Autonomy	✓	✓			
Self-Driving Car Studio at University of Surrey	✓	✓	✓		

Chapter 4

Vehicle model

This Chapter presents the derivation of the vehicle models employed throughout the thesis, which serve as the foundation for system identification and control design. The structure of the Chapter is as follows, first the identification is derived in Section 4.1. Subsequently, Section 4.2 discusses the dynamics similarity of the scaled car with a full scale vehicle using the Buckingham Pi theorem approach.

4.1 System Identification

This section provides a summary of the methodology employed to identify the scaled car model. The identification process is structured into three main stages. In the first stage, static parameters, including the vehicle mass, are estimated. The second stage involves conducting sinusoidal acceleration tests to identify parameters influencing the longitudinal dynamics. In the third stage, ramp steer experiments are performed to estimate the characteristics of the tire force response.

It is important to highlight several challenges associated with the identification of the miniature race car. Firstly, the objective is to derive a model that remains valid even near the handling limits. Secondly, the available measurement data is limited; in particular, wheel speed, body roll, and pitch information are not accessible. Lastly, tire characteristics can only be identified through driving experiments, as no dedicated tire testing equipment is available for the scaled platform.

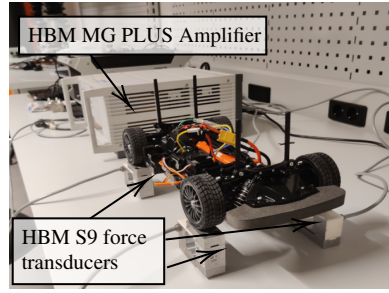


Fig. 4.1 Scaled-vehicle vehicle distribution test

Table 4.1 Scaled-vehicle weight distribution

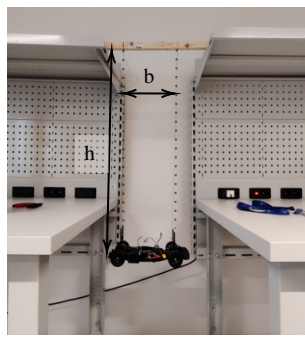
Weight Distribution	Forces [N]	Proportion
Left/Right	7.87/7.89	49.9%/50.1%
Front/Rear	7.20/8.56	45.7%/54.3%

4.1.1 Static Parameters

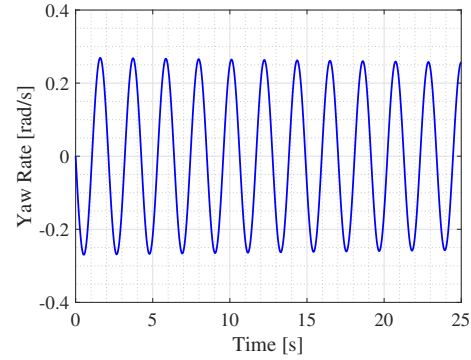
Since the scaled-car producer does not publish exact vehicle data, i.e. mass, center of gravity, moment of inertia and steering limits, these parameters have to be measured or identified. The vehicle mass m is measured using four HBM S9 force transducers connected to an MG PLUS amplifier system for signal conditioning and acquisition, as shown in Fig. 4.1. The center of gravity is estimated by analyzing the weight distribution at the four corners of the vehicle. In the lateral direction, it is assumed that the center of gravity lies at the geometric center of the vehicle.

As discussed in Section 3.2, the original vehicle exhibited a significant asymmetry, resulting in different dynamic behavior depending on the sign of the steering angle input. To address this issue, additional weights are added to the rear-right side of the vehicle to counterbalance the battery, which is mounted longitudinally along the left side. The effectiveness of the weight distribution adjustment is verified, as reported in Tab. 4.1. It is worth noting that a greater proportion of the total mass is concentrated at the rear, primarily due to the placement of the drive motor on the rear-right side of the vehicle.

The moment of inertia about the vertical axis (I_z) is determined using the bifilar pendulum method [57], in which the vehicle is suspended by two parallel strings and allowed to oscillate freely around its z axis. This technique is based on simple oscillation measurements and known geometric parameters, making it both easy



(a) Bifilar experimental test



(b) Yaw rate oscillation during the bifilar test

Fig. 4.2 Moment of inertia measurement

Table 4.2 Yaw moment of inertia I_z for different string heights h

	h [m]	I_z [$kg \cdot m^2$]
Test 1	0.68	0.0123
Test 2	0.92	0.0121
Test 3	1.15	0.0125
Test 4	1.53	0.0124

to implement and reproducible. The yaw moment of inertia is computed using the following equation:

$$I_z = \frac{Mgb^2T^2}{4\pi^2h} \quad (4.1)$$

where m is the mass of the vehicle, g is the gravitational acceleration, b is the spacing between the strings, T is the time between two consecutive peaks of yaw rate measurement and h is the string length. A visual representation of the experimental setup is provided in Fig. 4.2a. Figure 4.2b shows the yaw rate oscillation during the bifilar test.

To ensure the reliability of the measurements, the test was conducted at four different string lengths (h), as summarized in Table 4.2.

Finally, the last static parameter to be identified is the vehicle's steering limit. Simple tests were performed by setting the steering to its maximum angle, first to the left, then to the right, while driving at low speed, enabling the application of the kinematic steering formula:

Table 4.3 Summary of static parameters of the scale-car

Parameter	Value
Mass (m)	1.60 [kg]
Distance CoG to front axle (l_f)	0.14 [m]
Distance CoG to rear axle (l_r)	0.12 [m]
Wheelbase (l)	0.26 [m]
Yaw moment of inertia (I_z)	0.0123 [kg · m ²]
Steering angle limit (δ_{limit})	0.31 [rad]

$$\delta_{kin} = \frac{l}{R} \quad (4.2)$$

where l is the wheelbase and R is the radius of curvature.

In both cases, the radius of curvature is similar, with $R_{left} = 0.83$ and $R_{right} = 0.82$, further confirming the symmetry of the steering. The static parameters of the scaled vehicle are listed in Table 4.3.

4.2 Longitudinal dynamics

In the literature, the longitudinal dynamics of scaled vehicles are typically modeled through an equation that relates the longitudinal force generated by the powertrain to the duty cycle D of the pulse width modulation (PWM) used to control the torque of the DC motor. This is because the control input for the longitudinal dynamics is the duty cycle D of the electric drivetrain motor.

In [58], the longitudinal force F_x on the rear axle, since the vehicle is rear-wheel drive (RWD), is modeled using a DC motor representation, along with resistance forces, such as rolling resistance and aerodynamic drag:

$$F_x(t) = (C_{m1} - C_{m2}v(t))D - C_{rr} - C_d v(t)^2 \quad (4.3)$$

where the parameters C_{m1} and C_{m2} describe the motor model, C_{rr} represents the rolling resistance, and C_d accounts for the aerodynamic drag.

Similarly, [59] considers a four-wheel drive (4WD) vehicle, in which the electric motor generates a traction force distributed between the front and rear axles, parametrized by a torque split coefficient. The friction force is modeled as a Taylor

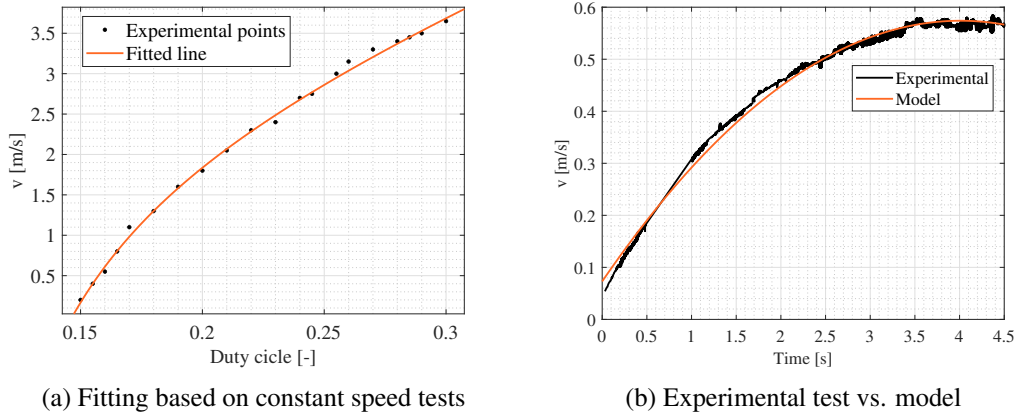


Fig. 4.3 Longitudinal vehicle model

expansion along the longitudinal direction, directly influencing the longitudinal velocity.

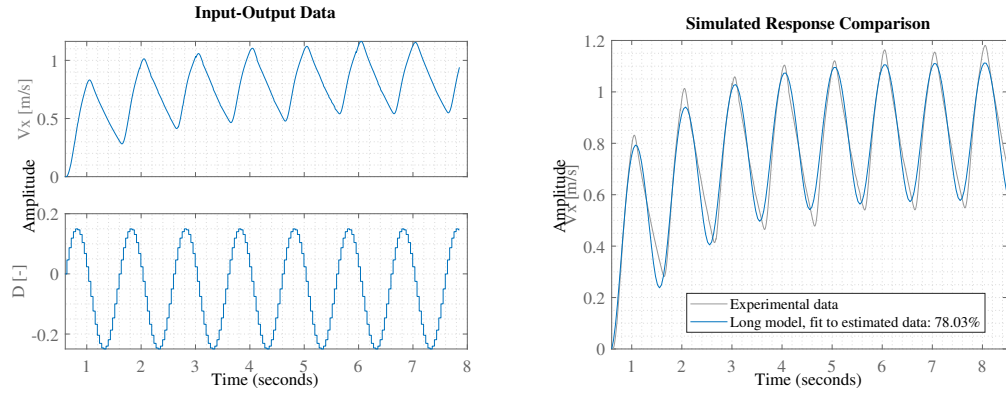
By performing constant-speed tests at different duty cycle values, a nonlinear relationship between the duty cycle and vehicle speed is obtained. The model parameters can be identified using a nonlinear least squares optimization problem of the form:

$$\min_{x_m} \frac{1}{2} \|\eta - M(x_m)\|_2^2 \quad (4.4)$$

where $x_m = [C_{m1}, C_{m2}, C_{rr}, C_d]$ are the parameters from equation 4.3 to be optimized for best fitting between measured data η and model output $M(x_m)$. The optimization algorithm used is Levenberg-Marquardt [60], a generalization of the Gauss-Newton method that enhances convergence robustness, especially when starting far from the solution.

Considering Eq. 4.3 under steady-state conditions, acceleration $\dot{v} = 0$, thus $F_x = 0$, the longitudinal vehicle model effectively fits the stationary points obtained from experimental tests, yielding an R-squared value of 0.99 and a Root Mean Square Error (RMSE) of 0.059, as shown in Fig. 4.3a. Comparing the experimental response of the scaled vehicle with that of the model, minimal mismatch between them is revealed, as illustrated in Fig. 4.3b.

However, this modeling approach is based on stationary operating points (constant speed tests) defined by fixed duty cycle values. For the identification of the longitudinal dynamics, dynamic experimental tests with varying duty cycles are required. In this work, a sinusoidal input in the duty cycle is used instead of step



(a) Input-output data during sinusoidal acceleration test (b) Comparison between experimental and simulated response

Fig. 4.4 Modeling vehicle longitudinal dynamics response

Table 4.4 Parameters defining the longitudinal dynamics of the scaled car using non-linear grey-box identification method

Parameters	Value	Std. deviation
C_{m1}	21.19 [N]	7.39
C_{m2}	8.89 [Ns/m]	9.74
C_{r0}	1.48 [Ns/m]	0.19
C_{r2}	1 [Ns/m]	0.30

changes. This choice is motivated by the fact that step inputs often lead to tire slip and saturation phenomena, which lie outside the operating range modeled by the internal vehicle model adopted in the Model Predictive Controller (MPC). Including such nonlinear behaviors—such as tire saturation—would compromise the validity of the identified model, as these effects are not accounted for in the simplified structure used for control design.

When the system is excited with a sinusoidal duty cycle input, as reported in Fig. 4.4a, the model provides a good fit with the experimental data, achieving a performance of approximately 78%, as illustrated in Fig.4.4b. However, this modeling approach presents a critical limitation: the identified parameters exhibit high standard deviations, as reported in Table 4.4. In particular, the large deviations associated with C_{m1} and C_{m2} indicate that these parameters vary substantially across different experimental trials. This variability undermines the reliability of the identification process, suggesting that the proposed model is insufficiently robust to accurately capture the longitudinal dynamics of the vehicle.

Therefore, it is necessary to develop an alternative formulation that accounts for model uncertainties through a different representation. Adopting an approach analogous to that presented in [61], the dynamics of a DC motor driven by a duty cycle (D) can be described through an equation derived from the following fundamental principles:

- Motor model: the motor speed $\omega(t)$ is determined by the applied voltage (proportional to D), the motor's mechanical and electrical characteristics, and any braking or frictional effects
- Control by Duty cycle:
 - When $D > 0$, the motor accelerates due to the applied voltage
 - When $D \leq 0$, braking occurs to reduce the speed to zero
- Braking behaviour: for Braking, a resistive force (or torque) proportional to speed is applied.

The general dynamic equation for the motor speed can be written as:

$$J \frac{d\omega(t)}{dt} = \tau_m - \tau_{load} - \tau_{brake} \quad (4.5)$$

where J is the rotational inertia of the motor, τ_{load} is the motor torque, $\tau_{load} = b_f \omega(t)$ is the load torque, where b_f is the damping coefficient that accounts for friction and $\omega(t)$ is the angular speed of the motor at time t , τ_{brake} is the braking torque. τ_{brake} is applied only when $D \leq 0$,

$$\tau_{brake} = \begin{cases} 0 & \text{if } D > 0 \\ -k_b \omega(t) & \text{if } D \leq 0 \end{cases} \quad (4.6)$$

where k_b is the braking constant. Substituting τ_{brake} , the Eq. 4.5 becomes:

$$J \frac{d\omega(t)}{dt} = \begin{cases} \tau_m - b_f \omega(t) & \text{if } D > 0 \\ -(b_f + k_b) \omega(t) & \text{if } D \leq 0 \end{cases} \quad (4.7)$$

During acceleration ($D > 0$), the motor accelerates proportionally to the duty cycle and slows due to frictional damping b_f , while during braking ($D \leq 0$), the

motor decelerates to zero speed due to combined friction and braking ($b_f + k_b$). To accurately capture the motor's dynamics, it is essential to account for the back electromotive force (V_{emf}) by considering the motor's electrical circuit. The presence of back-EMF effectively reduces the net voltage applied to the motor windings, thereby limiting the current flow and, consequently, the torque generation during acceleration phases. The motor's electrical equation is given by:

$$V_a = L \frac{di(t)}{dt} + R_m i(t) + V_{emf} \quad (4.8)$$

where V_a is the applied voltage, proportional to Duty cycle D , L is the motor inductance, R_m is the motor resistance, $i(t)$ is the motor current and $V_{emf} = K_e \omega(t)$ is the back-EMF, proportional to the motor speed $\omega(t)$ through the back electromotive force constant K_e . The torque generated by the motor τ_m is proportional to the current:

$$\tau_m = K_t i(t) \quad (4.9)$$

where K_t is the motor torque constant. The applied voltage is proportional to the Duty cycle:

$$V_a = D V_{max} \quad (4.10)$$

where V_{max} is the maximum DC motor voltage. Substituting $i(t) = \frac{V_a - V_{emf}}{R_m}$ into the Eq. 4.9:

$$\tau_m = K_t \frac{V_a - K_e \omega(t)}{R_m} \quad (4.11)$$

Then, substituting Eq. 4.11 into Eq. 4.5:

$$J \frac{d\omega(t)}{dt} = K_t \frac{V_a - K_e \omega(t)}{R_m} - b_f \omega(t) - \tau_{brake} \quad (4.12)$$

Experimental results indicate that the braking mechanism operates by short-circuiting the motor windings using an H-Bridge configuration commanded by the scaled vehicle ESC. Thus, the braking force arises from the current induced by the back-EMF as the motor decelerates. This mechanism is known as dynamic braking or short-circuit braking. The braking torque in this case is directly related to the back-EMF and the negative Duty cycle. When the Duty cycle is negative, the H-bridge applies a short circuit to the motor's windings. The back-EMF generated by the spinning motor drives a braking current through the motor's resistance and

inductance. The braking current is approximately:

$$i_{\text{brake}} \approx \frac{K_e \omega}{R_{\text{total}}} \quad (4.13)$$

where $R_{\text{total}} = R_m + R_{H\text{-bridge}}$ includes the motor resistance and any additional resistance in the H-bridge circuit. The braking torque is proportional to the braking current:

$$\tau_{\text{brake}} = K_t i_{\text{brake}} = \frac{K_t K_e \omega}{R_{\text{total}}} \quad (4.14)$$

Since the H-bridge controls the short-circuiting proportionally to the negative Duty cycle ($-D$), the braking torque scales with $-D$:

$$\tau_{\text{brake}} = -|D| \frac{K_t K_e \omega}{R_{\text{total}}} \quad (4.15)$$

Substituting Eq. 4.15 and factoring out terms, Eq. 4.12 becomes

$$J \frac{d\omega(t)}{dt} = \frac{K_t}{R_m} \cdot \max(D, 0) V_{\text{max}} - \omega(t) \left(\frac{K_t K_e}{R_m} + b_f + \frac{K_t K_e}{R_{\text{total}}} \cdot \min(D, 0) \right) \quad (4.16)$$

Thus, a single expression is obtained in which the duty cycle D inherently governs the transition between acceleration ($D > 0$) and braking phases ($D \leq 0$). The DC motor generates a torque τ_m via a shaft with radius R_s

$$F(t) = \frac{\tau_m(t)}{R_s} = J \frac{d\omega(t)}{R_s dt} \quad (4.17)$$

Thus, rearranging Eq. 4.16 to express the force produced by the DC electric motor, using Eq. 4.17:

$$F(t) = \begin{cases} \frac{K_t D V_{\text{max}}}{R_m R_s} - v(t) \left(\frac{K_t K_e}{R_m R_s^2} + \frac{b_f}{R_s^2} \right) & \text{if } D > 0 \\ -v(t) \left(|D| \frac{K_t K_e}{R_{\text{total}} R_s^2} + \frac{b_f}{R_s^2} + \frac{K_t K_e}{R_m R_s^2} \right) & \text{if } D \leq 0 \end{cases} \quad (4.18)$$

By simplifying Eq. 4.18 and lumping together all the unknown parameters of the electric motor, as well as the friction coefficient, a simplified version of the force

equation is obtained:

$$F_x(t) = \begin{cases} K_{m1}D - v(t)K_{m2} & \text{if } D > 0 \\ v(t)(K_{brake}D - K_{m2}) & \text{if } D \leq 0 \end{cases} \quad (4.19)$$

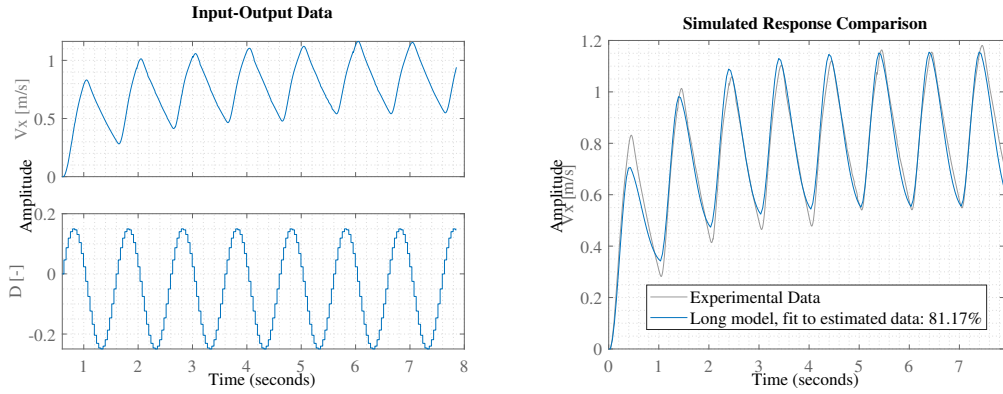
These 3 parameters can be identified using a hybrid strategy based on nonlinear grey-box identification [62]. This approach combines physical insight with data-driven techniques to capture complex system dynamics. Unlike black-box models, which rely solely on input-output data, or white-box models, which are purely based on first-principles equations, grey-box models incorporate partial knowledge of the system's structure while estimating unknown parameters from experimental data. This approach is particularly useful when the system exhibits nonlinear behavior that cannot be accurately represented using linear models, and when some underlying physical relationships are known but not fully characterized. Mathematically, a general continuous-time nonlinear grey-box model can be represented by a set of differential equations:

$$\begin{aligned} \dot{x}(t) &= f(x(t), u(t), \theta) \\ y(t) &= g(x(t), u(t), \theta) \end{aligned} \quad (4.20)$$

where $x(t) \in \mathbb{R}^n$ is the state vector, $u(t) \in \mathbb{R}^m$ is the input vector, $y(t) \in \mathbb{R}^p$ is the output vector, $\theta \in \mathbb{R}^q$ is a vector of unknown parameters, and f, g are known nonlinear functions derived from physical insight. This identification task consists of estimating the parameter vector θ such that the simulated outputs $y(t)$ best matches the measured data $\tilde{y}(t)$ over a time horizon. This is typically formulated as an optimization problem:

$$\min_{\theta} J(\theta) = \sum_{k=1}^N \|\tilde{y}(t_k) - y(t_k; \theta)\|^2 \quad (4.21)$$

where $J(\theta)$ is the cost function expressing the discrepancy between measured and model-predicted outputs. In our specific application, the optimization is constrained by the nonlinear state equations, and include regularization terms to account for prior knowledge and to enforce physical plausibility. Using this approach, a set of parameters, summarized in Tab. 4.5, is identified, achieving an 81.7% fit with the experimental data, as shown in Fig. 4.5b. Finally, employing a longitudinal dynamics model with fewer parameters compared to the previous formulation leads



(a) Input-output data during sinusoidal acceleration test (b) Comparison between experimental and simulated response

Fig. 4.5 Modeling vehicle longitudinal dynamics response

Table 4.5 Parameters defining the longitudinal dynamics of the scaled car using non-linear grey-box identification method

Parameters	Value	Std. deviation
K_{m1}	32.4 [N]	2.34
K_{m2}	0.75 [Ns/m]	0.07
K_{brake}	12 [Ns/m]	0.24

to reduced standard deviations in the identified parameters, thereby improving the robustness and reliability of the identification process.

4.3 Lateral dynamics

The miniature car is modeled using the bicycle model, following the approach described in [63, 64]. In this formulation, the four-wheeled car is represented as a single rigid body with mass m and yaw inertia I_z , while the four wheels are reduced to an equivalent two-wheel configuration. Lateral and longitudinal tire forces are applied at each of the two wheels, with the steering angle δ influencing the orientation of the front wheel.

The vehicle model, as shown Fig. 4.6, considers three degrees of freedom: two translational and one rotational. Body motions such as pitch and roll are neglected, and load transfer effects are not taken into account. Additionally, the longitudinal force is applied exclusively at the rear tires, consistent with the rear-wheel drive

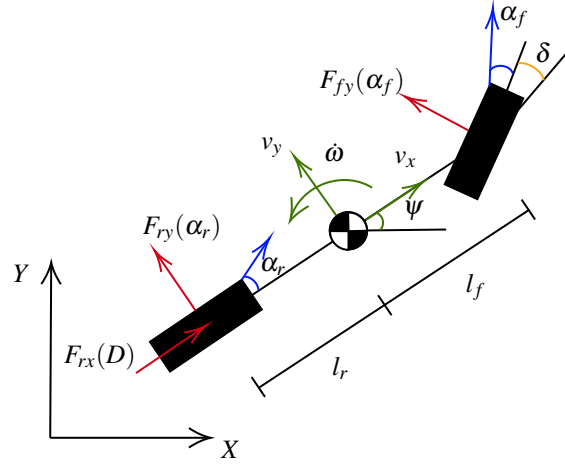


Fig. 4.6 Dynamic single-track model

configuration, while steering is implemented only at the front tires through the steering angle δ . The resulting dynamic equations are expressed as follows:

$$\begin{aligned}
 \dot{X} &= v_x \cos(\psi) - v_y \sin(\psi) \\
 \dot{Y} &= v_x \sin(\psi) + v_y \cos(\psi) \\
 \dot{\psi} &= \omega \\
 \dot{v}_x &= \frac{1}{m} (F_{rx} - F_{fy} \sin(\delta) + m v_y \omega) \\
 \dot{v}_y &= \frac{1}{m} (F_{fy} \cos(\delta) + F_{ry} - m v_x \omega) \\
 \dot{\omega} &= \frac{1}{I_z} (l_f F_{fy} \cos(\delta) - l_r F_{ry})
 \end{aligned} \tag{4.22}$$

where X and Y represent the global position coordinates of the vehicle, while ψ denotes its yaw angle or orientation. The variables v_x and v_y correspond to the longitudinal and lateral velocities in the vehicle's local reference frame, and ω indicates the yaw rate. The forces F_{rx} and F_{ry} refer to the longitudinal and lateral tire forces at the rear axle, whereas F_{fy} represents the lateral force at the front axle. The parameters l_f and l_r denote the distances from the vehicle's center of mass to the front and rear axles, respectively. The lateral tire forces are represented using a simplified version of the Pacejka Magic Formula [65], which offers a realistic approximation of tire dynamics. This model accounts for the nonlinear characteristics of tire behavior, including the saturation phenomenon, where increases in slip angle no longer result in higher lateral force. The lateral forces depend on the slip angle,

defined as the angle between the lateral and longitudinal velocity components at each tire. Assuming the vehicle behaves as a rigid body, all points share the same angular velocity ω . Given the velocity components at the center of gravity (CoG), it is possible to determine the following equation:

$$\vec{v}_i = \vec{v}_{CoG} + \vec{\omega} \wedge (P_i - CoG) \quad (4.23)$$

where $\vec{v}_{CoG}$ is the velocity at the CoG, ω is the angular velocity and P_i is the absolute position of a specific point i on the rigid body. Thus, it is obtained:

$$\begin{pmatrix} v_{xi} \\ v_{yi} \end{pmatrix} = \begin{pmatrix} v_x - \omega y_i \\ v_y + \omega x_i \end{pmatrix} \quad (4.24)$$

where v_x and v_y are the components of the velocity at the CoG, while x_i and y_i are the longitudinal and lateral distances respectively of the point P_i relative to the CoG. Thus, the side slip angle can be written as,

$$\alpha_i = \arctan \left(\frac{v_y + \omega x_i}{v_x - \omega y_i} \right) - \delta_i \quad (4.25)$$

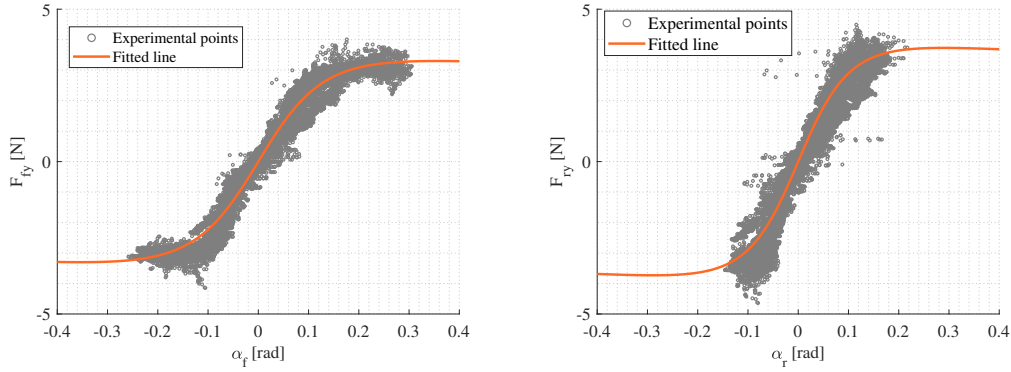
Substituting the x_i value, $-l_f$ for front tires and $-l_r$ for rear tires,

$$\begin{aligned} \alpha_f &= \arctan \left(\frac{v_y + \omega l_f}{v_x} \right) - \delta \\ \alpha_r &= \arctan \left(\frac{v_y - \omega l_r}{v_x} \right) \end{aligned} \quad (4.26)$$

Lateral forces, using simplified Pacejka Magic Formula are defined by

$$\begin{aligned} F_{fy} &= -D_f \sin(C_f \arctan(B_f \alpha_f)) \\ F_{ry} &= -D_r \sin(C_r \arctan(B_r \alpha_r)) \end{aligned} \quad (4.27)$$

where B_f , C_f , D_f , B_r , C_r , D_r , are the Pacejka parameters for the front and rear tires respectively. An initial estimate of the tire friction parameters can be derived under the assumption of steady-state conditions, i.e., when $\dot{v}_y = 0$ and $\dot{\omega} = 0$. Under this assumption, Eq. 4.22 leads to the following system of nonlinear equations



(a) Experimental side slip angle vs. lateral front tire force (b) Experimental side slip angle vs. lateral rear tire force

Fig. 4.7 Experimental lateral forces vs Pacejka simplified model

representing the lateral forces:

$$\begin{aligned} F_{r,y} &= \frac{ml_f}{l_f + l_r} v_x \omega \\ F_{f,y} &= \frac{ml_r}{l_f + l_r} \frac{v_x \omega}{\cos(\delta)} \end{aligned} \quad (4.28)$$

Therefore, once the velocities (v_x, v_y, ω) and the steering angle δ are known, the lateral tire forces are fully determined. However, only certain types of maneuvers satisfy the steady-state assumption. The most straightforward are steady-state tests, in which the inputs are held constant and the system reaches steady velocities. Despite their simplicity, such experiments can be time-intensive. As an alternative, ramp steer inputs at constant velocity can be employed, producing low accelerations and offering a more time-efficient approach.

In the case of miniature vehicles, the steering input is varied as gradually as possible to ensure the car remains within the constrained training area, while maintaining a constant duty cycle to minimize the influence of the drivetrain. These experiments are repeated for various duty cycles and for both left and right turning maneuvers.

Given the recorded velocities and steering angles across all experiments, the slip angles α_f and α_r , as well as the lateral forces $F_{f,y}$ and $F_{r,y}$, can be computed at each time step. This results in a cloud of data points representing lateral forces at their corresponding slip angles. A nonlinear least-squares optimization problem can then be formulated, in which the parameters of the Pacejka tire model are identified by

Table 4.6 Identified Pacejka parameters for front and rear tires

Parameters	Front	Rear	Std. dev. Front/Rear
B	5.85	7.5	0.7/0.65
C	1.4	1.38	0.05/0.06
D	3.3	3.73	0.2/0.2

minimizing the 2-norm of the difference between the measured forces and those predicted by the model, as described in Eq. 4.27:

$$\min_{x_p} \frac{1}{2} \|y_i - f(x_i, x_p)\|^2 \quad (4.29)$$

where y_i are the measured forces, x_i the measured inputs of f and $x_p = [B_{f,r}, C_{f,r}, D_{f,r}]$ are the parameters of the Pacejka formula to be identified. Their values are listed in Tab. 4.6. Figure 4.7 presents the lateral forces computed using Eq. 4.28 alongside those predicted by the identified tire model. From this comparison, two key observations can be made. First, the vehicle exhibits a certain degree of asymmetry, particularly noticeable at higher slip angles. Second, consistent with the findings reported in [64], the experimental maneuvers predominantly excited the front tires into their saturated region, whereas the rear tires remained mostly in the linear regime. As a result, the front tire model shows a reasonably accurate fit, while the rear tire model appears less reliable under saturation conditions. It is worth noting that the spread of the estimated forces in Fig. 4.7 is comparable to the results obtained in [64], where a full-scale vehicle was used.

However, the prediction model used within the controller can be simplified to favor a faster optimization process, as proposed by Werling et al. [66], and is expressed as follows:

$$\psi = \frac{v}{l \left[1 + \left(\frac{v}{v_{ch}} \right)^2 \right]} \delta \quad (4.30)$$

Equation 4.30 originates from the steady-state yaw dynamics of the linear single-track vehicle model [53], whose key parameter is the so-called characteristic speed v_{ch} :

$$v_{ch} = \sqrt{\frac{l^2 C_{\alpha f} C_{\alpha r}}{m (C_{\alpha r} l_r - C_{\alpha f} l_f)}} \quad (4.31)$$

where $C_{\alpha f}$ and $C_{\alpha r}$ represent the cornering stiffness of the front and rear tires, respectively. This model provides two main advantages over the conventional bicycle model. First, it is free from singularities at $v = 0$, allowing full deceleration to a complete stop. Second, its differential equations are not particularly stiff, which permits the use of relatively large integration steps in numerical simulations, significantly reducing computational load. Nevertheless, to avoid overloading the control loop during aggressive driving maneuvers, additional constraints must be applied to the trajectory—such as limiting the vehicle’s maximum achievable lateral acceleration—which is addressed in the following Section.

In this context, a non-linear least squares optimization problem can be formulated, where the cornering stiffness values $C_{\alpha f}$ and $C_{\alpha r}$ in Eq. 4.31 are identified. With

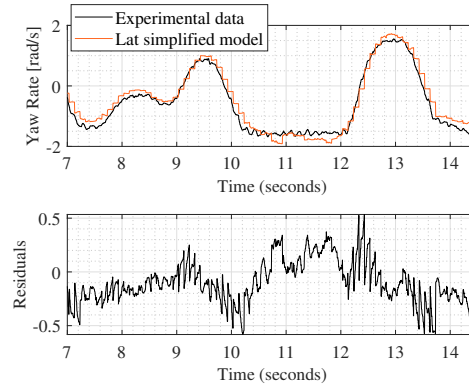


Fig. 4.8 Parameter identification of the simplified lateral dynamics model

reference to Eq. 4.30, the parameters $C_{\alpha f}$ and $C_{\alpha r}$ are estimated to be 27.03 and 43.53 $[N/\text{rad}]$, respectively. Considering that $C_{\alpha f} = B_f C_f D_f$ and $C_{\alpha r} = B_r C_r D_r$, the estimated front tire cornering stiffness is consistent with both formulation, yielding matching values. For $C_{\alpha r}$, however, a difference of approximately 5 $[N/\text{rad}]$ is observed between the two lateral dynamics models, which can likely be attributed to measurement uncertainties and errors. Despite its reduced complexity, the simplified lateral dynamics model exhibits a high level of fidelity in replicating experimental data as shown in Fig. 4.8, as evidenced by an R-squared value of 0.95 and an RMSE of 0.21. It is worth emphasizing that the dynamic test shown in Fig. 4.8 is conducted while avoiding tire saturation, in order to prevent the onset of nonlinear effects that are difficult to capture within the framework of this simplified modeling approach.

Table 4.7 Parameters defining the single-track vehicle model and their dimensions.

Parameters	Dimension
Mass (m)	$[M]$
Distance CoG to front axle (l_f)	$[L]$
Distance CoG to rear axle (l_r)	$[L]$
Wheelbase (l)	$[L]$
Front Cornering Stiffness (C_f)	$[ML/T^2]$
Rear Cornering Stiffness (C_r)	$[ML/T^2]$
Velocity (v)	$[L/T]$
Yaw moment of inertia (I_z)	$[M/L^2]$

4.4 Dynamics Similarity

To investigate dynamic similarity, the Buckingham Π theorem is utilized. This fundamental principle in dimensional analysis states that if a physically meaningful equation involves n physical variables, it can be reformulated using a set of $p = n - k$ dimensionless groups, denoted as $\Pi_1, \Pi_2, \dots, \Pi_p$. These dimensionless parameters are derived from the original variables, where k represents the number of fundamental physical dimensions, determined as the rank of a specific matrix related to the problem.

The first step involves collecting the relevant vehicle parameters and organizing them into dimensionless groups, known as Π parameters. This grouping is achieved by normalizing mass, time, and length using scaling factors that are directly related to the vehicle's mass, length, and velocity. By applying the theorem to the single-track vehicle model, 8 $[n]$ physical variables are obtained, listed in Table 4.7. $[M]$ indicates the mass, $[L]$ is the length and $[T]$ represents the time. These 3 define the fundamental physical dimensions k . Following Buckingham Π theorem, the number of dimensionless groups p is 5. These Π groups are shown in equation 4.32.

$$\Pi_1 = \frac{l_f}{l} \quad \Pi_2 = \frac{l_r}{l} \quad \Pi_3 = \frac{C_f l}{mv^2} \quad \Pi_4 = \frac{C_r l}{mv^2} \quad \Pi_5 = \frac{I_z}{ml^2} \quad (4.32)$$

By considering the parameters of the full-scale vehicle on which the 1:10 scale model is based, it is possible to verify the consistency of the scaled vehicle's parameters. The only available data refer to a wheelbase of 2.51 m, a mass of 1190 kg, and

a front/rear % weight distribution of 48/52, i.e. l_f 1.31 m and l_r 1.20 m. Therefore, validation can only be performed for the first two Π groups.

The first dimensionless parameter, Π_1 , is 0.52. Multiplying this by the wheelbase of the scaled model, as reported in Table 4.3, yields a distance of the CoG to the front axle l_f of approximately 0.14 m. Similarly, Π_2 is 0.48; multiplying this by the wheelbase gives a distance of the CoG to the rear axle b of 0.12 m, which aligns with measurements taken from the real-scaled model. As for the remaining parameters, no further verification can be carried out due to the lack of information on cornering stiffnesses and moment of inertia. The primary aim of this analysis is to emphasize the custom-designed, low-intrusiveness electronic system implemented on the scaled vehicle, which ensures dynamic similarity with a full-scale 1:1 counterpart.

Chapter 5

Car motion control problem

Based on the previous Chapter 4, a receding horizon racing car controller is presented in this Chapter. The schematic of the general control scheme is presented in Fig. 5.1. The vehicle states X, Y, Ψ extracted by the motion capture system, are then estimated using a Kalman filter. The controller is a Nonlinear Model Predictive Control (NMPC) path-tracking controller. Given the reference path (the track curvature as a function of the travelled distance), the controller also receives the position of static and dynamic agents to enable obstacle avoidance functionality. The computed control signal $acc_{x,target}$ is converted into duty cycle command $duty_{EM}$ via a low-level acceleration control block. This duty cycle command, together with the steering angle command δ is transmitted to the vehicle via a 2.4 GHz wireless communication. The entire framework is implemented in MATLAB/Simulink and runs in real time on a dSPACE SCALEXIO AutoBox [67], which features an Intel i7-6820EQ quad-core processor operating at 2.8 GHz.

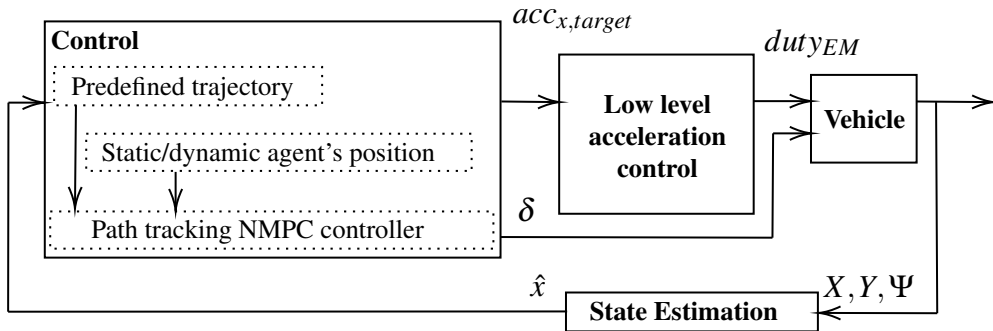


Fig. 5.1 General control scheme of the scaled-vehicle.

The controller is designed to race around a track with known boundaries and uses the vehicle model for longitudinal and lateral dynamics presented in Section 4.2 and 4.3. The Model Predictive Contouring Controller (MPCC), a widely utilized algorithm for planning and control in autonomous racing, is briefly reviewed herein, as presented in [34, 68]. This controller is designed to optimize progress along a specified reference trajectory while ensuring compliance with track boundary constraints. The associated control task is formulated as a nonlinear optimization problem. The structure of the chapter is as follows: Section 5.1 presents the general problem formulation and two case studies, the racing- and the passenger-car-oriented formulations. Section 5.2 discusses the obstacle avoidance functionality and Section 5.3 analyzes the real-time implementation.

5.1 Problem formulation

Usually, the architecture for an autonomous vehicle includes a perception module, a planning module, a control module, a localization module, and a database module [41]. Each module plays a role in the operation of the CAV. In use, the planning module receives data representing a destination and determines data representing a trajectory (sometimes referred to as a route) that can be traveled by the CAV to reach (e.g., arrive at) the destination. In order for the planning module to determine the data representing the trajectory, the planning module receives data from the perception module, the localization module, and the database module. The perception module identifies nearby physical objects using one or more sensors, e.g. camera, radar or LiDAR. The objects are classified (e.g., grouped into types such as pedestrian, bicycle, automobile, traffic sign, etc.), and a scene description including the classified objects is provided to the planning module. These data are updated at a frequency of 10 Hz, as they are derived from the sampling rate of the sensors used by the perception module. The planning module also receives data representing the CAV position from the localization module. The localization module determines the CAV position by using data from the sensors and data from the database module (e.g., a geographic data) to calculate a position. For example, the localization module uses data from a GNSS (Global Navigation Satellite System) sensor and geographic data to calculate the longitude and latitude of the CAV. In an embodiment, data used by the localization module includes high-precision maps of the roadway geometric

properties, maps describing road network connectivity properties, maps describing roadway physical properties (such as traffic speed, traffic volume, the number of vehicular and cyclist traffic lanes, lane width, lane traffic directions, or lane marker types and locations, or combinations of them), and maps describing the spatial locations of road features such as crosswalks, traffic signs or other travel signals of various types. The control module receives the data representing the trajectory and the data representing the CAV position and operates the control functions (e.g., steering, throttling, braking) of the CAV in a manner that will cause the CAV to travel the trajectory to the destination. For example, if the trajectory includes a left turn, the control module will operate the control functions in a manner such that the steering angle of the steering function will cause the CAV to turn left and the throttling and braking will cause the CAV to pause and wait for passing pedestrians or vehicles before the turn is made.

In the context of autonomous driving on a race track, the problem is significantly simplified by the fact that the start point, end point, and the complete geometry of the track are known a priori. Under these conditions, the control task is typically formulated as a path tracking problem based on a pre-computed reference path. When the controller is provided with a spatial representation of the track, such as a curvature profile parameterized by the arc length, e.g. s Frenet coordinate, the formulation aligns with the framework of Model Predictive Contouring Control (MPCC). MPCC is a specialized form of Model Predictive Control (MPC) in which the objective extends beyond temporal tracking of a trajectory, focusing instead on minimizing spatial errors. These are commonly defined as contouring error, representing the lateral deviation from the reference path, and lag error, quantifying the longitudinal discrepancy along the path.

This approach occupies an intermediate position between classical path tracking and full path planning. While the reference path remains fixed, the controller retains a degree of flexibility in determining how to traverse it optimally. In this sense, the MPCC does not involve generating a new path in real-time but allows limited deviation within predefined bounds, enabling performance improvements such as tighter cornering or smoother trajectories. As such, MPCC provides a more adaptable and performance-oriented alternative to conventional tracking methods, without the complexity of online path planning.

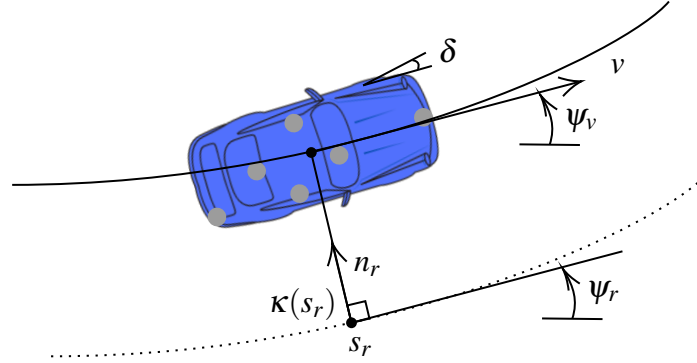


Fig. 5.2 Curvilinear prediction model

Due to the limited size of the test area and the reduced number of cameras composing the motion capture system, 1:10 scale vehicles cannot operate at high speeds, specifically, speeds above 3 m/s. Consequently, the test track has been designed to resemble an urban driving scenario. Since the vehicle operates well below its handling limits, a simplified prediction model has been adopted. In particular, a singularity-free kinematic model [69] was selected to avoid infeasibility issues at low speeds, especially near zero velocity.

As stated in [70], the kinematic model discretized at 200 ms achieves performance comparable to that of a dynamic model discretized at 100 ms. Moreover, their analysis highlights that the kinematic model exhibits lower prediction errors when discretized at 200 ms compared to 100 ms. These findings support the use of an MPC framework based on a kinematic bicycle model for autonomous vehicle control. While many existing MPC approaches [71] rely on high-fidelity models that incorporate tire-road interactions, the proposed method requires less computational effort and remains effective across a wide range of speeds, including standstill conditions.

The internal model is expressed through the following continuous-time formulation:

$$\dot{x}(t) = f(x(t), u(t)) \quad (5.1)$$

The state vector x is:

$$x = \begin{bmatrix} \delta & v & n_r & \psi_e & s_r \end{bmatrix}^\top \quad (5.2)$$

where δ is the steering angle, v is the longitudinal velocity, n_r represents the orthogonal distance from the path in Frenet coordinates, ψ_e is the relative heading angle

between the scaled car ψ_v and the reference path ψ_r , s_r denotes the progression along the reference path, as shown in Fig. 5.2. The control action vector u is defined as:

$$u = \begin{bmatrix} a_{x,target} & \Delta\delta \end{bmatrix}^\top \quad (5.3)$$

where $a_{x,target}$ is the longitudinal requested acceleration and $\Delta\delta$ is the variation of steering angle command. Compared to conventional approaches [34, 58, 59], where longitudinal dynamics is typically controlled using the Duty cycle D , this method aims for a more generic approach by directly using a target value of longitudinal acceleration $a_{x,target}$ as input. This approach enables the deployment of the same controller, originally validated on scaled vehicle models, on full-scale vehicles without modification. The procedure for converting the target longitudinal acceleration into a duty cycle suitable for PWM-based control of the electric motor will be detailed in Section 6.

The curvilinear internal model is expressed by:

$$\begin{aligned} \dot{\delta} &= \Delta\delta \\ \dot{v} &= a_{x,target} \\ \dot{n}_r &= v \sin(\psi_e) \\ \dot{\psi}_e &= \frac{v}{L} \tan(\delta) - \frac{v \cos(\psi_e)}{1 - n_r \kappa(s_r)} \kappa(s_r) \\ \dot{s}_r &= \frac{v \cos(\psi_e)}{1 - n_r \kappa(s_r)} \end{aligned} \quad (5.4)$$

where $\kappa_r(s_r)$ represents the curvature with respect to reference path arc length Frenet coordinate s_r . The optimal control problem (OCP) is of linear least squares type:

$$\begin{aligned} \min_{x(\cdot), u(\cdot)} \int_0^T & \underbrace{\frac{1}{2} \|V_x x(t) + V_u u(t) - y_{\text{ref}}\|_W^2}_{\text{Lagrange cost (stage cost)}} dt + \\ & \frac{1}{2} \begin{bmatrix} s_l(t) \\ s_u(t) \\ 1 \end{bmatrix}^\top \begin{bmatrix} Z_l & 0 & z_l \\ 0 & Z_u & z_u \\ z_l^\top & z_u^\top & 0 \end{bmatrix} \begin{bmatrix} s_l(t) \\ s_u(t) \\ 1 \end{bmatrix} dt + \\ & \underbrace{\frac{1}{2} \|V_x^e x(T) - y_{\text{ref}}^e\|_{W^e}^2}_{\text{Mayer cost (terminal cost)}} \end{aligned} \quad (5.5)$$

subject to

$$\dot{x}(t) = f(x(t), u(t)) \quad (5.6)$$

$$x(0) = x_0 \quad (5.7)$$

$$\underline{c}_x \leq J_{bx}x(t) + J_{sbx}s_{l,bx}(t) \quad (5.8)$$

$$J_{bx}x(t) - J_{sbx}s_{u,bx}(t) \leq \overline{c}_x \quad (5.9)$$

$$\underline{u} \leq J_{bu}u(t) \quad (5.10)$$

$$J_{bu}u(t) \leq \overline{u} \quad (5.11)$$

where x_0 is the initial state vector, $\underline{c}_x$ and $\overline{c}_x$ denote the vector of lower and upper bounds on the state constraints respectively:

$$\begin{aligned} \underline{c}_x &= \begin{bmatrix} \delta_{min} & v_{min} & n_{min} \end{bmatrix}^\top, \\ \overline{c}_x &= \begin{bmatrix} \delta_{max} & v_{max} & n_{max} \end{bmatrix}^\top \end{aligned} \quad (5.12)$$

J_{bx} is the state selection matrix that selects to which states the bounds are applied to, while J_{sbx} determines which of the state constraints can be softened using slack variables $s_{l,bx}(t)$ $s_{u,bx}(t)$ respectively, i.e., allowing small violations with penalties

$$J_{bx} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}, \quad J_{sbx} = I_3, \quad (5.13)$$

$\underline{u}$ and $\overline{u}$ represent the vector of lower and upper bounds on the control actions constraints, respectively:

$$\begin{aligned} \underline{u} &= \begin{bmatrix} a_{x,target,min} & \Delta\delta_{min} \end{bmatrix}^\top, \\ \overline{u} &= \begin{bmatrix} a_{x,target,max} & \Delta\delta_{max} \end{bmatrix}^\top \end{aligned} \quad (5.14)$$

J_{bu} is the control action selection matrix that selects which control action the bounds are applied to

$$J_{bu} = I_2 \quad (5.15)$$

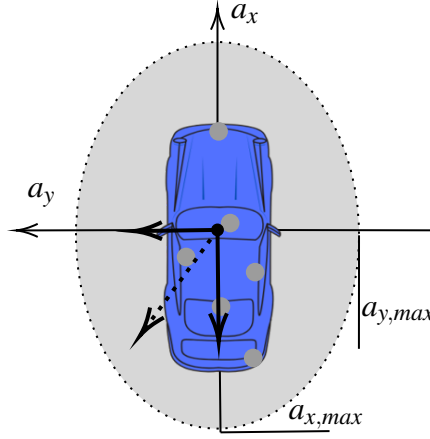


Fig. 5.3 Kamm's friction circle

Since the total transferable force of a tire cannot exceed a certain maximum value (the radius of the so-called friction circle, or Kamm's circle, as shown in Fig. 5.3), the vehicle's acceleration must also be limited. The introduction of this constraint serves to compensate for the limitations inherent in the model described in Eq. 5.4, thereby preventing the vehicle from operating beyond its handling limits, where tire saturation would occur. Thus, an additional non-linear constraint is added:

$$\begin{aligned} \underline{h_{a_{x,y}}} &\leq h_{a_{x,y}}(x(t), u(t)) + J_{sh}s_{l,h}(t) \\ h_{a_{x,y}}(x(t), u(t)) - J_{sh}s_{u,h}(t) &\leq \overline{h_{a_{x,y}}} \end{aligned} \quad (5.16)$$

where $h_{a_{x,y}}(x(t), u(t))$ is the non-linear constraint function represented by the Kamm's friction circle function, $s_{l,h}(t)$ and $s_{u,h}(t)$ are the slack variables for lower and upper bound, while $\underline{h_{a_{x,y}}}$ and $\overline{h_{a_{x,y}}}$ are the lower and upper bounds for the non-linear constraint function respectively

$$h_{a_{x,y}}(x(t), u(t)) = \left(\frac{a_x(t)}{a_{x,max}} \right)^2 + \left(\frac{a_y(t)}{a_{y,max}} \right)^2, \quad \underline{h_{a_{x,y}}} = 0, \quad \overline{h_{a_{x,y}}} = 1 \quad (5.17)$$

with $a_{x,max}$ the maximum longitudinal acceleration, $a_{y,max}$ the maximum lateral acceleration. J_{sh} is the selection matrix for the non-linear constraint function

$$J_{sh} = I_1 \quad (5.18)$$

The Lagrange cost term includes all the state and control actions components inside the least squares cost, being:

$$V_x = \begin{bmatrix} I_{5 \times 5} \\ 0_{2 \times 5} \end{bmatrix}, \quad V_u = \begin{bmatrix} 0_{5 \times 2} \\ I_{2 \times 2} \end{bmatrix} \quad (5.19)$$

Moreover, the Mayer cost contains all the final state components

$$V_x^e = I_{5 \times 5} \quad (5.20)$$

y_{ref} and y_{ref}^e represent the reference trajectory for states and controls and the terminal reference (at time T), respectively. W is the positive semi-definite weighting matrix and W^e is the terminal weighting matrix. Penalization for the use of slack variables $s_l(t)$ and $s_u(t)$ (for lower and upper bounds, respectively) is also included in the problem formulation by defining L2 terms Z_l and Z_u :

$$\begin{aligned} Z_l &= \text{diag} \left(w_{\delta_{\min}}, w_{v_{\min}}, w_{n_{\min}}, w_{h_{\min}} \right), \\ Z_u &= \text{diag} \left(w_{\delta_{\max}}, w_{v_{\max}}, w_{n_{\max}}, w_{h_{\max}} \right) \end{aligned} \quad (5.21)$$

where $w_{\delta_{\min}}$ and $w_{\delta_{\max}}$ are the penalty weights for the slacks on lower and upper constraint for the steering angle δ , $w_{v_{\min}}$ and $w_{v_{\max}}$ for the velocity v , $w_{n_{\min}}$ and $w_{n_{\max}}$ for the orthogonal deviation n , $w_{h_{\min}}$ and $w_{h_{\max}}$ for the non-linear constraint $h(x(t), u(t))$. Linear penalty terms z_l and z_u are set to zero. Since the proposed path tracking control aims to follow a desired path rather than reach a specific endpoint, minimizing deviation from the path along all the points of the trajectory, the terminal cost has the same weight as the stage cost. Finally, slack variables are not used to soften the terminal costs. Moreover, terminal constraints are not enforced. In goal-reaching problems, such as scenarios of autonomous parking, terminal constraints must be strictly imposed to force the system to reach a specific state; violating them could lead to unsafe outcomes, such as possible collisions. Thus, for path tracking problem formulation, the solver is allowed greater flexibility at the terminal time T , meaning that bounds on states or nonlinear constraints at the final time can remain undefined.

In summary, the constraints include upper and lower bounds on system states such as steering angle δ , vehicle speed v , and lateral deviation n . The steering constraint enforces the physical limitations of the steering system in both directions. The speed

constraint restricts the vehicle's velocity to a minimum of 0 m/s and a maximum of 3 m/s. This upper bound is due to the limited length of the test track and the design of the ESC, which controls only traction and braking through PWM. Importantly, the ESC does not support reverse motion, meaning that the scaled vehicle cannot drive backward. In addition to these state constraints, bounds are imposed on the rate of change of control inputs: namely, the longitudinal acceleration command $a_{x,target}$ and the change in steering angle $\Delta\delta$. These constraints are introduced to manage actuator effort and ensure smooth and stable system behavior. However, bounding the longitudinal acceleration alone does not guarantee that the vehicle will remain within its handling limits. To address this, an additional nonlinear constraint has been incorporated. This constraint captures the combined tire force limitations and prevents the vehicle from exceeding its feasible acceleration envelope, particularly in scenarios where tire saturation might occur. This safety constraint is modeled using Kamm's friction circle, which effectively limits the total available tire force and ensures realistic tire-ground interaction. For path tracking, the cost function includes terms penalizing lateral deviation and heading error, consistent with established approaches in the literature [71, 72].

5.1.1 Racing-oriented optimal control formulation

The formulation of the motion control problem allows for the definition of different reference objectives depending on the specific goal and the application context. In a racing-oriented approach, the controller must be capable of minimizing the time and space required to follow the trajectory while strictly adhering to the constraints on states, control inputs, and system nonlinearities. In this context, the cost function prioritizes forward progress along the curvilinear reference path while penalizing lateral and yaw errors to maintain close adherence to the racing line. Referring to the formulation in Eq. 5.5, the problem can therefore be expressed as follows:

$$\begin{aligned}
& \min_{x(\cdot), u(\cdot)} \int_0^T \frac{1}{2} \|x(t) - x_{ref}\|_Q^2 + \frac{1}{2} \|u(t)\|_R^2 dt + \\
& \frac{1}{2} \begin{bmatrix} s_l(t) \\ s_u(t) \\ 1 \end{bmatrix}^\top \begin{bmatrix} Z_l & 0 & z_l \\ 0 & Z_u & z_u \\ z_l^\top & z_u^\top & 0 \end{bmatrix} \begin{bmatrix} s_l(t) \\ s_u(t) \\ 1 \end{bmatrix} dt + \\
& \frac{1}{2} \|x(T) - x_{ref}^e\|_{Q_e}^2
\end{aligned} \tag{5.22}$$

where x_{ref} is the reference state trajectory, Q is the weight matrix for state tracking over the horizon, R is the weight matrix for control effort, x_{ref}^e is the desired state at time T and Q_e is the weight matrix for terminal cost. In this problem formulation, the controller handles both longitudinal and lateral vehicle dynamics. Therefore, the reference state x_{ref} is defined in terms of the lateral deviation n , the heading error ψ_e , and the arc length progress s , enabling the problem to be reformulated with an objective of maximizing progress along the path. The reference values for lateral deviation n_{ref} and heading error $\psi_{e,ref}$ are set to zero, guiding the controller to follow the reference path. As for the arc length progress s_{ref} , it is defined such that the vehicle is encouraged to move forward toward a target located a few meters ahead. The steering angle δ_{ref} and velocity v_{ref} references are not included, as the weighting matrix Q assigns weights only to the three aforementioned states. This approach defines the evolution of the reference progress s_{ref} over the time horizon T as follows:

$$s_{ref}(t) = s_0 + (s_{ref,T} - s_0) \cdot \frac{t}{T} \tag{5.23}$$

where s_0 is the current progress on the track, $s_{ref,T}$ is the final arc length at time T , while the terminal reference progress s_{ref}^e is defined as

$$s_{ref}^e = s_0 + s_{ref,T} \tag{5.24}$$

In this context, a reference progress of $s_{ref,T} = 4 \text{ m}$ is considered, indicating that the controller is configured to advance the vehicle by 4 meters within the prediction horizon T . For a time horizon of $T = 2 \text{ s}$, this corresponds to a reference velocity of 2 m/s . In racing applications, the value of $s_{ref,T}$ is typically set to a significantly higher level, often exceeding the vehicle's actual capabilities, to encourage the controller to minimize both the distance traveled and the time required, while promoting

aggressive acceleration behavior. The matrix Q has normalized penalties on state variables:

$$Q = \text{diag} \left(0, 0, 5 \cdot \left(\frac{1}{n_{\max}} \right)^2, 0.5 \cdot \left(\frac{1}{\psi_{dev}} \right)^2, 1 \cdot \left(\frac{1}{s_{dev}} \right)^2 \right) \quad (5.25)$$

where ψ_{dev} is the deviation from the reference heading angle, s_{dev} is the deviation from reference progress. Weights are normalized with respect to their respective maximum allowable deviation. It is worth mentioning that the zero weights for δ and v mean deviations in those states are not penalized.

Furthermore, the weight matrix of the terminal cost for state tracking Q^e is assumed equal to Q . The weight matrix for control effort R is defined as:

$$R = \text{diag} \left(100 \cdot \left(\frac{1}{a_{x,target,max}} \right)^2, 10 \cdot \left(\frac{1}{\Delta \delta_{max}} \right)^2 \right) \quad (5.26)$$

Weights are normalized to their respective maximum allowable value. Penalties on slack variables $s_l(t)$ and $s_u(t)$ are set:

$$\begin{aligned} Z_l &= \text{diag} \left(10^6 \cdot \left(\frac{1}{\delta_{\min}} \right)^2, 10^6 \cdot \left(\frac{1}{v_{\min}} \right)^2, 10^6 \cdot \left(\frac{1}{n_{\min}} \right)^2, 10^4 \cdot \left(\frac{1}{h_{a,y}} \right)^2 \right), \\ Z_u &= \text{diag} \left(10^6 \cdot \left(\frac{1}{\delta_{\max}} \right)^2, 10^6 \cdot \left(\frac{1}{v_{\max}} \right)^2, 10^6 \cdot \left(\frac{1}{n_{\max}} \right)^2, 10^4 \cdot \left(\frac{1}{h_{a,y}} \right)^2 \right) \end{aligned} \quad (5.27)$$

Figures 5.4 and 5.5 show the behaviour of the proposed racing-oriented optimal motion control problem. The simulation starts from an initial condition where the vehicle has a lateral deviation of $n = 0.4$ m and a longitudinal velocity v of 2 m/s. Figure 5.5a shows the distribution of the vehicle's scaled acceleration components in the longitudinal a_x and lateral a_y directions. The dashed circle represents the admissible acceleration envelope, which accounts for the physical limits of tire-road interaction, represented by the non-linear constraint $h(x(t), u(t))$. The blue markers represent the actual vehicle acceleration during the simulation. Throughout the maneuver, the vehicle remains within the acceleration limits defined by the non-linear constraint of the control optimization problem.

Figure 5.5b reveals that the vehicle deviates significantly from the reference path, particularly in cornering sections. This behavior is expected, as the progress-maximizing controller intentionally cuts corners to reduce the traveled distance. The color-coded velocity profile shows almost constant longitudinal speeds of 2 m/s in

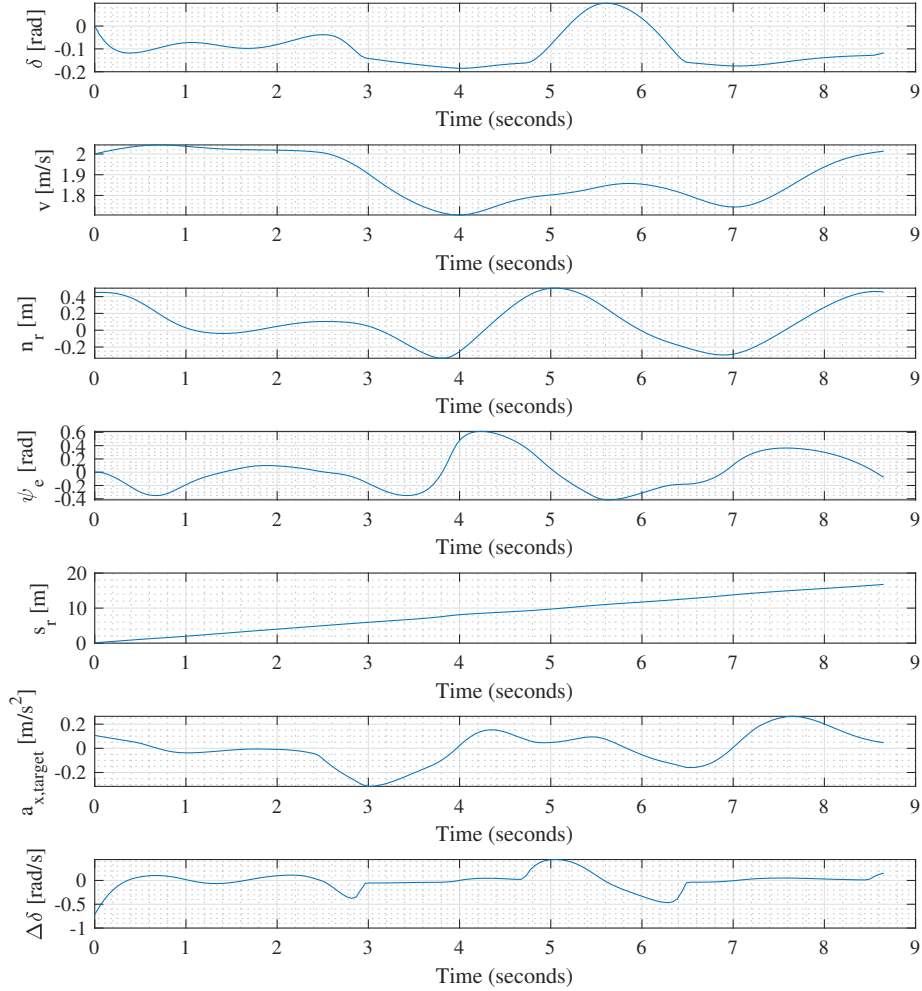
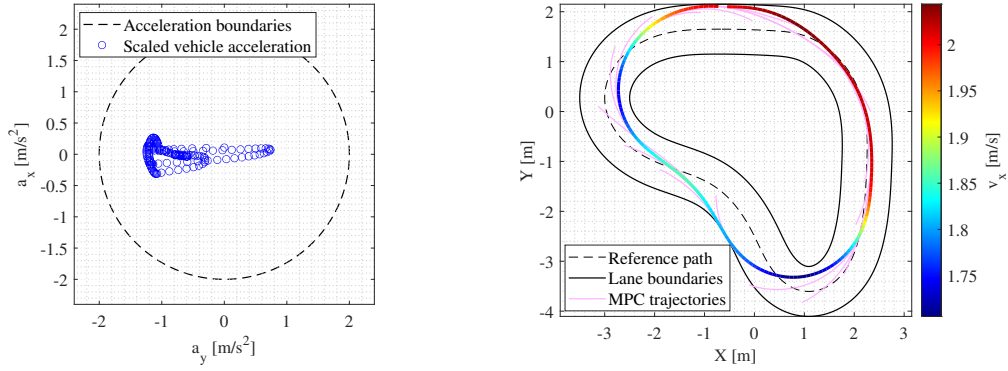


Fig. 5.4 States and control commands evolution along the circuit

order to track the reference progress of 4 m, especially along straights and wide turns, indicating a more performance-driven driving style. By appropriately weighting the progress s relative to the lateral deviation n and heading angle ψ , the controller is able to cut through corners while consistently staying within the track boundaries.



(a) Longitudinal and lateral acceleration plot (b) Poses and velocity profile along the race track

Fig. 5.5 Racing-oriented motion control behaviour

5.1.2 Passenger-car-oriented optimal control formulation

In the context of urban or extra-urban driving scenarios, the objective of the optimization problem is no longer to maximize progress, but rather to track a reference velocity, which is typically provided by the path planning module [41]. In this class of problems, greater emphasis is placed on minimizing the lateral deviation from the reference path n_r , as well as the relative heading angle ψ_r . Given that the trajectory is predefined in terms of curvature $\kappa(s)$, the path tracking controller is required to follow the path generated by the planning module strictly. In this case, the reference state x_{ref} is defined in terms of the lateral deviation n , the heading error ψ_e , and the velocity v , enabling the problem to be reformulated with a pure reference tracking problem. The cost function still penalizes lateral and heading deviations, but with greater emphasis on lateral error. This reflects the goal of keeping the vehicle tightly aligned with the reference path, which is more critical for comfort, safety, and accuracy in non-racing applications. The reference values for lateral deviation n_{ref} and heading error $\psi_{e,ref}$ are set to zero, guiding the controller to follow the reference path, while the reference velocity v_{ref} , allow the optimal control problem to behave in a smoother and more conservative manner, i.e. in terms of longitudinal and lateral perceived accelerations, suitable for road use. The matrix Q has normalized penalties on state variables:

$$Q = \text{diag} \left(0, \quad 1 \cdot \left(\frac{1}{v_{dev}} \right)^2, \quad 50 \cdot \left(\frac{1}{n_{max}} \right)^2, \quad 25 \cdot \left(\frac{1}{\psi_{dev}} \right)^2, \quad 0 \right) \quad (5.28)$$

Weights are normalized with respect to their respective maximum allowable deviation. Furthermore, the weight matrix of the terminal cost for state tracking Q^e is assumed equal to Q . The weight matrix for control effort R is defined as:

$$R = \text{diag} \left(100 \cdot \left(\frac{1}{a_{x,target,max}} \right)^2, 10 \cdot \left(\frac{1}{\Delta \delta_{max}} \right)^2 \right) \quad (5.29)$$

Weights are normalized to their respective maximum allowable value. Penalties on slack variables $s_l(t)$ and $s_u(t)$ are set:

$$\begin{aligned} Z_l &= \text{diag} \left(10^6 \cdot \left(\frac{1}{\delta_{min}} \right)^2, 10^6 \cdot \left(\frac{1}{v_{min}} \right)^2, 10^6 \cdot \left(\frac{1}{n_{min}} \right)^2, 10^4 \cdot \left(\frac{1}{h_{a,y}} \right)^2 \right), \\ Z_u &= \text{diag} \left(10^6 \cdot \left(\frac{1}{\delta_{max}} \right)^2, 10^6 \cdot \left(\frac{1}{v_{max}} \right)^2, 10^6 \cdot \left(\frac{1}{n_{max}} \right)^2, 10^4 \cdot \left(\frac{1}{h_{a,y}} \right)^2 \right) \end{aligned} \quad (5.30)$$

Figures 5.6 and 5.7 illustrate the behavior of the proposed passenger-car-oriented optimal motion control formulation. The simulation starts from an initial condition where the vehicle has a lateral deviation of $n = 0.4$ m and a longitudinal velocity v of 2 m/s, as previous case. For this passenger-car-oriented formulation, the controller successfully maintains the vehicle along the reference path, tracking the imposed reference velocity, as shown in Fig.5.6. Assuming a road scenario, the velocity is set to 1.38 m/s, which corresponds to approximately 50 km/h at a 1:1 scale. By assigning greater weight to the lateral deviation from the reference path n_r and the heading angle error ψ_e , the controller is able to follow closely the reference trajectory. Figure 5.7a shows the distribution of the vehicle's scaled acceleration components in the longitudinal a_x and lateral a_y directions. The dashed circle represents the admissible acceleration envelope, which accounts for the physical limits of tire-road interaction, represented by the non-linear constraint $h(x(t), u(t))$. The blue markers represent the actual vehicle acceleration during the simulation. It can be observed that the acceleration demands remain within the feasible boundaries, almost touching the negative lateral acceleration one. Unlike the race-car scenario, where lateral acceleration limits were not reached because of its turn-cutting strategy, the vehicle now executes tighter turns with smaller radii of curvature in order to closely follow the reference trajectory.

Figure 5.7b displays the reference path (dashed line), lane boundaries (black solid lines), and the resulting MPC trajectories (color-coded by longitudinal velocity

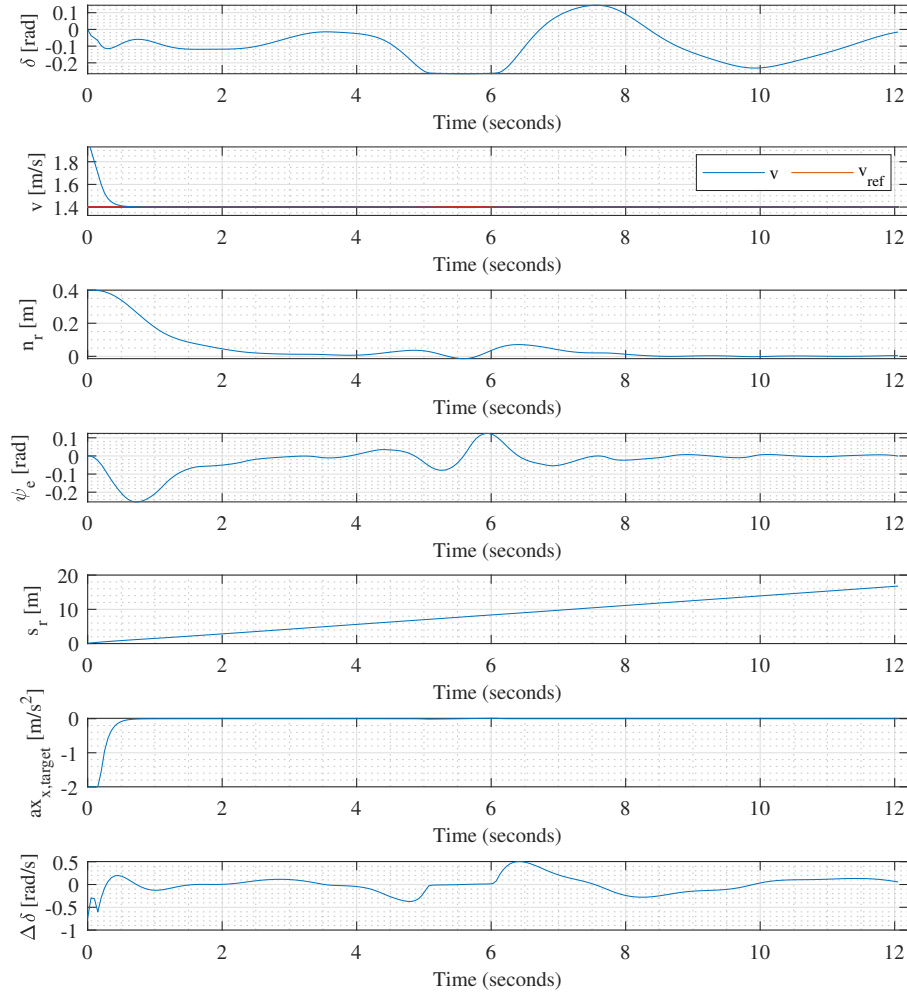
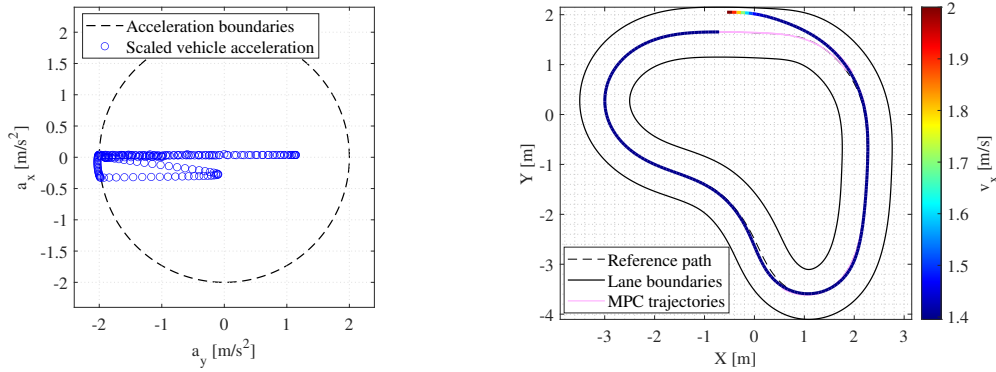


Fig. 5.6 States and control commands evolution along the circuit

v). Starting with the same initial condition as in the previous racing-oriented case, the controller immediately guides the vehicle toward the reference path. The color gradient indicates a moderate deceleration on the second right turn. The controller requests the deceleration in order to perform the turning maneuver following the reference path. On the remaining part of the circuit, the vehicle maintains constant velocity, tracking the reference velocity v_{ref} , set to 1.38 m/s. The controller effectively tracks the reference path while respecting constraints, highlighting its suitability



(a) Longitudinal and lateral acceleration plot (b) Poses and velocity profile along the race track

Fig. 5.7 Passenger-car-oriented motion control behaviour

for urban or extra-urban scenarios where safety, comfort, and path adherence are prioritized over time-optimal performance. This behavior contrasts with the scenario in which the progress term is weighted in the cost function, where the controller prioritizes rapid forward motion over strict path adherence.

5.2 Collision Avoidance

Collision Avoidance (CA) is a fundamental requirement for CAVs and has become a critical focus of research and development. As the deployment of CAVs continues to expand, ensuring their safe operation and preventing collisions with other vehicles, pedestrians, and static or dynamic obstacles has become paramount. To address this challenge, researchers have explored a variety of strategies, including obstacle detection, path planning (PP), path tracking control (PTC), risk prediction, vehicle dynamics control, and decision-making algorithms. These techniques enable CAVs to perceive their environment, recognize potential hazards, plan collision-free paths, predict and assess collision risks, dynamically adjust their motion, and make informed navigation decisions to enhance safety. In most cases, CA is integrated into the path planning module, which computes an obstacle-free trajectory subsequently tracked by the path tracking control system.

Various approaches to path planning for collision avoidance have been proposed. Graph-based planners [73] (e.g., Hybrid A*) discretize the state space, treating obstacles as invalid nodes to ensure obstacle-free path generation. These methods

are complete, meaning they are guaranteed to find a solution if one exists, but they suffer from discretization artifacts, producing paths that are often non-smooth and suboptimal from a kinematic perspective. As a result, post-processing is typically required to refine the trajectory.

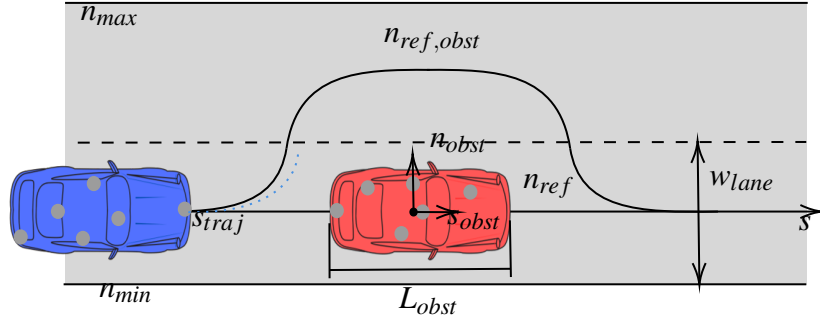
Sampling-based planners [74, 75] (e.g., Rapidly Random Exploring Trees (RRT, RRT*)) explore the free space probabilistically, naturally accommodating nonholonomic constraints during tree expansion. While RRT* is asymptotically optimal, improving the solution quality with increased sampling density, these methods often produce trajectories with discontinuities or excessive curvature, necessitating smoothing techniques to ensure safe and feasible path tracking.

Optimization-based methods [76–78] (e.g., trajectory optimization, Model Predictive Control (MPC)) formulate collision avoidance as a constrained optimization problem, embedding vehicle dynamics and obstacle constraints within the cost functional. These methods are capable of generating dynamically feasible and smooth trajectories and support reactive re-planning in response to environmental changes. However, they face significant challenges in real-time applications due to the nonconvexity of the problem, particularly in the presence of moving obstacles, often requiring warm-starting strategies or convex approximations to achieve computational feasibility.

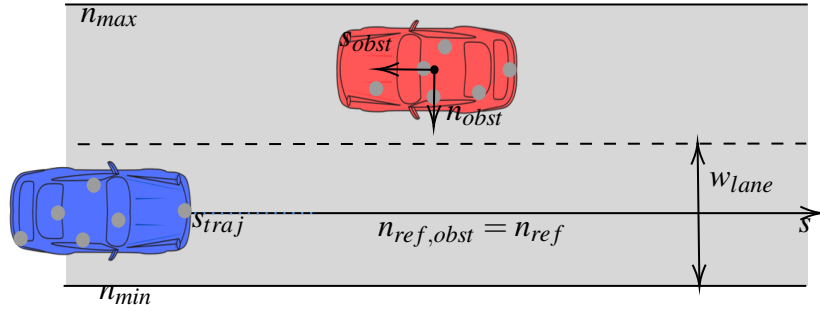
Interpolated curve methods [74] (e.g., splines, clothoids) construct paths by fitting continuous curves between waypoints, ensuring low curvature and bounded acceleration profiles. Although computationally efficient and suitable for ensuring smooth motion, these methods typically assume a quasi-static environment and lack predictive capabilities necessary for handling dynamic obstacles.

In the proposed case study, three distinct strategies were implemented. The first strategy consists in the dynamic modulation of the reference lateral deviation n_{ref} as a function of the real-time obstacle position, effectively adapting the target path to avoid collisions. The second and third strategies introduce an additional nonlinear constraint into the optimal control problem, explicitly formulated as a function of the obstacle's position, which is treated as an external parameter to the optimization problem, thereby enforcing obstacle avoidance directly within the trajectory generation and tracking process.

5.2.1 Reference lateral deviation generator logic



(a) Reference lateral deviation generator with dual logistic curve



(b) Reference lateral deviation generator is inactive when obstacle is located out of the lane width

Fig. 5.8 Reference lateral deviation generator logic

In this collision avoidance (CA) strategy, the lateral deviation reference $n_{ref,obst}$ is dynamically updated based on the predicted position of the obstacle, through an upstream algorithm that adjusts the reference n_{ref} prior to the MPC optimization. Specifically, a smooth lateral deviation profile is generated by means of a double logistic function, centered on the obstacle's progress arc length coordinate s_{obst} . This double logistic function is inspired by the ISO 3888-2 double lane-change test [79], scaled down by a factor of 10 as shown in Table 5.1.

The scaled layout of the double lane change maneuver is illustrated in Fig. 5.9. The ISO 3888-2-based maneuver is used as a reference only for the first obstacle avoidance scenario, in which the lateral deviation reference is defined based on the static/dynamic position of the obstacle. For the other cases considered, ISO 3888-2 is no longer applicable, as the obstacles are defined through non-convex functions inside the prediction model and the trajectory is directly selected by the NMPC controller.

Table 5.1 ISO 3888-2 Double lane change maneuver. Comparison between full scale and 1:10 scale

ISO 3888-2	Full Scale [m]	1:10 Scale [m]
Vehicle Width	2.6	0.26
Lane Width	3.5	0.35
Lenght of Entry Lane (L1)	12	1.2
Lenght of Transition Lane (L2)	13.5	1.35
Lenght of Exit Lane (L3)	11	1.1
Lenght of Exit Lane (L4)	12.5	1.25
Lenght of Exit Lane (L5)	12	1.2
Lateral offset of transition	3.5	0.35
Total Lenght	61	6.1

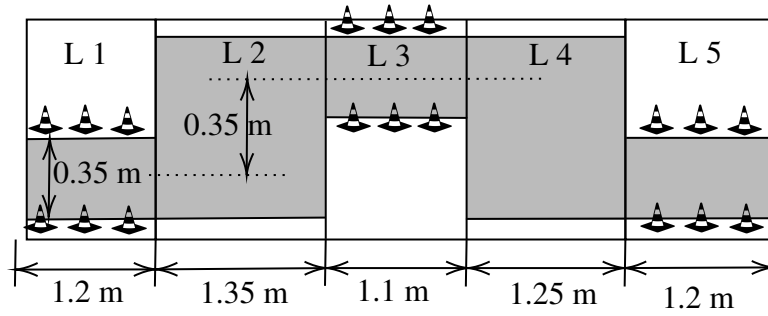


Fig. 5.9 ISO 3888-2 layout for scaled vehicle.

The fundamental concept is to define a continuous lateral offset along the path s , allowing for a progressive deviation away from the obstacle followed by a smooth rejoining to the nominal trajectory. The resulting lateral reference deviation $n_{ref,obst}$ is defined as:

$$n_{ref,obst}(s) = A_{\sigma} \left(\frac{1}{1 + e^{-k(s-s_1)}} - \frac{1}{1 + e^{-k(s-s_2)}} \right) \quad (5.31)$$

where

$$s_1 = s_{obst} - \frac{L_{obst}}{2}, \quad s_2 = s_{obst} + \frac{L_{obst}}{2} \quad (5.32)$$

The variable s_{obst} represents the progression of the obstacle along the path, L_{obst} is the length of the obstacle, A_{σ} is the lateral deviation amplitude (e.g., half lane width) and k is the steepness parameter that regulates the sharpness of the lateral transition generated by the logistic function. The lateral deviation is then evaluated along the

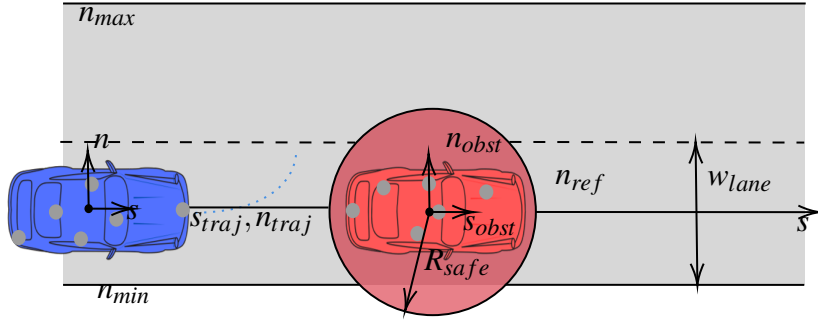


Fig. 5.10 Obstacle constraint using circle safe-region

predicted progress s_{traj} using an interpolation:

$$n_{ref,obs}(s_{traj}) = \mathcal{J}(s_{traj}; s_{vect}, n_{ref,obs}(s_{vect})) \quad (5.33)$$

where s_{vect} is a dense-grid of s -values over which $n_{ref,obs}$ is calculated. However, the lateral deviation reference adjustment is applied only if the progress trajectory s_{traj} is monotonically increasing (i.e., $\frac{ds_{traj}}{dt} \geq 0$) and the lateral deviation n_{obst} of the obstacle is within the width of the lane ($|n_{obst}| \leq \frac{w_{lane}}{2}$). Figure 5.8 illustrates the logic used to generate the reference lateral deviation.

5.2.2 Obstacle nonlinear constraint with circle-safe region

In the second collision avoidance (CA) approach, obstacle avoidance is directly embedded as an additional nonlinear constraint within the optimal control problem (OCP). Instead of adjusting the path reference dynamically as in the previous method, here the obstacle constraint is enforced during the optimization, ensuring that the predicted trajectory remains at a safe distance from obstacles. The OCP extends the vehicle dynamics by introducing obstacle parameters as external inputs p . The collision avoidance constraint is formulated using a circle-based safe region, as shown in Fig. 5.10. The vehicle must stay outside a circular region centered at (s_{obst}, n_{obst}) with a radius R_{safe} . Thus the corresponding additional non-linear constraint is:

$$\begin{aligned} \underline{h_{obst}} &\leq h_{obst}(x(t), p(t)) + J_{sh} s_{l,h}(t) \\ h_{obst}(x(t), p(t)) - J_{sh} s_{u,h}(t) &\leq \overline{h_{obst}} \end{aligned} \quad (5.34)$$

where $h_{obst}(x(t), p(t))$ is the non-linear constraint function represented by the circle safe-region function, $s_{l,h}(t)$ and $s_{u,h}(t)$ are the slack variables for lower and upper bound, while $\underline{h_{obst}}$ and $\overline{h_{obst}}$ are the lower and upper bounds for the non-linear constraint function respectively. J_{sh} is the selection matrix for the non-linear constraint function (i.e. identity matrix I_1). Thus, the obstacle non-linear constraint is:

$$\begin{aligned} h_{obst}(x(t), p(t)) &= (n(t) - n_{obst})^2 + (s(t) - s_{obst})^2, \\ \underline{h_{obst}} &= R_{safe}^2, \quad \overline{h_{obst}} = +\infty \end{aligned} \quad (5.35)$$

The matrix Q has normalized penalties on state variables:

$$Q = \text{diag} \left(0, \quad 1 \cdot \left(\frac{1}{v_{dev}} \right)^2, \quad 50 \cdot \left(\frac{1}{n_{max}} \right)^2, \quad 10 \cdot \left(\frac{1}{\theta_{dev}} \right)^2, \quad 0 \right) \quad (5.36)$$

Weights are normalized with respect to their respective maximum allowable deviation. Furthermore, the weight matrix of the terminal cost for state tracking Q^e is assumed equal to Q . The weight matrix for control effort R is defined as:

$$R = \text{diag} \left(100 \cdot \left(\frac{1}{a_{x,target,max}} \right)^2, \quad 10 \cdot \left(\frac{1}{\Delta \delta_{max}} \right)^2 \right) \quad (5.37)$$

Weights are normalized to their respective maximum allowable value. Penalties on slack variables $s_l(t)$ and $s_u(t)$ are set:

$$\begin{aligned} Z_l &= \text{diag} \left(10^6 \cdot \left(\frac{1}{\delta_{min}} \right)^2, \quad 10^6 \cdot \left(\frac{1}{v_{min}} \right)^2, \quad 10^6 \cdot \left(\frac{1}{n_{min}} \right)^2, \quad 10^4 \cdot \left(\frac{1}{h_{ax,y}} \right)^2, \quad 10^4 \cdot \left(\frac{1}{\underline{h_{obst}}} \right)^2 \right), \\ Z_u &= \text{diag} \left(10^6 \cdot \left(\frac{1}{\delta_{max}} \right)^2, \quad 10^6 \cdot \left(\frac{1}{v_{max}} \right)^2, \quad 10^6 \cdot \left(\frac{1}{n_{max}} \right)^2, \quad 10^4 \cdot \left(\frac{1}{h_{ax,y}} \right)^2, \quad 10^4 \cdot \left(\frac{1}{\overline{h_{obst}}} \right)^2 \right) \end{aligned} \quad (5.38)$$

The last component represents the slack penalty and normalization associated with the additional nonlinear constraint.

5.2.3 Obstacle nonlinear constraint with probabilistic region

In this approach, collision avoidance (CA) is not modeled as a hard geometric constraint (like a circle), but instead as a probabilistic (Gaussian) constraint on the vehicle's position relative to the obstacle. Thus, the corresponding additional

non-linear constraint is:

$$\begin{aligned} \underline{h}_{obst} &\leq h_{obst}(x(t), p(t)) + J_{sh}s_{l,h}(t) \\ h_{obst}(x(t), p(t)) - J_{sh}s_{u,h}(t) &\leq \overline{h}_{obst} \end{aligned} \quad (5.39)$$

where $h_{obst}(x(t), p(t))$ is the non-linear constraint function represented by the probabilistic safe-region function, $s_{l,h}(t)$ and $s_{u,h}(t)$ are the slack variables for lower and upper bound, while $\underline{h}_{obst}$ and $\overline{h}_{obst}$ are the lower and upper bounds for the non-linear constraint function respectively. The obstacle avoidance constraint is defined as:

$$h_{obst}(x(t), p(t)) = A_{obst} \exp \left(- \left(\frac{(s(t) - s_{obst})^2}{2\sigma_s^2} + \frac{(n(t) - n_{obst})^2}{2\sigma_n^2} \right) \right) \quad (5.40)$$

$$\underline{h}_{obst} = 0, \quad \overline{h}_{obst} = \beta_{obst}$$

where A_{obst} is the amplitude of the obstacle probabilistic safe region, σ_s is the standard deviation along the progress Frenet coordinate, σ_n is the standard deviation along the lateral deviation Frenet coordinate and β_{obst} is the confidence bound of the Gaussian distribution. Figure 5.11 illustrates the obstacle constraint using the proposed probabilistic safe region. The matrix Q has normalized penalties on state variables:

$$Q = \text{diag} \left(0, \quad 1 \cdot \left(\frac{1}{v_{dev}} \right)^2, \quad 50 \cdot \left(\frac{1}{n_{max}} \right)^2, \quad 10 \cdot \left(\frac{1}{\theta_{dev}} \right)^2, \quad 0 \right) \quad (5.41)$$

Weights are normalized with respect to their respective maximum allowable deviation. Furthermore, the weight matrix of the terminal cost for state tracking Q^e is assumed equal to Q . The weight matrix for control effort R is defined as:

$$R = \text{diag} \left(100 \cdot \left(\frac{1}{a_{x,target,max}} \right)^2, \quad 10 \cdot \left(\frac{1}{\Delta \delta_{max}} \right)^2 \right) \quad (5.42)$$

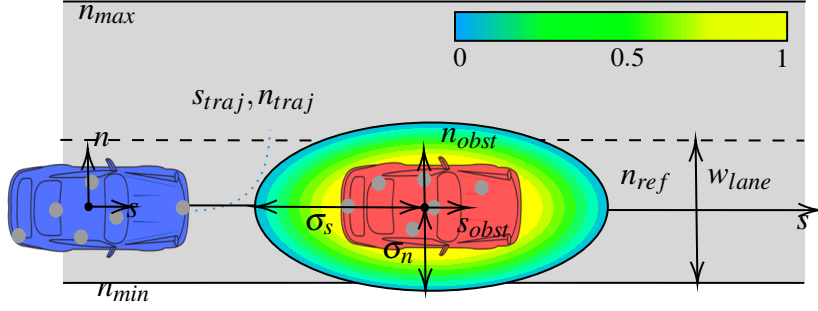


Fig. 5.11 Obstacle constraint using probabilistic region

Weights are normalized to their respective maximum allowable value. Penalties on slack variables $s_l(t)$ and $s_u(t)$ are set:

$$Z_l = \text{diag} \left(10^6 \cdot \left(\frac{1}{\delta_{\min}} \right)^2, 10^6 \cdot \left(\frac{1}{v_{\min}} \right)^2, 10^6 \cdot \left(\frac{1}{n_{\min}} \right)^2, 10^4 \cdot \left(\frac{1}{h_{a_{x,y}}} \right)^2, 10^4 \cdot \left(\frac{1}{h_{\text{obst}}} \right)^2 \right),$$

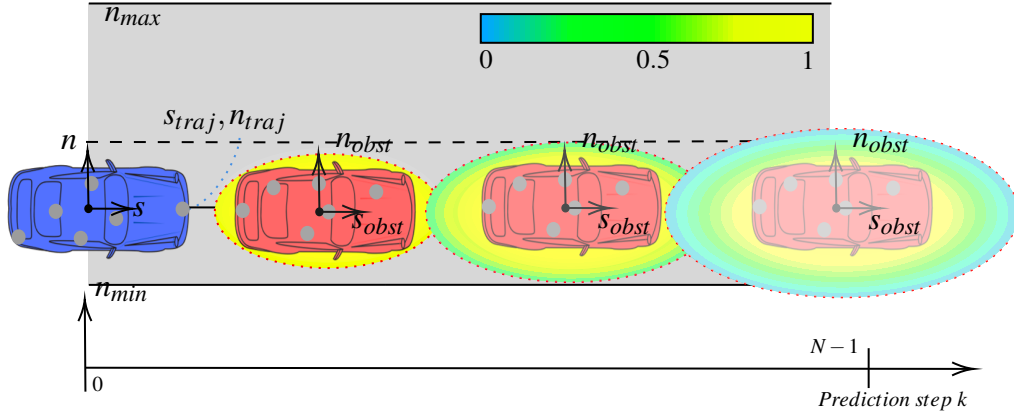
$$Z_u = \text{diag} \left(10^6 \cdot \left(\frac{1}{\delta_{\max}} \right)^2, 10^6 \cdot \left(\frac{1}{v_{\max}} \right)^2, 10^6 \cdot \left(\frac{1}{n_{\max}} \right)^2, 10^4 \cdot \left(\frac{1}{h_{a_{x,y}}} \right)^2, 10^4 \cdot \left(\frac{1}{h_{\text{obst}}} \right)^2 \right) \quad (5.43)$$

To improve robustness against prediction uncertainties, it is introduced a stage-varying constraint bound along the prediction horizon. In particular, the upper bound $\overline{h_{\text{obst},k}}$ on the obstacle avoidance constraint is made stricter at the beginning of the horizon (i.e. prediction index $k = 0$) and progressively relaxed toward the end ($k = N_{\text{steps}} - 1$). This choice reflects the fact that predictions close to the current time are more accurate, whereas uncertainty increases farther into the future. Figure 5.12a illustrates the variation of confidence bound of the probabilistic safe-region during prediction horizon evolution. Therefore, a tighter constraint is enforced near-term to guarantee immediate safety, while a looser constraint is allowed at long-term predictions to account for modeling errors and external disturbances. Mathematically, the constraint is written as

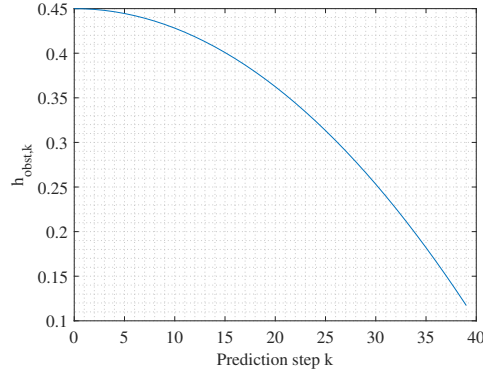
$$h(x_k, p_k) \leq \overline{h_{\text{obst},k}}, \quad k = 0, \dots, N_{\text{steps}} - 1 \quad (5.44)$$

where $\overline{h_{\text{obst},k}}$ is a deterministic, monotonically increasing function of the prediction step k ,

$$\bar{h}_k = \bar{h}_{\text{init}} + \left(\frac{k}{N_{\text{steps}}} \right)^\gamma (\bar{h}_{\text{fin}} - \bar{h}_{\text{init}}) \quad (5.45)$$



(a) Confidence bound decrease during prediction horizon evolution



(b) Stage-varying obstacle constraint upper bound

Fig. 5.12 Obstacle constraint using probabilistic region with dynamic constraint upper bound definition

where $\overline{h_{init}}$ is the confidence bound at initial prediction step k , $\overline{h_{fin}}$ is the relaxed bound at the end of the prediction horizon, and γ controls the rate of relaxation across the prediction horizon, as shown in Fig. 5.12b. A higher value of γ leads to a slower initial relaxation and a more abrupt loosening near the horizon end. This formulation allows the controller to tightly respect the probabilistic constraint at the beginning of the horizon, where accurate predictions are available, and gradually relax it for future steps, where uncertainty grows. Such an approach enables a balance between safety and feasibility, avoiding overly conservative behavior while ensuring probabilistic obstacle avoidance.

5.3 Real-time implementation

The solution of the optimization problem outlined in the previous section is achieved through the use of advanced frameworks for non-linear programming. Among these, *acados* [4] is recognized for its efficiency and real-time capabilities, making it particularly suitable for deployment on embedded platforms such as the dSPACE SCALEXIO AutoBox system [67], as demonstrated in [80]. This tool facilitates the rapid and reliable implementation of Non-linear Model Predictive Control (NMPC) algorithms in real-time environments.

5.3.1 Acados

The *acados* software package [4] is a collection of solvers specifically tailored for fast embedded applications, making it highly suitable for real-time control tasks. Its high-level interfaces enable the rapid development of optimization-based control algorithms through the integration of modular and easily interchangeable components. Leveraging a high-performance linear algebra backend, *acados* achieves significant computational efficiency while maintaining a high degree of flexibility. Key features of *acados* include:

- Implementation of efficient optimal control algorithms specifically targeting embedded devices, written in C code
- Use of the high-performance BLASFEO library [81] for linear algebra operations
- Interfaces available for MATLAB and Python
- Full compatibility with the CasADi [82] modeling language.

A significant challenge in the development of embedded optimal control software lies in achieving a balance between flexibility, memory usage, and computational efficiency. This trade-off is particularly critical in real-time applications, where strict timing constraints must be met despite the limited computational resources typical of embedded platforms. Earlier tools such as ACADO toolkit [83], addressed this challenge primarily through automatic code generation. In this approach, problem-specific C code is generated offline, including self-contained linear algebra routines

tailored to the given optimal control formulation. While this method can result in efficient implementations, it inherently limits flexibility. Once the code is generated, both the problem dimensions and algorithmic structures are fixed. Moreover, the maintenance of large volumes of generated code can be difficult, especially in iterative development cycles or when targeting diverse hardware platforms. In contrast, *acados* adopts a more flexible and modern approach by leveraging BLASFEO [81], a high-performance linear algebra library optimized for small to medium-scale matrices commonly encountered in embedded optimization. Rather than relying on code generation for linear algebra routines, *acados* uses BLASFEO at runtime to handle the most computationally intensive operations, such as matrix factorizations and linear system solves. This is a crucial design choice, as linear algebra computations typically dominate the overall computational complexity in optimal control solvers. Empirical results show that BLASFEO often outperforms even hand-crafted or code-generated routines in this domain, providing both speed and adaptability. As a result, *acados* achieves a favorable trade-off: it maintains computational efficiency while offering greater flexibility and ease of maintenance. This makes it particularly well suited for embedded applications that require rapid prototyping, frequent re-configuration, and reliable real-time performance. Another essential aspect in the design of embedded control software is the selection of the modeling language and the tools used for automatic differentiation, as these choices directly influence the system's flexibility, memory footprint, and runtime performance. Common symbolic computation tools such as Mathematica [84], SymPy [85], and the MATLAB Symbolic Toolbox [86] typically represent mathematical expressions using expression trees. While intuitive and powerful for symbolic manipulation, this representation often leads to increased code size, higher memory usage, and slower evaluation of higher-order derivatives, particularly in the context of complex dynamical models.

In contrast, CasADi adopts an expression graph-based representation, which enables more compact instruction sequences, reduced code size, and significantly faster execution. These features make CasADi particularly well suited for embedded optimization applications, where computational efficiency and resource constraints are critical. Moreover, CasADi is open-source and freely available, further contributing to its widespread adoption for modeling nonlinear functions and differential-algebraic equations in control systems. While the core of CasADi is implemented in self-contained C++ code, it is primarily accessed through its user-friendly interfaces

for Python and MATLAB. In the proposed implementation, MATLAB interface is preferred.

In addition to its support for CasADi, acados offers further flexibility by allowing users to integrate custom dynamic models written in C, either manually or via external code-generation pipelines. This capability enables the seamless incorporation of highly optimized or hardware-specific model implementations, thus broadening the range of applications and improving adaptability across different embedded platforms.

5.3.2 Implementation

The algorithm used to solve the non-linear programming (NLP) problem is the sequential quadratic programming (SQP), that solves the full quadratic problem (QP) at every iteration. It provides robust convergence but it is more computationally intensive. The alternative real-time iteration (SQP-RTI) performs only one QP solve per sampling instant; thus, it is faster but less accurate per step. For the proposed case, it was observed that the controller is able to solve the optimization problem within 0.05 s. Therefore, it was decided to retain this implementation without further investigating the SQP-RTI scheme. For a detailed comparison between linear MPC and NMPC, as well as an in-depth analysis of MPC based on the RTI scheme, the reader is referred to [87]. The resulting QP is solved using partial condensing high performance interior point method (HPIPM) solver, a high performance QP solver based on the BLASFEO [81] linear algebra library. This fast interior-point solver is designed specifically for embedded systems. The condensing feature reduces the QP size by exploiting the structure of the OCP, leading to faster QP solves while retaining sparsity. The Hessian in the SQP method are approximated with the Gauss-Newton method, which assumes the problem is least-squares-like. An efficient numerical integration of the system dynamics is carried out using an explicit fourth-order Runge-Kutta integrator. The implementation uses a prediction time horizon T of 2 s, discretized into 40 steps N_{steps} , corresponding to a time step of 0.05 s. This discretization is consistent with values commonly found in the literature [25]. The chosen time horizon $T = 2$ s represents a trade-off between the vehicle's dynamic response and the computational effort required to solve the optimization problem at each time step. A shorter horizon, such as $T = 1$ s, does not capture the full dynamics of the vehicle, leading to suboptimal control actions and degraded

prediction accuracy. On the other hand, increasing the horizon to $T = 3$ s or more increases computational complexity, with limited improvement in performance, particularly at moderate operating speeds.

Chapter 6

Experimental tests

This chapter discusses the challenges encountered during the experimental tests and outlines the design of the control architecture for the scaled vehicle. Figure 6.1 illustrates a simplified overview of the modular architecture developed in MATLAB/Simulink. The control pipeline includes a plant, which can represent either a simulated vehicle dynamic model or real hardware, such as the scaled vehicle. Sensor data is used to monitor the system's state, which is then processed by a state estimator to reconstruct a complete and accurate representation of the current system state. Based on this estimate, the controller computes the appropriate control inputs. The modular design of the framework allows for flexible integration of custom models and algorithms, making it straightforward to construct and test different control strategies.

In the following sections, each component shown in Figure 6.1 is described in detail, highlighting the specific models and algorithms used.

6.1 Sensor

Sensors play a crucial role in state estimation. The scaled vehicle platform integrates two primary sensing systems: a vision-based localization system and an onboard Inertial Measurement Unit (IMU). The vision system, composed of four VICON Vero v2.2 infrared cameras operating at 200 Hz, provides global position and orientation data with sub-millimeter precision under ideal conditions. However, latency (approximately 9 ms) and occasional marker occlusions introduce outliers in the

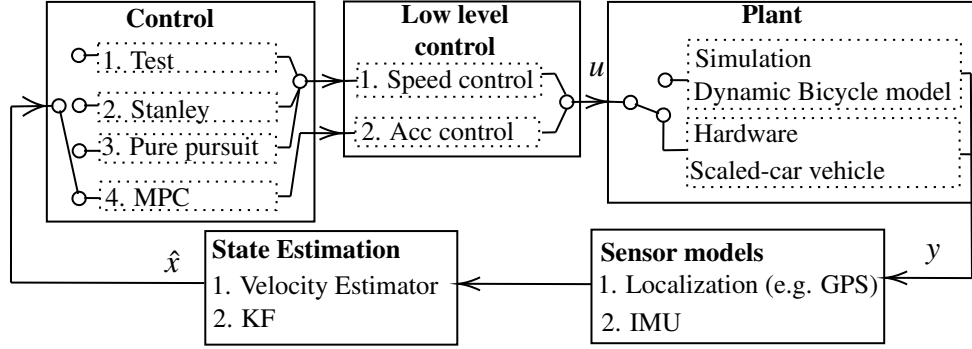


Fig. 6.1 Control pipeline for experimental tests. The proposed framework can be used for both simulations and hardware experiments.

position and orientation data streams. Moreover, when the system remains active for an extended period, the effectiveness of the vision system in tracking the vehicles tends to degrade, leading to the generation of outliers in the vehicle pose estimates. As a result, the vision system needs to be recalibrated. The onboard IMU (MPU-9250) provides acceleration and angular velocity data at 100 Hz. While offering high-frequency data suitable for capturing rapid vehicle dynamics, it is prone to drift and noise, especially in the longitudinal and lateral acceleration channels, due to vibrations and sensor bias. Notably, while IMU acceleration signals are not integrated into the control loop, the proposed control strategy relies solely on the pose data provided by the motion capture system. Therefore, to ensure accuracy and reduce the impact of noise and outliers on vehicle motion control, it is essential to filter the motion capture data using robust state estimation algorithms. The following sections introduce two representative sensors that are integrated into the control pipeline.

6.1.1 Localization systems

The localization system (e.g. GPS), such as the motion capture in the proposed scaled-car framework, provides only the pose of the system, that is, its position and orientation with respect to the world reference frame. This leads to the following discrete-time sensor data $G(k)$:

$$G(k) = \begin{bmatrix} X_G(k) & Y_G(k) & \Psi_G(k) \end{bmatrix}^T \quad (6.1)$$

where X_G , Y_G and Ψ_G are the world reference frame cartesian coordinates measured by the motion capture system.

6.1.2 Inertial Measurement Unit

The Inertial Measurement Unit (IMU) provides measurements of linear acceleration and angular velocities. This leads to the following discrete-time sensor data $I(k)$:

$$I(k) = \begin{bmatrix} a_{x,I}(k) & a_{y,I}(k) & a_{z,I}(k) & \dot{\phi}_I(k) & \dot{\theta}_I(k) & \dot{\psi}_I(k) \end{bmatrix}^T \quad (6.2)$$

where $a_{x,I}$ is the longitudinal acceleration, $a_{y,I}$ is the lateral acceleration, $a_{z,I}$ is the vertical acceleration, $\dot{\phi}_I$ is the rotation angle velocity around vehicle x axis, $\dot{\theta}_I$ is the rotation angle velocity around vehicle y axis and $\dot{\psi}_I$ is the rotation angle velocity around z axis, all measured by the IMU sensor on board the scaled-vehicle. Measured accelerations are typically subject to a high level of noise, while angular velocity measurements exhibit greater smoothness. Acceleration data are often integrated within an Extended Kalman Filter (EKF) for force state estimation in dynamic systems based on dynamic bicycle vehicle models [59]. In this specific case, the accelerations measured by the IMU are not integrated into the control pipeline. The yaw rate obtained from the IMU gyroscope sensor is compared with the time derivative of the yaw angle measured by the motion capture system, in order to verify proper calibration of motion capture system and IMU sensor installed onboard the vehicle and the plausibility of both measured signals.

6.2 State estimators

The majority of localization systems provide only the position and orientation of the vehicle in space, requiring linear and angular velocities to be estimated separately. This section introduces two approaches for estimating these quantities: a velocity estimator and a kinematic Kalman filter (KF). The first approach is a simplified method that estimates velocities by applying discrete differentiation to position and angle measurements. The second approach, by contrast, relies on a KF that estimates velocities using a kinematic model. As detailed in the following Sections, the KF is preferred over discrete differentiation due to its superior performance.

6.2.1 Velocity estimator

In the velocity estimator algorithm, the x- and y-velocities in world frame and the yaw rate are first estimated via finite differences, i.e.,

$$\begin{aligned} v_x^w(k) &= \frac{X_G(k) - X_G(k-1)}{T_s}, \quad v_y^w(k) = \frac{Y_G(k) - Y_G(k-1)}{T_s}, \\ \omega(k) &= \frac{\Psi_G(k) - \Psi_G(k-1)}{T_s} \end{aligned} \quad (6.3)$$

where T_s is the sampling time. These quantities can be then filtered using a discretized third-order Butterworth low-pass filter obtaining $\hat{v}_x^w(k)$, $\hat{v}_y^w(k)$ and $\hat{\omega}_x(k)$, i.e., the estimated x- and y-velocities in the world frame and the yaw rate. The filter is discretized at 200 Hz, as this matches the sampling rate of the input data acquired by the motion capture system. The cut-off frequency is set to 10 Hz, as the vehicle's dynamic response lies around 3–4 Hz. This choice ensures that higher-frequency noise is effectively attenuated without introducing significant phase delay within the bandwidth of interest. To retrieve the estimate of the linear velocities in the body frame, the following transformation is performed

$$\begin{aligned} \hat{v}_x(k) &= \hat{v}_x^w(k) \cos(\hat{\psi}(k)) + \hat{v}_y^w(k) \sin(\hat{\psi}(k)), \\ \hat{v}_y(k) &= -\hat{v}_x^w(k) \sin(\hat{\psi}(k)) + \hat{v}_y^w(k) \cos(\hat{\psi}(k)) \end{aligned} \quad (6.4)$$

6.2.2 Kinematic Kalman Filter

The implemented Kalman Filter (KF) is designed to estimate the state of the scaled-vehicle using a discrete-time linear kinematic model. The state vector $\hat{x}$ is defined as

$$\hat{x} = \begin{bmatrix} \hat{X} & \hat{Y} & \hat{\Psi} & \hat{v}_x^w & \hat{v}_y^w & \hat{\omega} \end{bmatrix}^\top \quad (6.5)$$

where $\hat{X}$, $\hat{Y}$ represent the position, $\hat{\Psi}$ is the orientation (yaw), $\hat{v}_x^w$ and $\hat{v}_y^w$ are the linear velocities in the respective directions, and $\hat{\omega}$ is the angular velocity (yaw rate). The filter uses measurements of position and yaw coming from the motion capture system, expressed as:

$$z = \begin{bmatrix} X_G & Y_G & \Psi_G \end{bmatrix}^\top \quad (6.6)$$

The state transition is governed by a constant-velocity model, with the state transition matrix F defined as:

$$F = \begin{bmatrix} 1 & 0 & 0 & T_s & 0 & 0 \\ 0 & 1 & 0 & 0 & T_s & 0 \\ 0 & 0 & 1 & 0 & 0 & T_s \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.7)$$

where $T_s = 5 \times 10^{-3}$ is the time step. The process noise covariance matrix Q models the uncertainty in the system dynamics, assuming process noise standard deviation σ_q . It is structured using standard second-order integration assumptions:

$$Q = \sigma_q^2 \begin{bmatrix} \frac{T_s^4}{4} & 0 & 0 & \frac{T_s^3}{2} & 0 & 0 \\ 0 & \frac{T_s^4}{4} & 0 & 0 & \frac{T_s^3}{2} & 0 \\ 0 & 0 & \frac{T_s^4}{4} & 0 & 0 & \frac{T_s^3}{2} \\ \frac{T_s^3}{2} & 0 & 0 & T_s^2 & 0 & 0 \\ 0 & \frac{T_s^3}{2} & 0 & 0 & T_s^2 & 0 \\ 0 & 0 & \frac{T_s^3}{2} & 0 & 0 & T_s^2 \end{bmatrix} \quad (6.8)$$

The observation matrix H maps the state to measurement space:

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \quad (6.9)$$

The measurement noise covariance matrix R is

$$R = \text{diag}(\sigma_X^2, \sigma_Y^2, \sigma_\Psi^2) \quad (6.10)$$

where σ_X is the measurement noise standard deviation for position x , σ_Y is the measurement noise standard deviation for position y and σ_Ψ is the measurement noise standard deviation for orientation angle Ψ . Given the current estimated stated vector $\hat{x}_{k-1|k-1}$ at time step $k-1$, the KF performs a predict step k using the state transition matrix F :

$$\hat{x}_{k|k-1} = F\hat{x}_{k-1|k-1} \quad (6.11)$$

The KF predicts the error covariance matrix $P_{k|k-1}$, which captures the uncertainty in the state estimate at time step k :

$$P_{k|k-1} = F P_{k-1|k-1} F^\top + Q \quad (6.12)$$

where $P_{k-1|k-1}$ is the previous estimate covariance matrix. Subsequently, the KF performs the update step using the incoming measurements z_k at time step k . The innovation $\hat{y}$ or residual is:

$$\hat{y}_k = z_k - H \hat{x}_{k|k-1} \quad (6.13)$$

with innovation covariance S_k :

$$S_k = H P_{k|k-1} H^\top + R \quad (6.14)$$

Finally, the Kalman gain K_k is computed as:

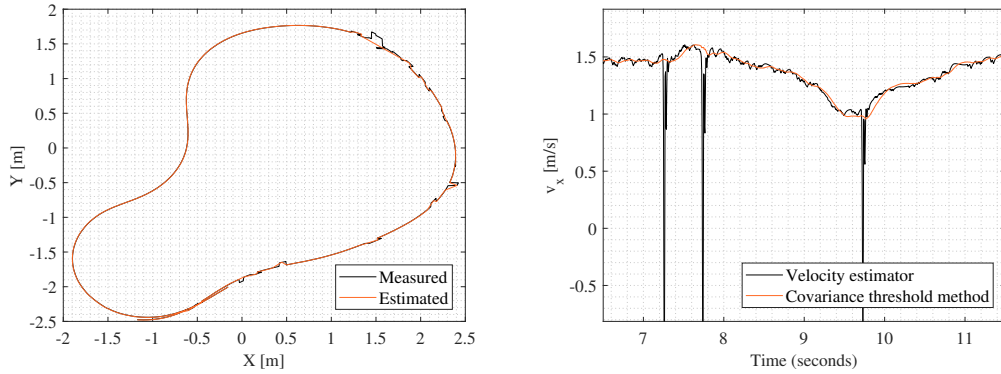
$$K_k = P_{k|k-1} H^\top S_k^{-1} \quad (6.15)$$

Then, the state estimate and its covariance are updated:

$$\begin{aligned} \hat{x}_{k|k} &= \hat{x}_{k|k-1} + K_k y_k \\ P_{k|k} &= (I - K_k H) P_{k|k-1} \end{aligned} \quad (6.16)$$

6.2.3 Outlier rejection

In the context of Kalman Filters, outlier rejection is a mechanism to detect and ignore measurements that significantly deviate from the expected value. These could be caused by sensor glitches, motion capture system (i.e., GPS) spikes or communication noise. If used in the KF update step, outliers can seriously degrade state estimates. After the prediction step, the KF compares the predicted measurement with the actual measurement through Eq. 6.13 and computes the innovation covariance matrix S with Eq. 6.14. If outlier is detected, instead of updating the estimate with measurement z_k , state and error covariance are directly projected ahead using Eq. 6.11 and 6.12. Two outlier rejection methods are developed, the covariance threshold and the Chi-squared method, respectively. These approaches are well explained in [59].



(a) Kalman filter using covariance threshold as outlier rejection method for path estimation (b) Kalman filter using covariance threshold as outlier rejection method for v_x estimation

Fig. 6.2 KF with covariance threshold outlier rejection method

Covariance threshold

A measurement z_k is rejected if the norm of the innovation is greater than some multiple c_{th} of the innovation covariance matrix S :

$$\|\hat{y}_k\| \geq c_{th} \cdot \sqrt{\sum_{i=1}^m s_{ii}} \quad (6.17)$$

where s_{ii} are the diagonal elements of the innovation covariance matrix S and m is the number of measurements. Figure 6.2a shows the path of the scaled vehicle measured by the motion capture system during an experimental trial (black line). As can be observed, the system introduces measurement errors while tracking the position of the scaled vehicle. These measurements are not suitable for use as control inputs for the vehicle motion controller. By adopting this approach, an estimate of the path (orange line) that avoids measurement spikes is obtained. As shown in Fig. 6.2b, the measurement of longitudinal velocity is also affected by noise (black line). Using a covariance threshold outlier rejection method, estimation of longitudinal velocity without measurement noise (orange line) is obtained.

Chi-squared

This method uses the fact that, under Gaussian noise assumptions, the Mahalanobis distance squared between the measurement and its prediction follows a Chi-squared

distribution. if the Mahalanobis distance is zero, the measurement corresponds to the mean of the distribution. This distance is computed as:

$$d = \sqrt{\hat{\mathbf{y}}^\top \mathbf{S}^{-1} \hat{\mathbf{y}}} \quad (6.18)$$

This measures the normalized distance of the measurement from the predicted mean, accounting for the uncertainty in each direction (via $\mathbf{S}$). Since $d^2 \sim \chi_m^2$, where χ_m^2 is the Chi-squared distribution with m degrees of freedom, it is possible to set a probabilistic threshold as:

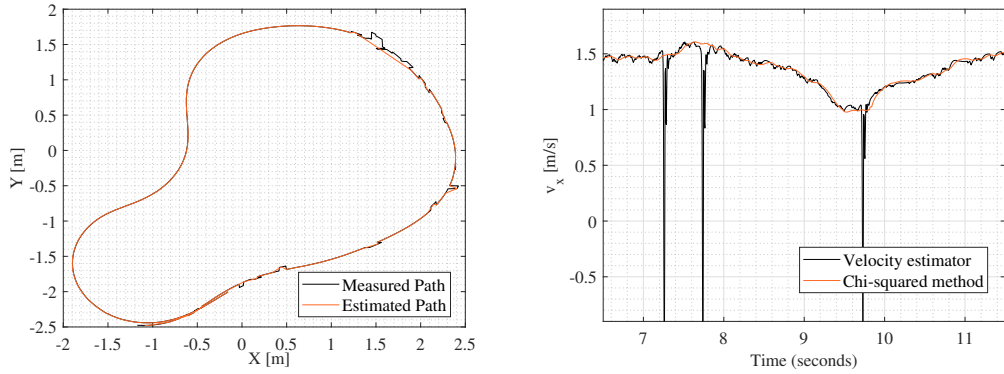
$$d^2 \geq t \quad (6.19)$$

where t is computed as:

$$t = 2\Gamma^{-1}\left(\frac{m}{2}, p \cdot \gamma\left(\frac{m}{2}\right)\right) \quad (6.20)$$

with $\Gamma()$ as gamma function, $\gamma()$ as lower incomplete gamma function and p as desired confidence level. This method inherently accounts for correlations and directional uncertainties in the measurement space, and allows for principled tuning based on a user-defined confidence level (e.g., 0.95, 0.99). However, it requires matrix inversion or decomposition, which may be computationally intensive and numerically sensitive, especially in real-time environments. Figure 6.3a shows the path of the scaled vehicle measured by the motion capture system during an experimental trial (black line). As can be observed, the system introduces measurement errors while tracking the position of the scaled vehicle. By adopting the Chi-squared approach, an estimate of the path (orange line) that avoids measurement spikes is obtained. As shown in Fig. 6.3b, the measurement of longitudinal velocity is also affected by noise (black line). Thus, an estimation of longitudinal velocity without measurement noise (orange line) is obtained.

In contrast, the covariance threshold method simplifies the outlier detection process by comparing the Euclidean norm of the innovation vector to a scaled square root of the trace of the innovation covariance. This heuristic test, while computationally efficient and easy to implement, does not fully capture the structure of uncertainty and lacks a probabilistic interpretation. It assumes that deviations in all directions are equally significant, which may not hold in cases where the measurement noise is anisotropic or correlated. Despite its limitations, the covariance threshold approach remains a practical choice for low-dimensional systems or embedded applications where computational simplicity is paramount. Table 6.1 presents a comparison of the



(a) Kalman filter using Chi-squared as outlier rejection method for path estimation (b) Kalman filter using Chi-squared as outlier rejection method for v_x estimation

Fig. 6.3 KF with Chi-squared rejection method

Table 6.1 Performance indicators for proposed outlier rejection methods

	Covariance Threshold		Chi-squared	
	RMSE	MAE	RMSE	MAE
$X[m]$	0.014	0.004	0.053	0.014
$Y[m]$	0.015	0.005	0.036	0.011
$v_x[m/s]$	0.110	0.016	0.113	0.024

proposed outlier rejection methods in terms of Root Mean Square Error (RMSE) and Mean Absolute Error (MAE), evaluated over position and velocity measurements. As shown, the covariance threshold method yields lower RMSE values across all three variables when compared to the Chi-squared approach, indicating a smaller average deviation between the estimated and true states. Similarly, the MAE values are also lower for the covariance threshold, suggesting that this method provides more consistent and accurate estimates in this specific application. Therefore, the covariance threshold is preferable as an outlier rejection strategy in the given context.

6.3 Control

The retained scaled-vehicles can be operated using various control strategies. Specifically, several types of controllers have been designed. A state machine is implemented within the Simulink environment to enable experimental testing by switching between the developed control schemes as needed. The structure of the state machine

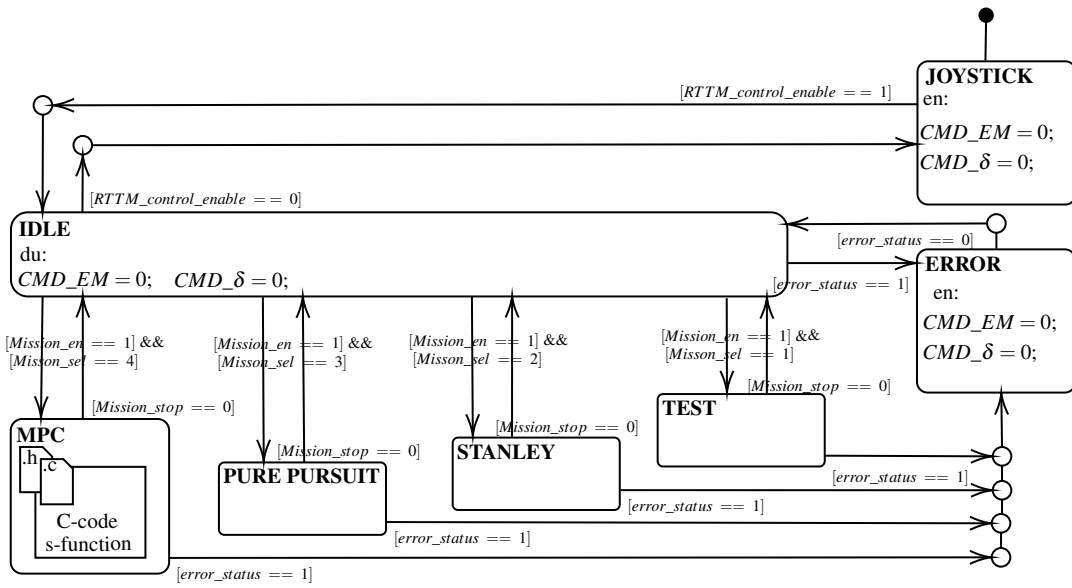


Fig. 6.4 Vehicle control state machine

is illustrated in Fig. 6.4. It transitions between states depending on mission commands, control enable signals, and error statuses. The outputs of the state machine are CMD_{EM} and CMD_{δ} , which are the commands for the electric motor and steering servo, respectively.

When powered on, the vehicle starts in JOYSTICK mode, meaning it can be remotely operated using a generic console controller. For safety reasons, this feature must first be enabled by the ground station through a physical switch, which activates Bluetooth communication and pairing for remote control.

To switch to control via the Real-Time Target Machine (RTTM), the system must receive a cyclic signal, $RTTM_control_enable = 1$. Once this signal is active, the vehicle transitions into the default IDLE mode, in which the system remains inactive, i.e., the output signals CMD_{EM} and CMD_{δ} are both set to zero.

To initiate a specific mission, namely TEST, STANLEY, PURE PURSUIT, or MPC, the variable $Mission_sel$ must be set accordingly, followed by a trigger pulse ($Mission_en = 1$) to enter the selected mission. Conversely, to exit the mission and return to IDLE, a low pulse ($Mission_stop = 0$) is sent.

An ERROR state is also implemented: when $error_status = 1$, the system enters the error state. It will automatically return to IDLE once the condition clears (i.e., when $error_status = 0$).

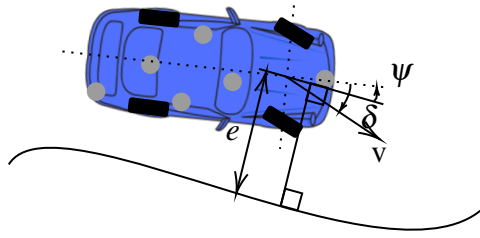


Fig. 6.5 Stanley controller kinematic model

6.3.1 TEST control

In this mode, the duty cycles for controlling both the electric motor and the steering servo via PWM are manually imposed. This configuration is essential for conducting quasi-static tests, such as fixed steering or ramp steer maneuvers, at constant velocity conditions, as described in [34, 58]. This operating mode is particularly useful for vehicle model identification and actuator mapping.

6.3.2 STANLEY control

The Stanley controller [88] provides closed loop tracking of the desired vehicle path, as determined by the path planner. It helped Stanford to win the DARPA [89] challenge in 2006. The kinematic motion of a vehicle, with speed v , can be described with the crosstrack error, e , of the guiding wheels, and the angle of those wheels with respect to the nearest segment of trajectory to be tracked, $(\psi - \delta)$, as in Figure 6.5. Here, ψ is the yaw angle of the vehicle with respect to the closest trajectory segment, and δ is the angle of the front wheels with respect to the vehicle. This reference frame allows the actuator to act directly on the error metric, yielding collocated sensing and control. For forward driving, the guiding wheels are the front wheels, and the derivative of the crosstrack error is

$$\dot{e} = v \sin(\psi - \delta) \quad (6.21)$$

where the steering is mechanically limited to $|\delta| < \delta_{max}$. The derivative of the yaw angle is:

$$\dot{\psi} = -\frac{v \sin(\delta)}{l_f + l_r} \quad (6.22)$$

where l_f and l_r are the distance from the CoG to the front and rear wheels, respectively. By inspecting Equations 6.21 and 6.22, a controller is selected such that the resulting differential equation has a globally asymptotically stable equilibrium at zero crosstrack error. This results in the steering control law

$$\delta = \begin{cases} \psi + \arctan \frac{ke}{v+v_b} & \text{if } \left| \psi + \arctan \frac{ke}{v+v_b} \right| < \delta_{max} \\ \delta_{max} & \text{if } \psi + \arctan \frac{ke}{v+v_b} \geq \delta_{max} \\ -\delta_{max} & \text{if } \psi + \arctan \frac{ke}{v+v_b} \leq -\delta_{max} \end{cases} \quad (6.23)$$

where v_b is constant to avoid singularity when velocity v is zero. In this context, the control strategy is applied to the lateral dynamics of the vehicle. The longitudinal dynamics, on the other hand, is governed by a PI controller regulating the vehicle's speed. This approach assumes a decoupled control framework, in which the longitudinal and lateral dynamics are treated independently.

6.3.3 PURE PURSUIT control

The Pure Pursuit controller is a path tracking algorithm that determines the appropriate curvature required to guide a vehicle from its current position to a target point along a predefined path. This target lies at a fixed lookahead distance l ahead of the vehicle's current location. To compute the steering command, the controller constructs a circular arc connecting the vehicle's current position to the goal point. The length of the chord connecting these two points corresponds to the lookahead distance, serving as a key geometric constraint to uniquely define the arc. Conceptually, the lookahead distance can be likened to the visual point ahead that a human driver would intuitively focus on to follow the road.

Figure 6.6 illustrates the Pure Pursuit control geometry. The vehicle is depicted along with the axes of its local coordinate frame, in which the x-axis is aligned with the rear axle and originates at the rear differential. As demonstrated by Shin [90], positioning the coordinate system in this manner ensures a geometric decoupling between propulsion and steering actions.

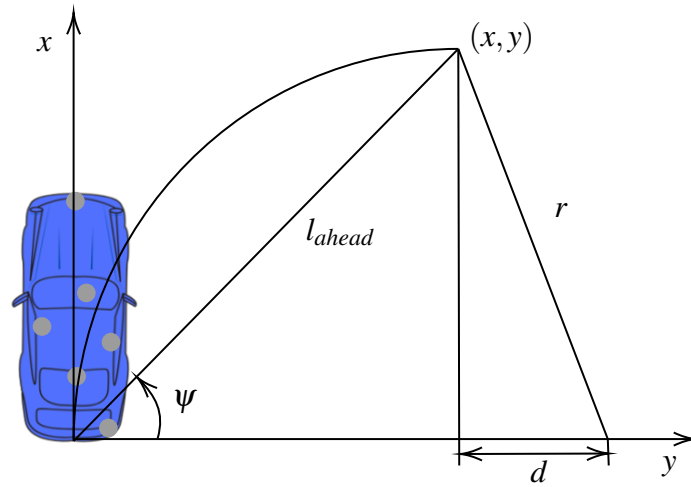


Fig. 6.6 Pure Pursuit controller model

A target point (x, y) , located at a fixed lookahead distance l from the origin, is also shown in the Fig 6.6. This point lies on the reference path and serves as the goal toward which the vehicle steers. The control objective is to determine the curvature κ of the circular arc that connects the vehicle's current position (the origin) to the goal point (x, y) , such that the arc has a chord length equal to the lookahead distance l_{ahead} . The geometric relationship between the arc and the goal point is captured by the following system of equations:

$$\begin{aligned} x^2 + y^2 &= l_{ahead}^2 \\ y + d &= r \end{aligned} \quad (6.24)$$

The first equation defines a circle of radius l_{ahead} centered at the origin, representing the locus of feasible goal points. The second equation expresses the relation between the arc radius r and the horizontal displacement x of the goal point, with d denoting the distance from the center of the arc to the origin. This formulation implies that the radius and horizontal offset differ by a constant d , allowing one to uniquely determine the arc geometry. By algebraically manipulating Eq. 6.24, the curvature κ of the arc, by definition the reciprocal of the radius r , can be derived as a function of the lookahead distance:

$$\kappa = \frac{2y}{l_{ahead}^2} \quad (6.25)$$

This curvature value is used to generate the steering command required to align the vehicle with the desired trajectory. Curvature κ and steering angle δ are related by

the geometry:

$$\delta = \tan^{-1}(l\kappa) \quad (6.26)$$

where l is the wheelbase. Combining Eq. 6.25 and 6.26, knowing the geometrical relationship $y = \sin(\psi)l_{ahead}$, the steering angle δ is

$$\delta = \tan^{-1}\left(\frac{2l \sin(\psi)}{l_{ahead}}\right) \quad (6.27)$$

In this control framework, the lateral and longitudinal dynamics of the vehicle are decoupled. The lateral motion is regulated by the curvature computed via the Pure Pursuit algorithm, while the longitudinal behavior is independently governed by a PI controller that adjusts the vehicle's speed.

6.3.4 MPC control

The MPC controller adopts a combined approach, simultaneously computing control actions for both the longitudinal dynamics $a_{x,target}$ and the lateral dynamics δ . The prediction model is based on the curvilinear vehicle formulation described in Eq. 5.4. The optimization problem formulation is presented in Section 5.1.

6.4 Low-level control

The low-level controller determines the target Duty cycle D to be imposed on the electric motor to ensure the desired performance is attained, both in terms of reference velocity or reference acceleration. When either the Stanley or Pure Pursuit controller is active, the vehicle is controlled through a reference velocity, while under MPC control, it is governed by a reference acceleration. The MPC controller computes the reference inputs for both the electric motor and the steering servo. In contrast, the Stanley and Pure Pursuit controllers are path-tracking controllers that solely generate the steering angle reference δ for the steering servo, without addressing longitudinal dynamics. In the context of trajectory planning and control, it is common practice to compute a velocity profile that ensures the lateral acceleration remains below a

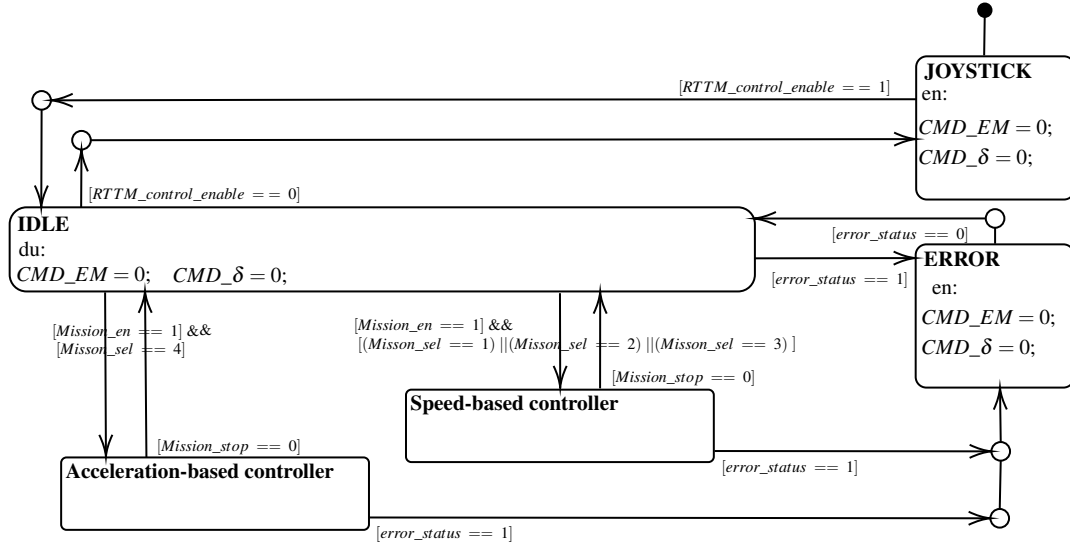


Fig. 6.7 Low level control state machine

specified maximum value $a_{y,max}$ [91]. This constraint is typically enforced by:

$$v_{\max} = \sqrt{\frac{a_{y,\max}}{|\kappa|}} \quad (6.28)$$

Thus, lateral and longitudinal dynamics are interconnected through the path curvature κ . However, for the Stanley and Pure Pursuit controllers, a feedback (FB) control strategy is developed based on a Proportional-Integral (PI) velocity controller. Since the primary focus of the study is on the MPC controller, alternative control techniques for these lateral controllers are not further explored. For the MPC controller, a more advanced control scheme is implemented, combining a FB component, again based on a PI velocity controller, with a feedforward (FF) component derived by inverting the longitudinal dynamics model, as detailed in Section 4.2. This approach enables the design of a high-level control system that is generalizable across different vehicle types, from full-scale to scaled models, since the computed control input corresponds to a target longitudinal acceleration $a_{x,target}$. Conversely, the low-level control remains dependent on the specific hardware used: in this case, the target longitudinal acceleration is converted into a Duty cycle D to drive the electric motor.

Figure 6.7 illustrates the state machine governing the low-level control system. Similar to the vehicle control state machine, the vehicle initially operates in JOY-STICK mode. Transition to control via the RTTM requires the system to receive a continuous signal, namely $RTTM_control_enable = 1$. Upon activation of this

signal, the vehicle enters the default IDLE state, during which no commands are issued; specifically, both CMD_{EM} and CMD_{δ} are set to zero.

Depending on the selected mission, the vehicle may transition into one of two distinct modes: Speed-based Controller or Acceleration-based Controller. The Speed-based Controller mode is activated when the mission selected is either TEST, STANLEY, or PURE PURSUIT ($Mission_{sel} = 1$, $Mission_{sel} = 2$, $Mission_{sel} = 3$), whereas the Acceleration-based Controller mode is exclusively associated with the MPC mission ($Mission_{sel} = 4$). In Speed-based Controller mode, a FB controller based on a PI control of the vehicle's speed is utilized. In contrast, the Acceleration-based Controller mode employs a combined FB and FF control strategy, incorporating both a PI controller on velocity and a model-based feedforward component derived from the longitudinal dynamics.

Transition into either of these two control modes requires a trigger signal ($Mission_{en} = 1$). Conversely, to revert to the IDLE state, a low signal ($Mission_{stop} = 0$) must be issued.

Additionally, an ERROR state is implemented: whenever $error_status = 1$, the system automatically transitions to the Error state. It will subsequently return to IDLE once the error condition is cleared (i.e., when $error_status = 0$).

6.4.1 Speed-based control

The speed controller implemented is a Proportional-Integral (PI) controller, designed to regulate the vehicle's longitudinal speed. The proportional gain, K_p , is set to 0.3, reflecting a moderate proportional action tailored to the system's dynamic response. The integral time constant, τ_i , is set to 8 seconds, implying that the integrator produces a unit of output for a unit of input over a period of 8 seconds. The discrete-time integral gain, K_i , is set $5 \times K_p / \tau_i$, in order to increase the aggressiveness and convergence of the integrator. Furthermore, the controller output is constrained within saturation limits, with an upper bound of 0.3 and a lower bound of -0.3 , to prevent excessive control actions that could destabilize the system. These saturation bounds are defined to mitigate the risk of tire saturation (i.e., spinning) at the rear driving wheels, particularly in light of the reduced surface adhesion characteristic of the test environment, while still preserving adequate vehicle responsiveness and agility. In summary, the tuning of the controller parameters reflects a compromise

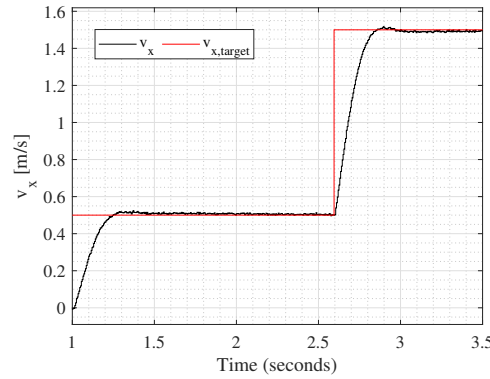


Fig. 6.8 FB velocity control

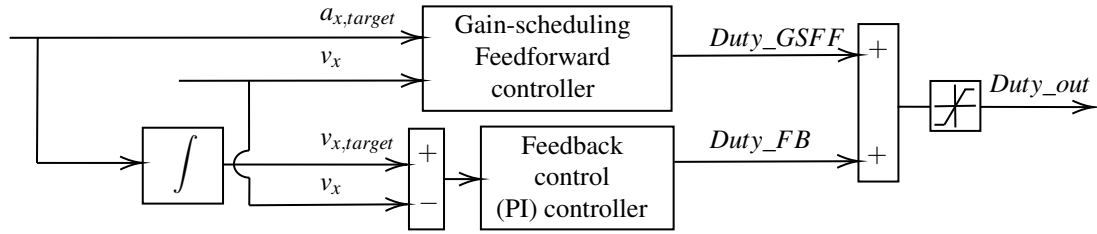


Fig. 6.9 Acceleration-based control scheme

between dynamic performance and closed-loop stability, carefully tailored to the vehicle's longitudinal dynamics and the physical limitations of the actuation system. Figure 6.8 shows the behaviour of the proposed feedback velocity control, where the red line represents the reference speed $v_{x,target}$ and v_x is the actual velocity of the scaled vehicle. As can be observed, the vehicle exhibits a first-order response characterized by a reduced overshoot and a rapid settling towards the reference value.

6.4.2 Acceleration-based control

Figure 6.9 presents the architecture of the acceleration-based control strategy for longitudinal vehicle speed regulation. The system receives as inputs the desired longitudinal acceleration command, $a_{x,target}$ generated by the MPC controller, along with the current longitudinal velocity, v_x . A gain-scheduling feedforward controller calculates an open-loop control input $Duty_{GSFF}$, by inverting the longitudinal vehicle dynamics model to directly map the target acceleration into a control input. Concurrently, a PI feedback controller generates a corrective Duty cycle command, $Duty_{FC}$, based on the velocity tracking error, defined as the difference between the

target velocity (v_{target}) and the measured velocity (v_x). The PI controller is tuned via the gains K_p and K_i , and incorporates saturation limits to constrain the control effort, in alignment with the design principles adopted for the previous Speed-based controller. Additionally, the architecture includes reset mechanisms to reinitialize both the integrator state and the target reference value when required. The final control signal ($Duty_out$) is obtained by summing the GSFF and FB contributions. This integrated control approach leverages the predictive capabilities of feedforward compensation to achieve fast dynamic response, while ensuring steady-state accuracy and robustness through closed-loop feedback correction.

Gain-scheduling feedforward control

The gain-scheduling feedforward controller aims to compute the actuator command Duty Cycle D required to achieve a desired longitudinal acceleration $a_{x,target}$. The control action is based on inverting a simplified longitudinal force model of the vehicle, which accounts for both driving and braking dynamics. The net longitudinal force F_x generated by the system as a function of the duty cycle $D \in [-1, 1]$ and the current vehicle velocity v_x is modeled as:

$$F_x(D, v_x) = K_{m1} \cdot \max(0, D) + K_{brake} \cdot v_x \cdot \min(0, D) - K_{m2} \cdot v_x \quad (6.29)$$

Vehicle modeling is discussed in Section 4.2. Given a target acceleration $a_{x,target}$, the corresponding target force is

$$F_{x,target} = m \cdot a_{x,target} \quad (6.30)$$

where m is the mass of the scaled-vehicle. The controller discretizes the duty cycle range into a finite set D_i and evaluates $F_x(D_i, v_x)$ over all samples. The optimal duty cycle D^* is selected by minimizing the absolute difference between the modeled and target forces:

$$D^* = \arg \min_{D_i} |F_x(D_i, v_x) - F_{x,target}| \quad (6.31)$$

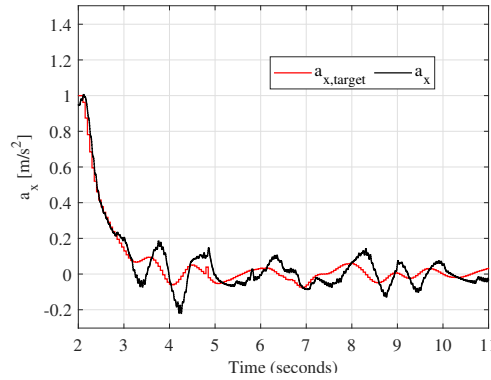


Fig. 6.10 Acceleration control with FB and GSFF control

Finally, the computed FF Duty cycle is saturated to ensure it remains within the allowable physical limits:

$$Duty_GSFF = \min(\max(D^*, -1), 1) \quad (6.32)$$

Figure 6.10 illustrates the response of the scaled vehicle when controlled using a combined feedback (FB) and a gain-scheduling feedforward (GSFF) approach. In this configuration, the acceleration reference, shown as the red line ($a_{x,target}$), is continuously updated by the MPC controller since the controller is working in Racing-oriented control formulation mode, while the duty cycle D is regulated to track the reference signal. The actual vehicle longitudinal acceleration, represented by the black line (a_x), closely follows the desired trajectory. As observed, the control strategy effectively ensures accurate tracking of the reference longitudinal acceleration.

6.5 Plant

The designed architecture enables the testing of autonomous driving control algorithms through a vehicle simulator before their integration into the real hardware control loop, namely the scaled-vehicles. This approach allows for extensive debugging and functional safety validation across the entire simulation loop, thereby avoiding the deployment of code affected by errors. When the simulator is active, the steering angle command δ computed by the Control layer is directly sent as

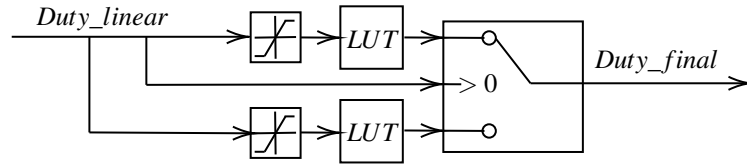


Fig. 6.11 Implementation of the feedback linearization logic.

input to the vehicle model. Similarly, the duty cycle command D , which manages the electric motor, is calculated by either the speed-based or acceleration-based controller according to the previously described logic, and then directly applied to the vehicle model. The simulator environment is based on the vehicle model introduced in Section 4.3 and described by the Eq. 4.22, specifically to introduce a model mismatch between the prediction model used by the MPC and the actual plant. This mismatch is intentional to evaluate the robustness and reliability of the control system. When the control loop is executed with real hardware, the commands computed by the two control layers are processed through a feedback linearization subsystem before being applied to the vehicle.

Feedback linearization

The feedback linearization block aims to compensate for the nonlinear relationship between the input command computed by the controllers and the actual actuator input required to produce the intended physical effect on the system. The input duty ($Duty_linear$), computed under the assumption of linear actuator behaviour, is split into two branches, one for positive values (acceleration/throttle - $+\delta$) and one for negative values (deceleration/braking - $-\delta$). Each branch passes through a nonlinear function (i.e., Lookup table (LUT)), which compensates for the actuator's real behaviour. The nonlinear maps are derived through static calibration procedures of the actuators. In particular, the deadband of each actuator is experimentally characterized by identifying the minimum Duty cycle required to induce motion in the scaled vehicle, and the minimum input necessary for the steering system to apply a measurable steering angle δ . The final duty ($Duty_final$) is the feedback-linearized command actually sent to the actuators. This logic holds for both the command sent to the electric motor and the steering servo. Figure 6.11 shows the feedback-linearization scheme. With this proposed approach, it is possible to design a piecewise feedback linearization where the actuator command is corrected separately

for throttle and brake actions, as well as for positive and negative steering angles. This approach ensures that, even though the control layer assumes a linear input-to-output relation, the real actuator's nonlinearities are compensated internally before actuation. This correction mechanism ensures that the real vehicle response remains consistent with the control assumptions, thus preserving the accuracy, robustness, and functional safety of the entire system, especially in the presence of model-based predictive controllers such as MPC.

Chapter 7

Experimental results and Discussion

This Chapter presents the experimental results related to the proposed collision avoidance (CA) controllers introduced in Section 5.2. The experiments were conducted at the Museo Nazionale dell'Automobile di Torino (MAUTO), which, in collaboration with the Centro Nazionale per la Mobilità Sostenibile (MOST), provided an exhibition space dedicated to testing autonomous driving technologies developed within this infrastructure. The so-called "Living lab" serves multiple purposes: it allows the general public to engage with autonomous driving technologies, supports research activities, and provides an experimental and practical environment where students can apply the theoretical knowledge acquired during theory lessons. The infrastructure is particularly suited for student use, as it is based on MATLAB/Simulink tools.

Figure 7.1 shows the layout of the test circuit, which covers an area of $5 \times 6 \text{ m}^2$. The track consists of two squares connected with minimum corner radii of 0.7 m to ensure that the vehicle can physically complete the turns without exceeding its steering limits, thus avoiding saturation of the steering command δ calculated by the MPC controller. The two squares intersect at two points, creating two crossings. The track features two-lane roads with possibly one or two-way traffic. Each lane has a width of 0.35 m, resulting in a total road width of 0.7 m. These track dimensions are coherent with scaled-vehicles.

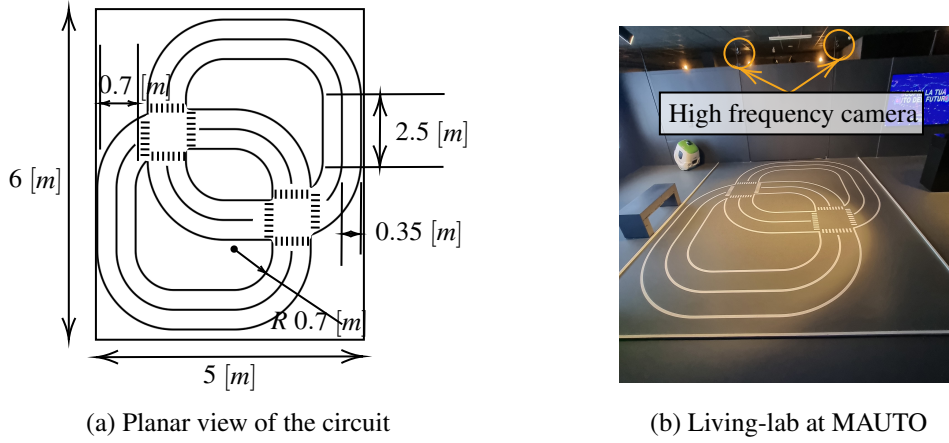


Fig. 7.1 Experimental layout of the circuit

7.1 Reference lateral deviation generator logic

The presented experimental results provide a comprehensive validation of the proposed Collision Avoidance (CA) strategy based on lateral deviation reference tracking, as introduced in Section 5.2.1. Figure 7.2 shows the temporal evolution of the states during the experimental test. The first subplot illustrates the steering angle δ , demonstrating smooth yet dynamic variations, which reflect the vehicle's need to adjust its trajectory to avoid obstacles while maintaining stability. The second plot compares the actual longitudinal velocity v_x with the target velocity $v_{x,target}$. The two curves align closely, indicating effective tracking performance and confirming that the low level control module maintains the desired speed profile even during complex maneuvers. The third subplot presents the lateral deviation n and its reference n_{ref} . The deviation from the nominal path, particularly around 4 to 7 seconds, corresponds to obstacle avoidance behavior, after which the vehicle gradually returns to the original path. The fourth plot displays the yaw angle error ψ_e , with moderate oscillations that remain within acceptable limits, further confirming good path-tracking performance. The final plot depicts the longitudinal position s , which increases linearly over time as expected.

Figures 7.3 complements the time-domain analysis. Figure 7.3a presents the evolution of the commanded control actions along the CA maneuver. The top plot shows the longitudinal acceleration command $a_{x,target}$. Initially, the command shows a deceleration phase, needed to adapt to the vehicle speed reference $v_{x,target}$ of about 1.4 m/s. After approximately $t = 4$ seconds the longitudinal acceleration stabilizes

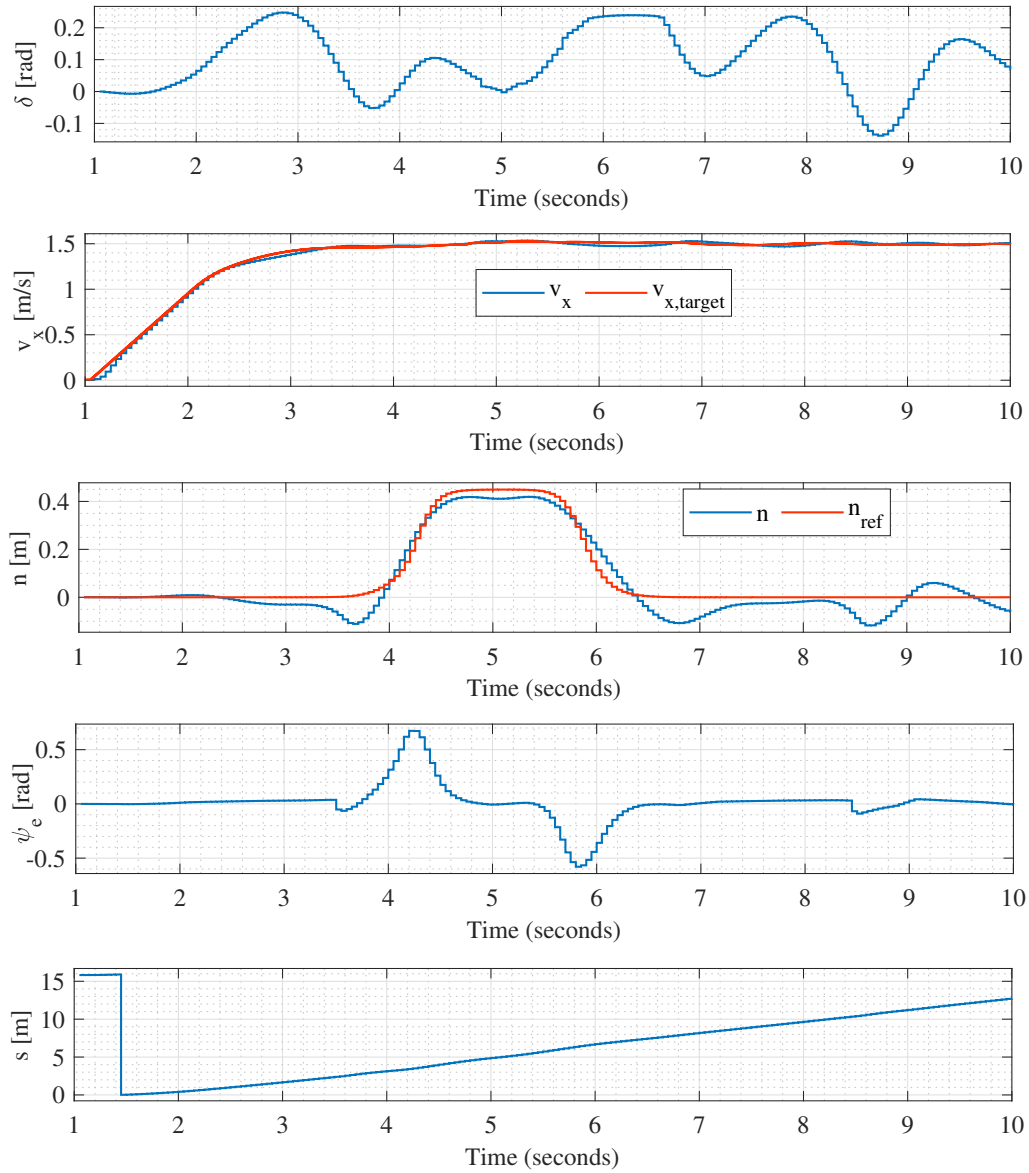


Fig. 7.2 States evolution along the maneuver. Collision avoidance with static obstacle using Reference lateral deviation generator logic

around zero, indicating that the vehicle maintains a steady velocity while navigating through the circuit, with only minor adjustments to compensate for the path curvature and obstacle avoidance. The bottom plot depicts the steering rate command $\Delta\delta$ over time. Here, periodic variations can be observed, corresponding to the steering actions necessary to follow the circuit and execute lateral deviation maneuvers around the static obstacle. Notable peaks and saturations along the entire maneuver highlight the vehicle's increased steering efforts when negotiating tighter curves or intersections.

It is important to emphasize that the urban environment is not inherently suited to accommodate this category of scaled racing vehicles, which are characterized by a significantly limited turning radius. As a result, maneuvering within narrow lanes and sharp corners presents considerable challenges for their operation, due to steering servo command saturations. Overall, the control commands remain within feasible operational boundaries, demonstrating that the CA logic effectively balances collision avoidance objectives with static obstacle constraints.

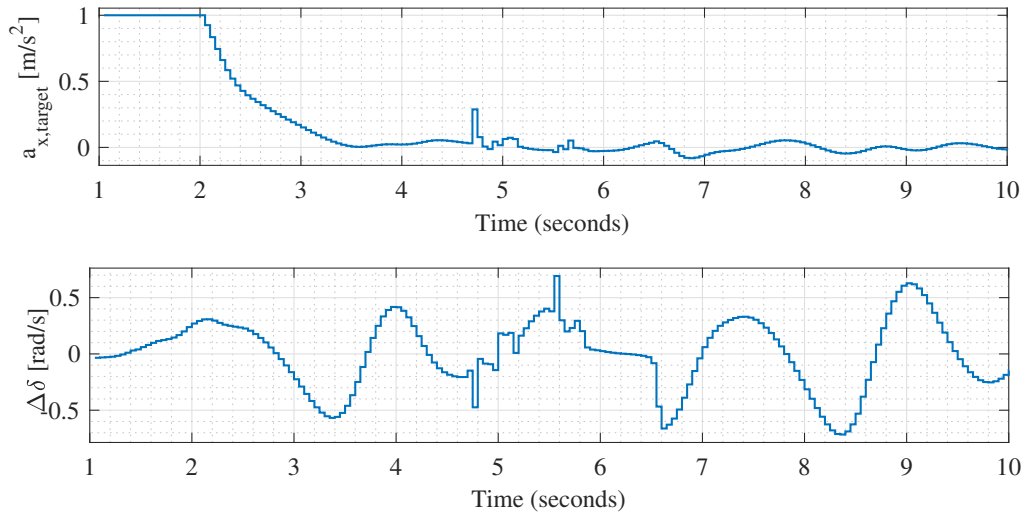
Figure 7.3b shows the acceleration behavior in the a_x and a_y plane. The scaled vehicle acceleration values remain well within the defined acceleration boundaries defined by the Kamm's circle through the nonlinear constraint, confirming that the vehicle operates safely within its dynamic limits during the test. This result is crucial for ensuring stability and avoiding loss of control during aggressive maneuvers.

Figure 7.3c provides a top-down view of the vehicle trajectory (blue rectangle) within the defined track boundaries. The reference path n_{ref} is closely followed, with slight deviations occurring near obstacle regions, as expected (i.e., $n_{ref,obst}$ is active). The track boundaries are respected throughout the maneuver, indicating that the vehicle successfully avoids collisions while staying within the drivable area. The red rectangles denote the positions of the obstacles. Overall, the experimental results demonstrate that the proposed CA approach is effective in real-scaled-world conditions, ensuring accurate path tracking, safe obstacle avoidance, and operation within the vehicle's dynamic constraints.

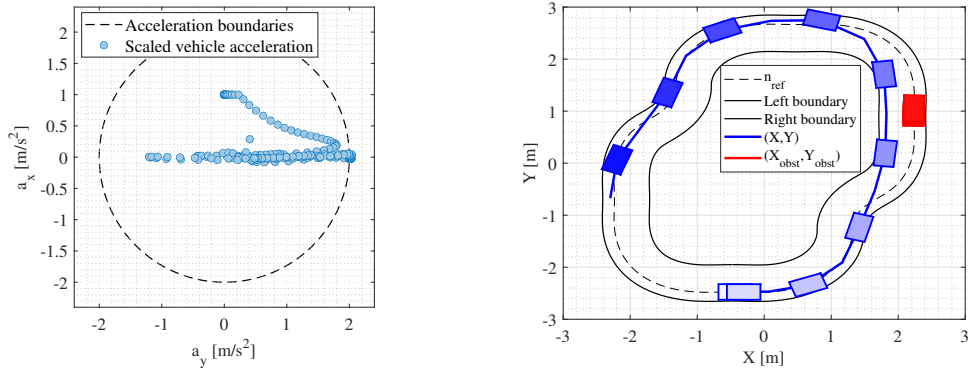
The proposed control strategy is further validated under conditions involving a moving obstacle. In this experimental test, the autonomous vehicle is tasked with following the predefined track at a reference fixed velocity, while the obstacle, navigated the same track using a Pure Pursuit controller operating at a lower constant velocity of 0.7 m/s.

Figure 7.4 illustrates the evolution of the system states during the collision avoidance maneuver. The main distinction compared to the previous case lies in the computation of the lateral deviation reference $n_{ref,obst}$, which is now dynamically adapted based on the real-time position of the obstacle. In this scenario, the controller exhibits greater difficulty in precisely tracking the reference trajectory; however, the resulting lateral deviation error remains bounded within approximately 0.2 meters.

Figure 7.5a shows the control command. Initially, high deceleration is observed to adjust the vehicle speed to the target one while approaching the obstacle, followed



(a) Command control actions evolution along the circuit. CA with static obstacle using Reference lateral deviation generator logic



(b) Longitudinal and lateral acceleration plot (c) Vehicle (blue) and static obstacle (red) poses

Fig. 7.3 Commands, vehicle accelerations and poses evolution along the maneuver. Collision avoidance with Reference lateral deviation generator logic behaviour with static obstacle

by smaller corrections. Furthermore, the $\Delta\delta$ control action clearly exhibits smooth, oscillatory lateral shifts, consistent with the double logistic reference lateral deviation profile. Vehicle accelerations, shown in Fig. 7.5b, remain within the vehicle's Kamm's envelope, ensuring feasible control inputs throughout the maneuver. Finally, Fig. 7.5c depicts the vehicle and obstacle positions over time, where identical transparency levels of the rectangular markers correspond to the same time instants. It can be clearly observed that the vehicle performs a smooth lateral deviation to avoid the obstacle and subsequently re-aligns with the centerline of the lane. The

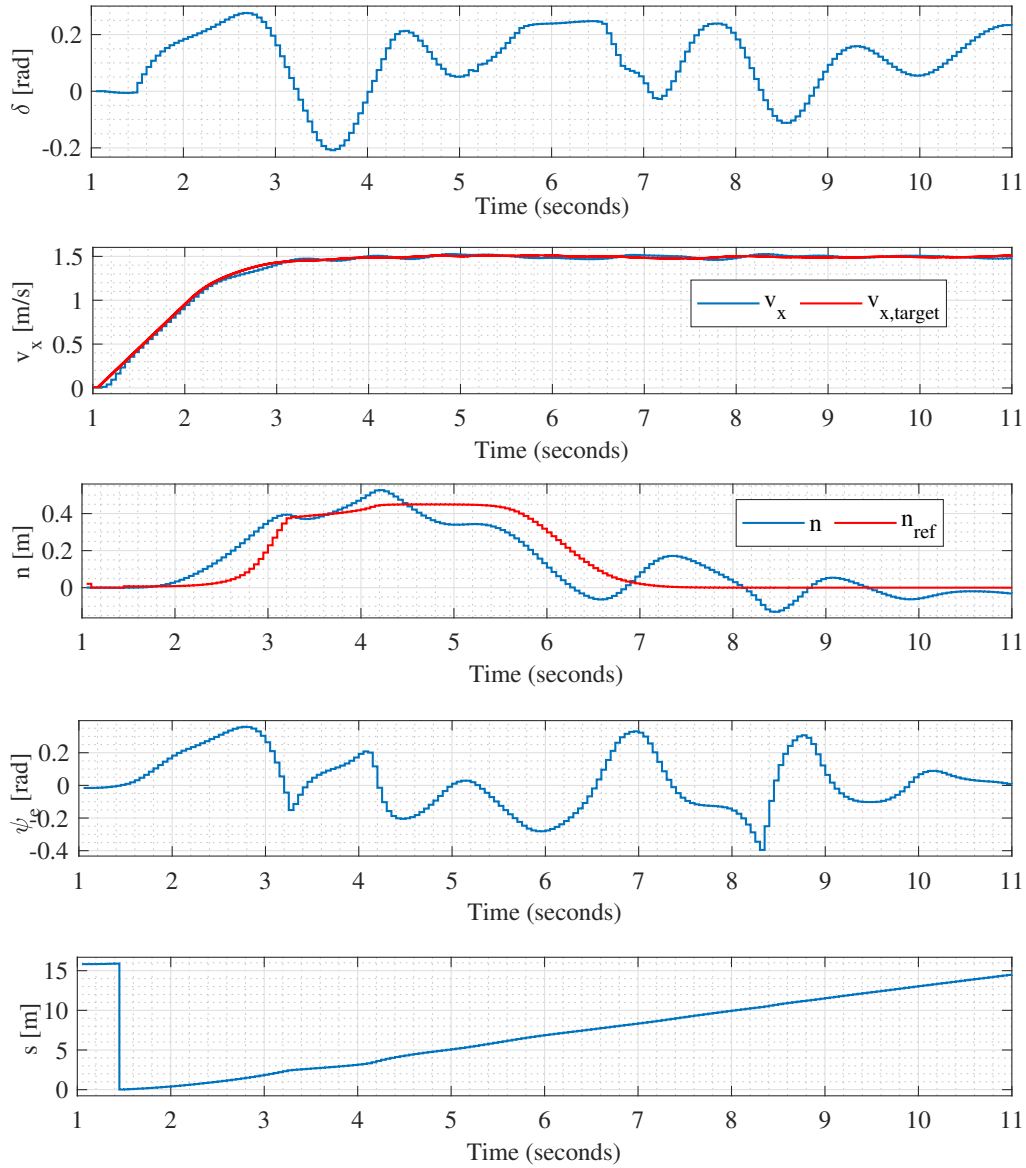
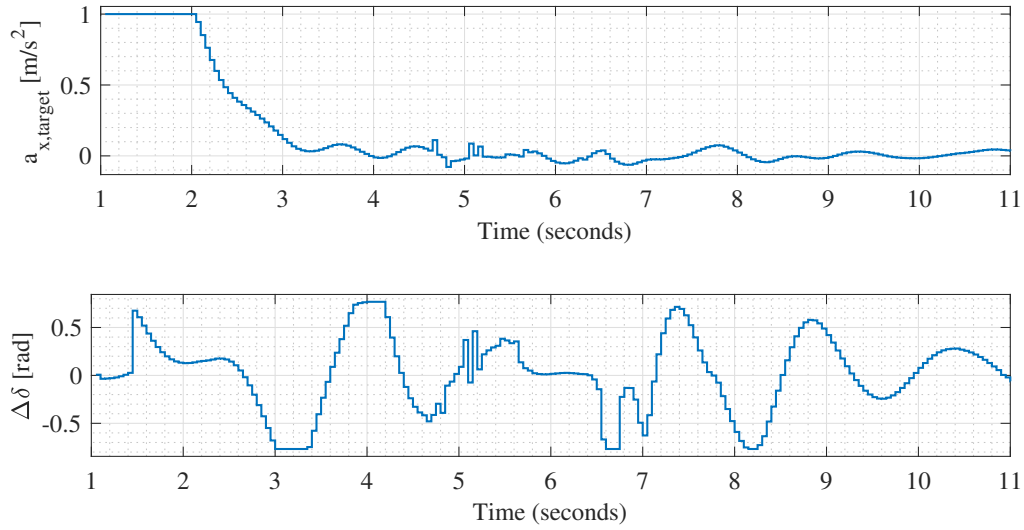


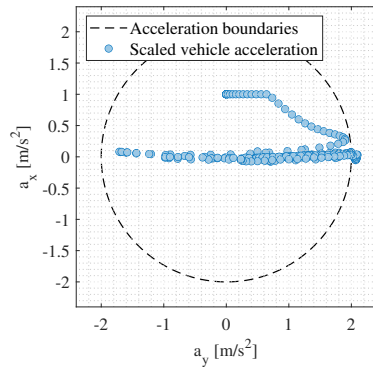
Fig. 7.4 States evolution along the maneuver. Collision avoidance with dynamic obstacle using Reference lateral deviation generator logic

reference lateral deviation generator based on a double logistic profile effectively enables smooth obstacle avoidance, also in the condition of dynamic obstacles.

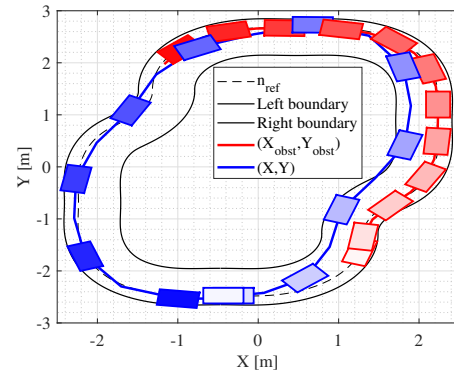
The experimental results are also reported using the travelled distance on the x-axis for both static (Figures 7.6 and 7.7) and dynamic obstacles (Figures 7.8 and 7.9).



(a) Command control actions evolution along the circuit. CA with dynamic obstacle using Reference lateral deviation generator logic



(b) Longitudinal and lateral acceleration plot



(c) Vehicle (blue) and static obstacle (red) poses

Fig. 7.5 Commands, vehicle accelerations and poses evolution along the maneuver. Collision avoidance with Reference lateral deviation generator logic behaviour with moving obstacle

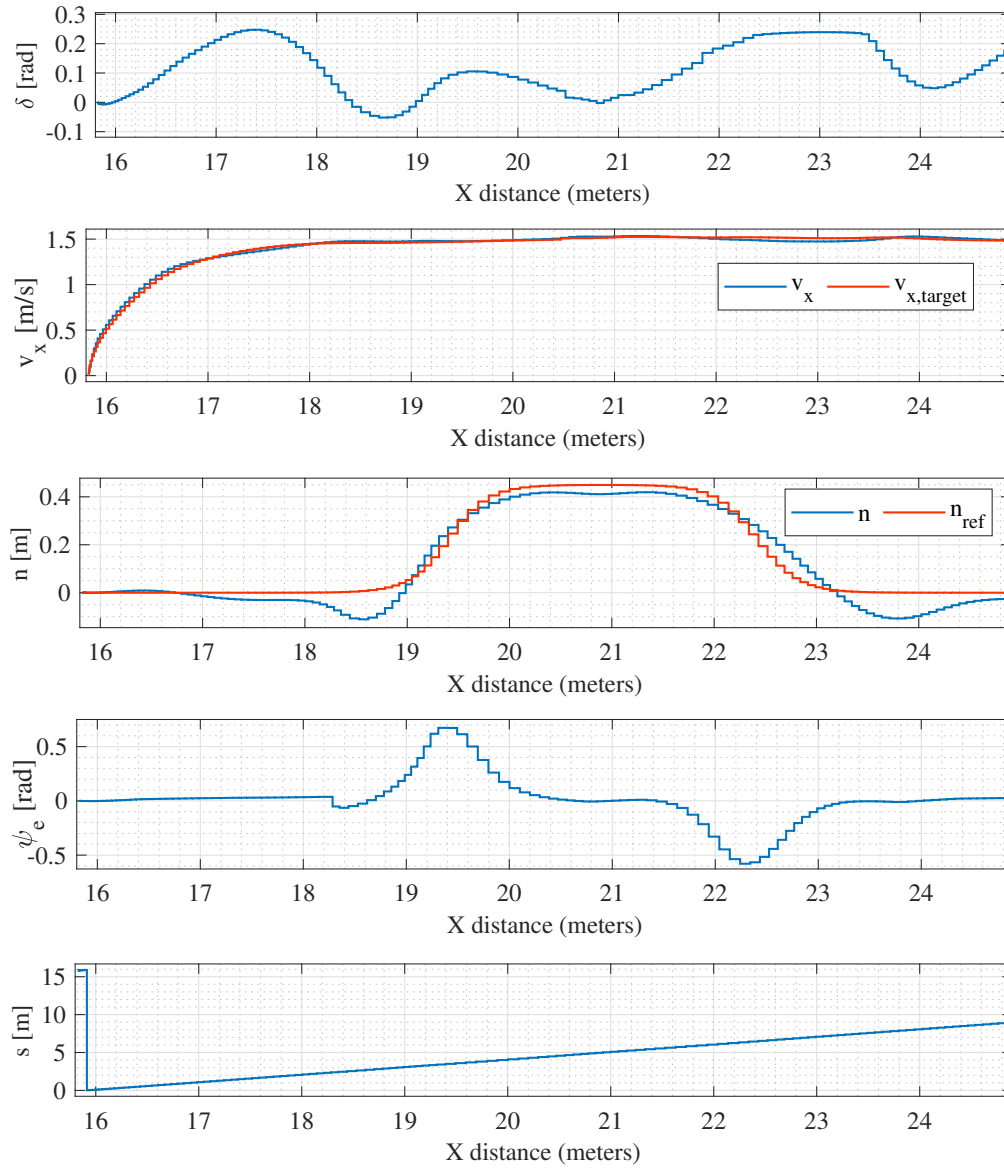


Fig. 7.6 States evolution along the maneuver. Collision avoidance with static obstacle using Reference lateral deviation generator logic. Experimental results are reported using travelled distance on the x-axis.

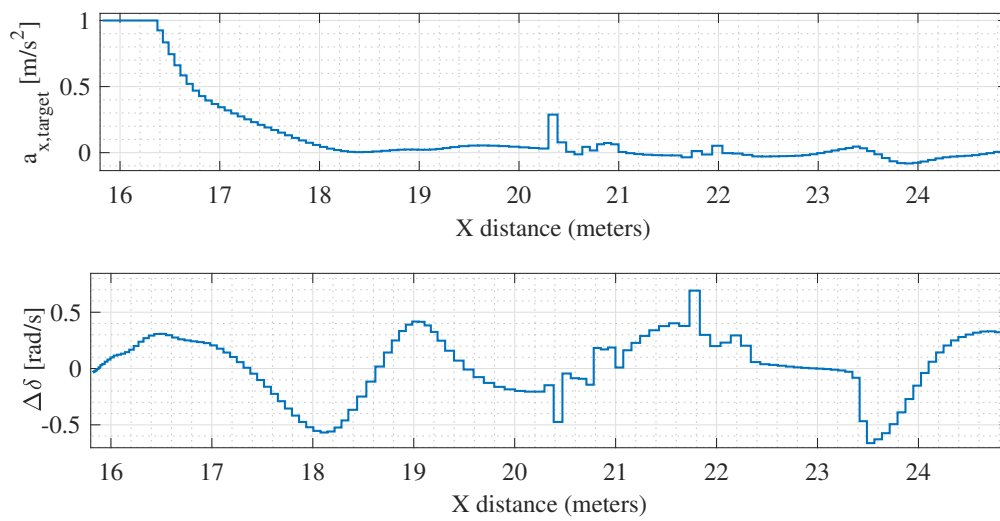


Fig. 7.7 Commands control actions evolution along the maneuver. CA with static obstacle using Reference lateral deviation generator logic. Experimental results are reported using travelled distance on the x-axis.

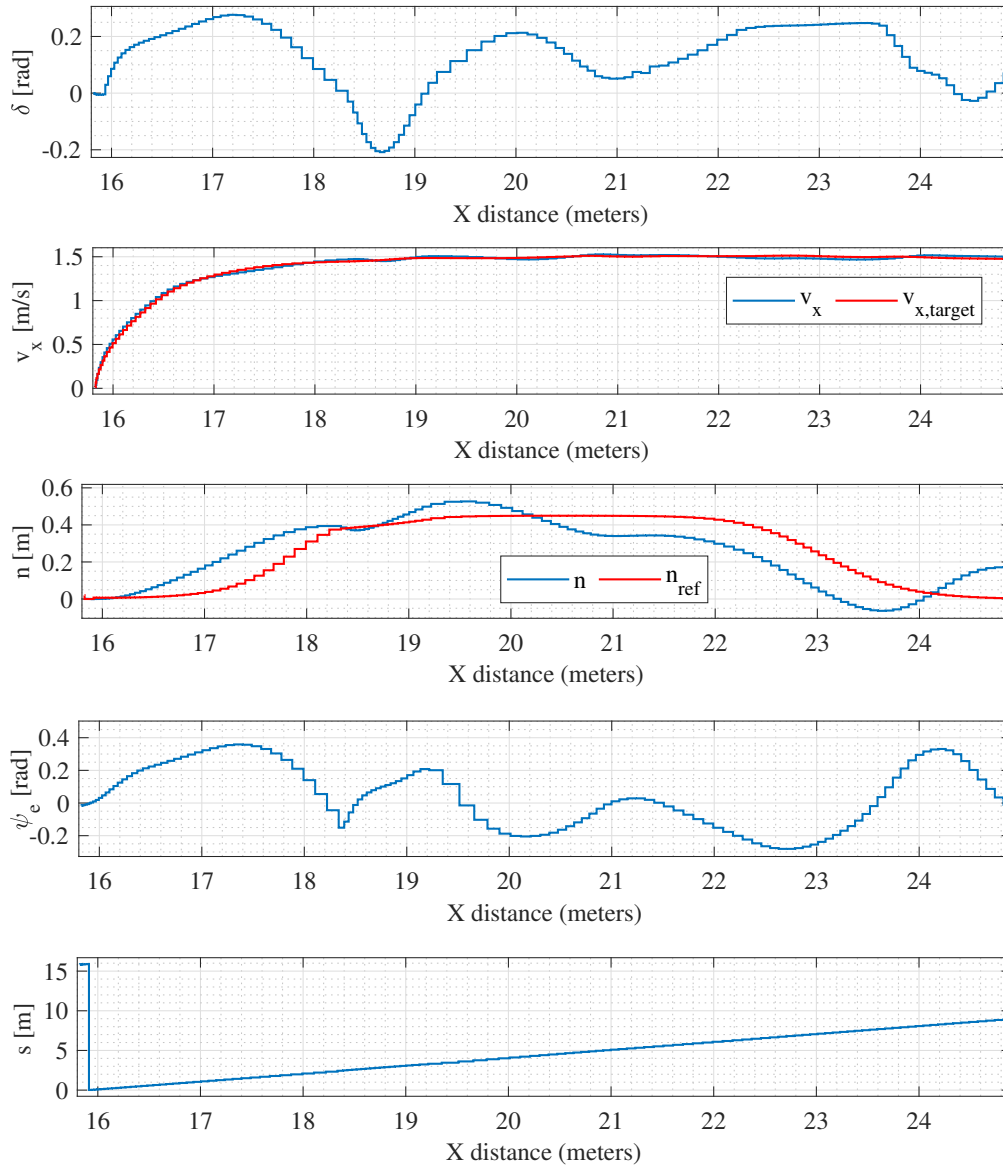


Fig. 7.8 States evolution along the maneuver. Collision avoidance with dynamic obstacle using Reference lateral deviation generator logic. Experimental results are reported using travelled distance on the x-axis.

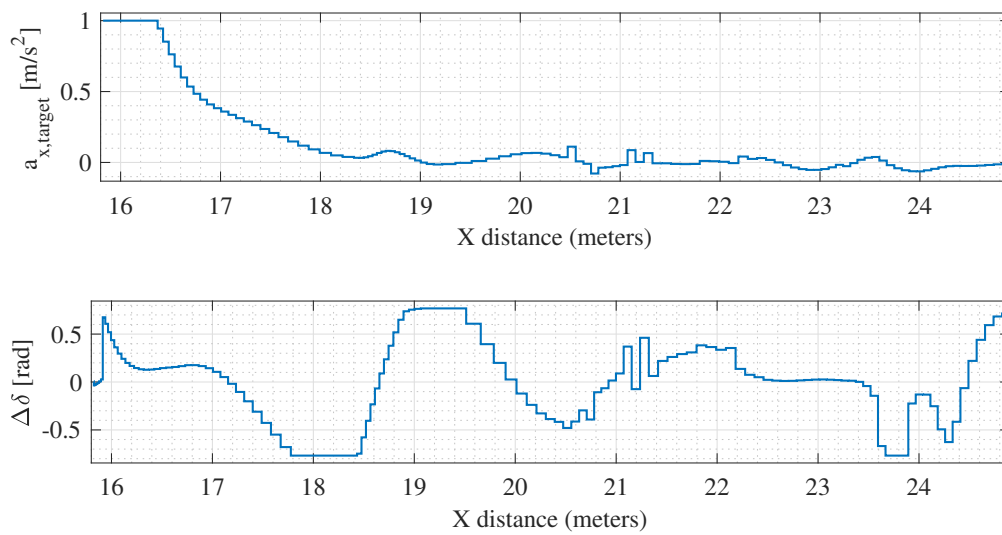


Fig. 7.9 Commands control actions evolution along the maneuver. CA with dynamic obstacle using Reference lateral deviation generator logic. Experimental results are reported using travelled distance on the x-axis.

7.2 Obstacle nonlinear constraint with circle-safe region

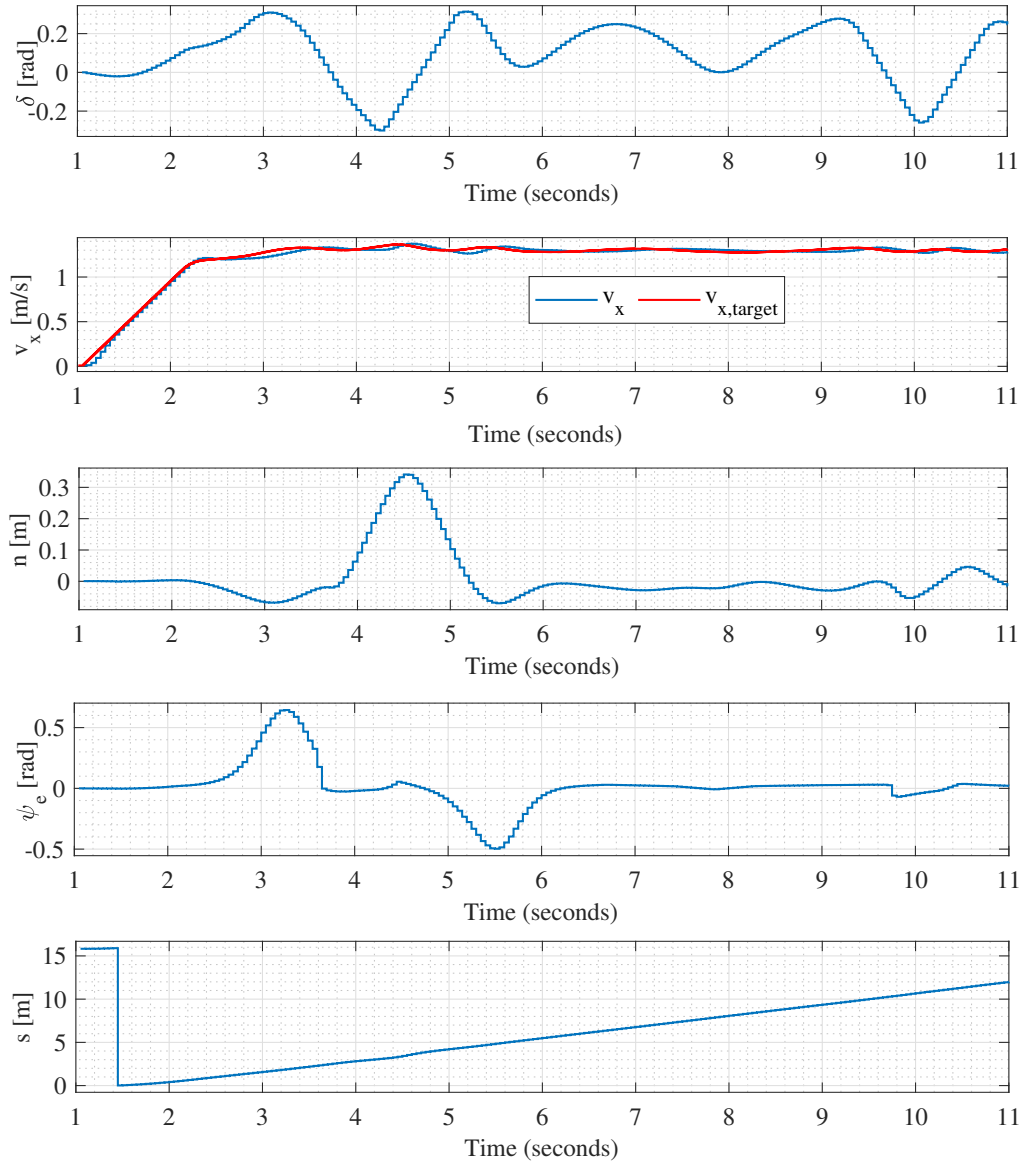
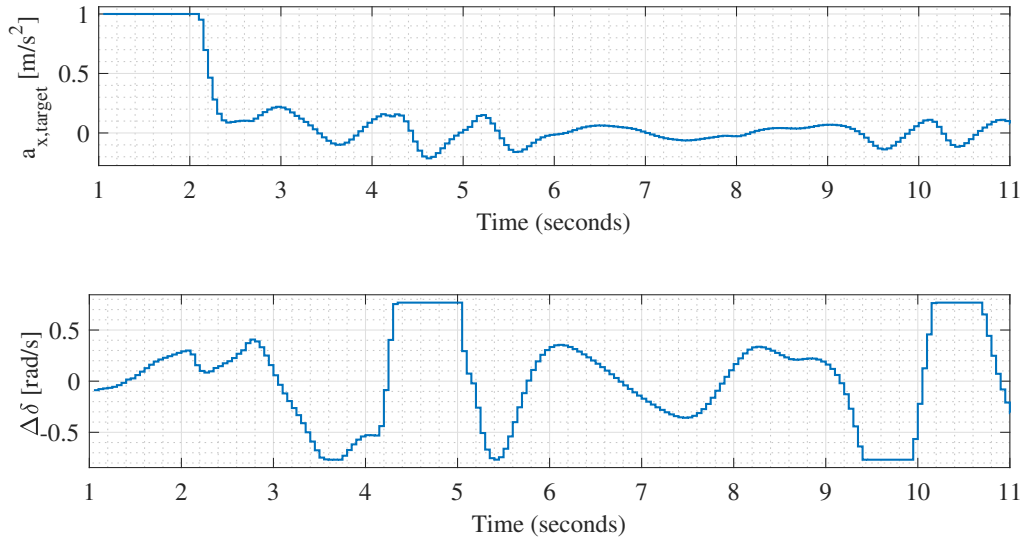
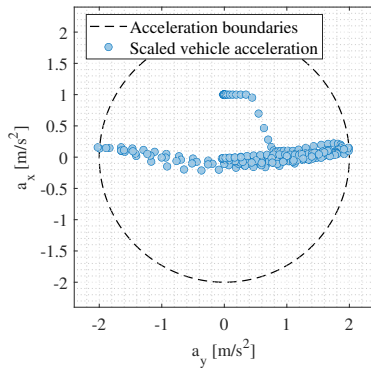


Fig. 7.10 States evolution along the maneuver. Collision avoidance with static obstacle using circle safe region

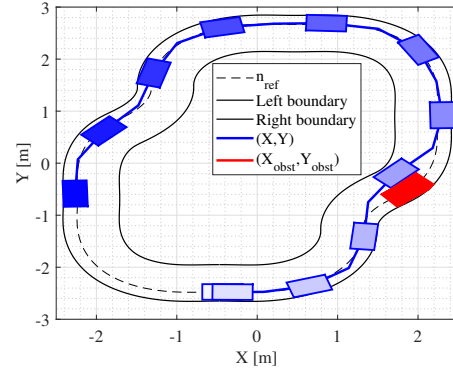
This section presents the results of the collision avoidance (CA) strategy using a non-linear constraint on the obstacle, with a circular safe region. Figures 7.10 and 7.11a show the predicted states and control inputs. The overtaking maneuver occurs



(a) Command control actions evolution along the circuit. CA with static obstacle using circle safe region



(b) Longitudinal and lateral acceleration plot



(c) Vehicle (blue) and static obstacle (red) poses

Fig. 7.11 Commands, vehicle accelerations and poses evolution along the maneuver. Collision avoidance with circle safe region logic behaviour with static obstacle

at approximately 4 seconds, as indicated by the lateral deviation increasing to nearly 0.3 m before decreasing again to follow the reference n_{ref} . Compared to the previous case, there are no oscillations in the control actions that might induce instability in the scaled vehicle. The vehicle's accelerations, shown in Fig. 7.11b, remain within the bounds defined by the Kamm's circle constraint. Figure 7.11c illustrates the positions of the vehicle and the static obstacle (blue and red rectangles, respectively). In contrast to the previous case, where the lateral deviation reference followed a sigmoid profile, leading to a two-step overtaking maneuver resembling a double lane

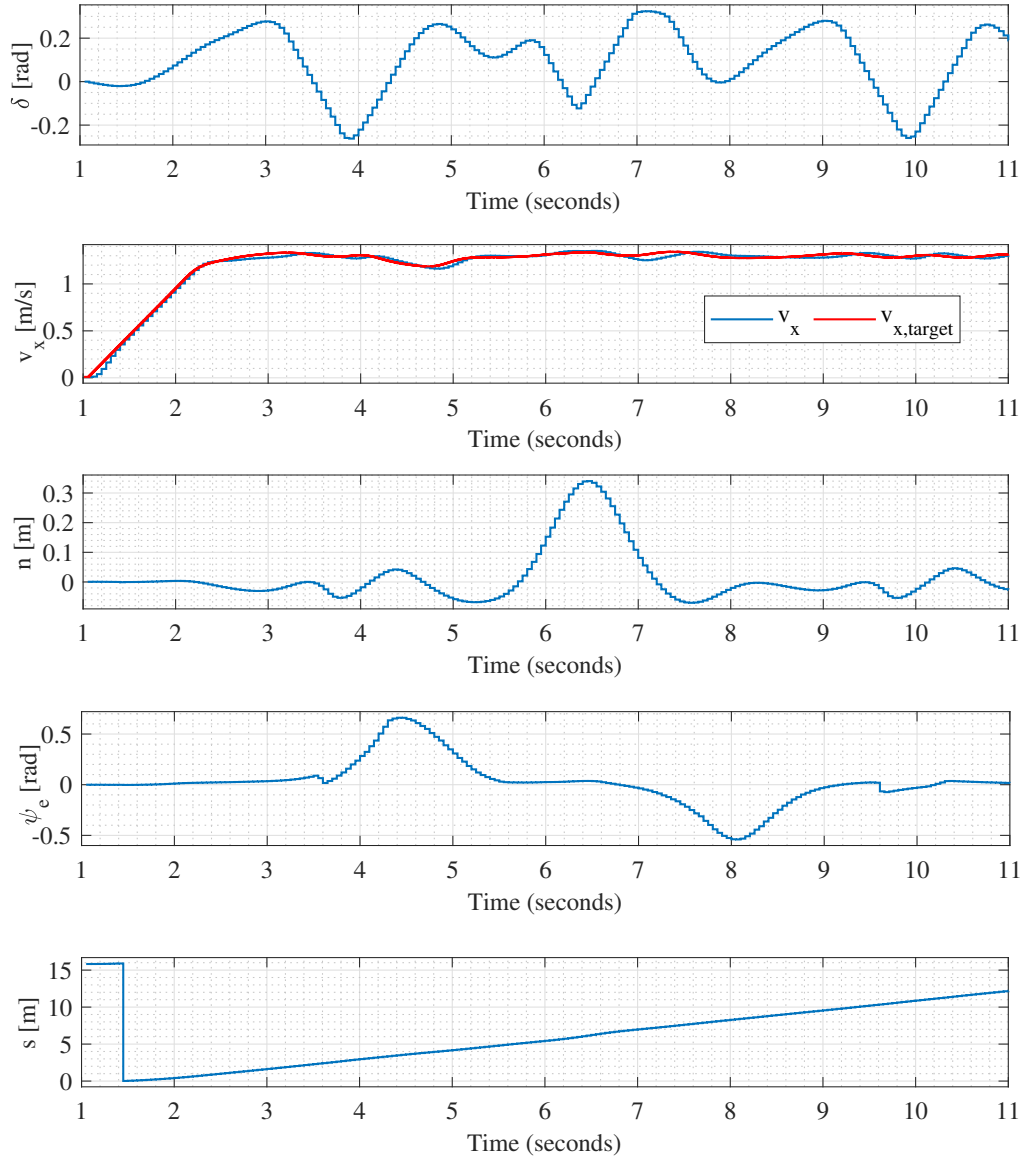
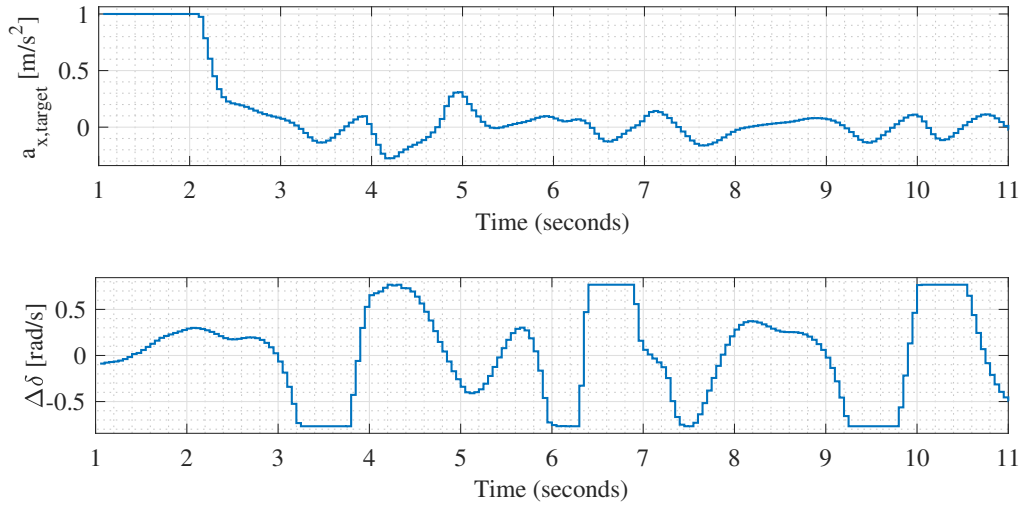
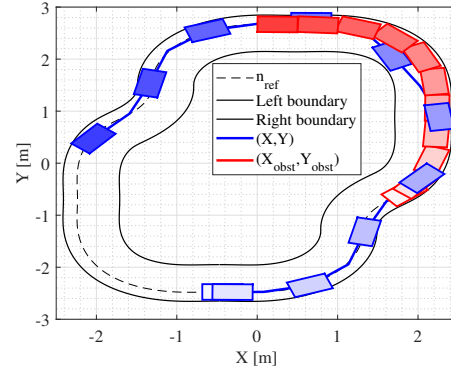
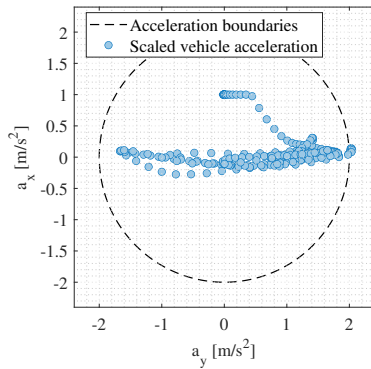


Fig. 7.12 States evolution along the maneuver. Collision avoidance with dynamic obstacle using circle safe region

change, the vehicle here follows the circular obstacle constraint and performs a faster lane change in a single steering action. Figures 7.12, 7.13a, 7.13b, 7.13c correspond to the scenario where this strategy is applied to a moving obstacle. Also in this case, no oscillatory behavior is observed in the control actions, resulting in a smoother maneuver compared to the previous CA approach.



(a) Command control actions evolution along the circuit. CA with dynamic obstacle using circle safe region



(b) Longitudinal and lateral acceleration plot (c) Vehicle (blue) and static obstacle (red) poses

Fig. 7.13 Commands, vehicle accelerations and poses evolution along the maneuver. Collision avoidance with circle safe region logic behaviour with moving obstacle

The experimental results are also reported using the travelled distance on the x -axis for both static (Figures 7.14 and 7.15) and dynamic obstacles (Figures 7.16 and 7.17).

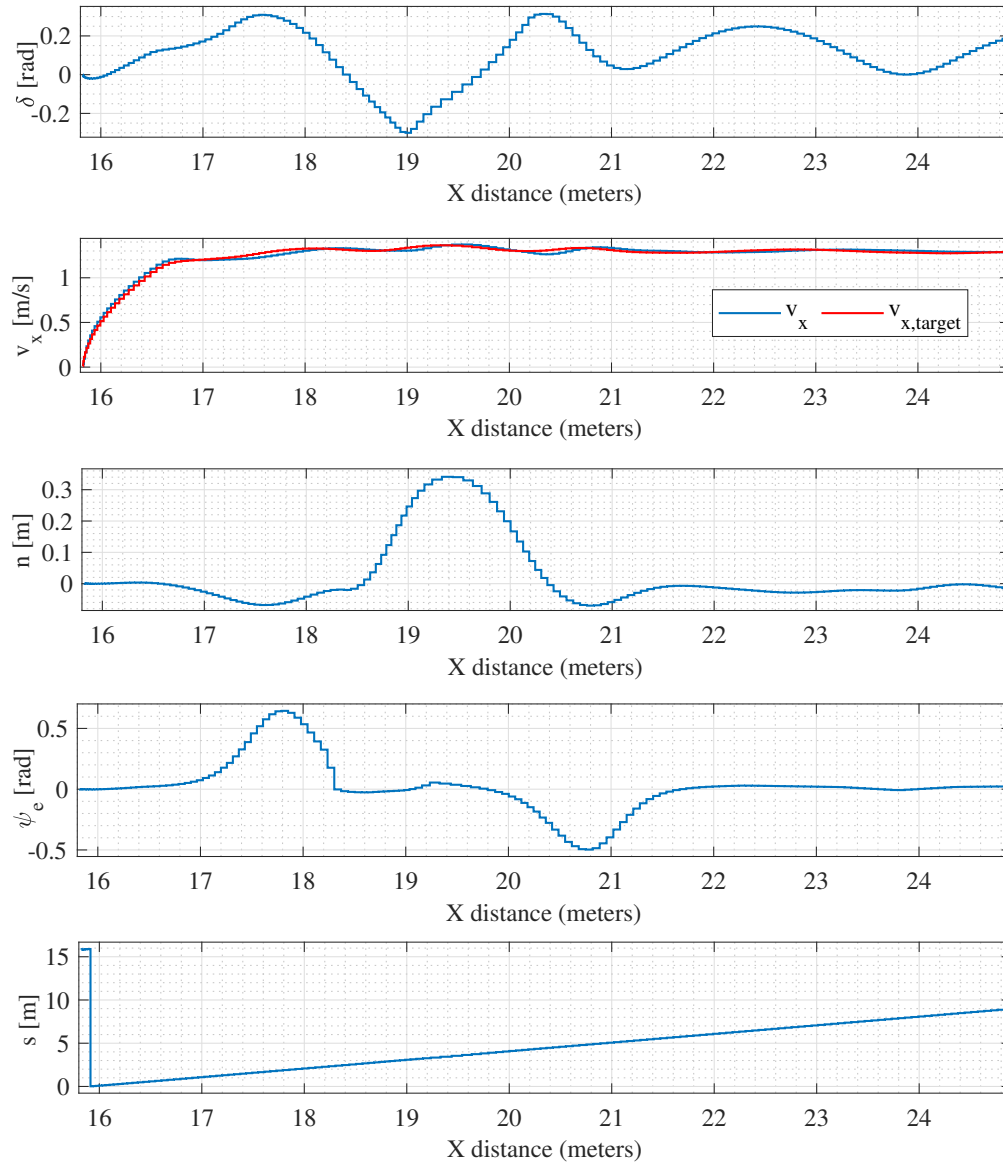


Fig. 7.14 States evolution along the maneuver. Collision avoidance with static obstacle using circle safe region logic. Experimental results are reported using travelled distance on the x-axis.

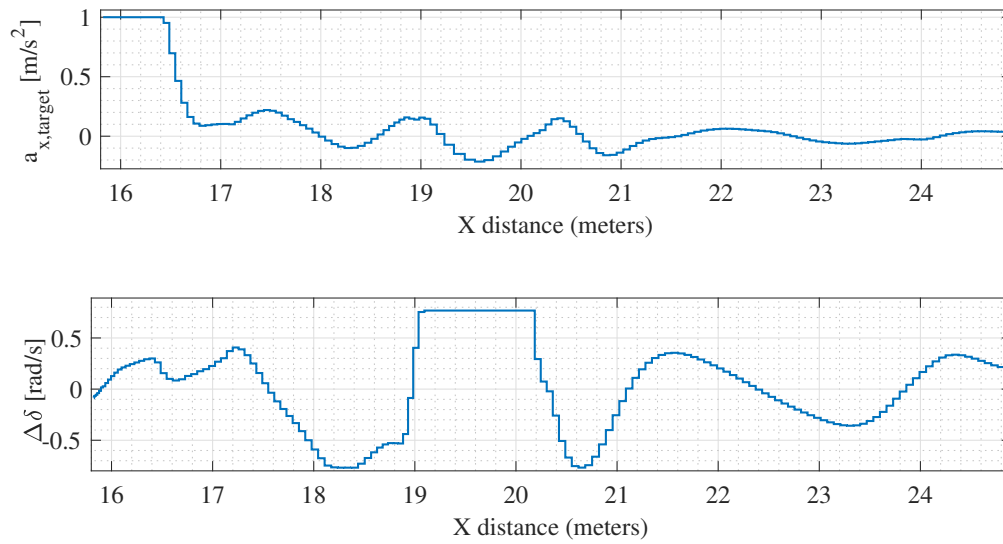


Fig. 7.15 Commands control actions evolution along the maneuver. CA with static obstacle using circle safe region logic. Experimental results are reported using travelled distance on the x-axis.

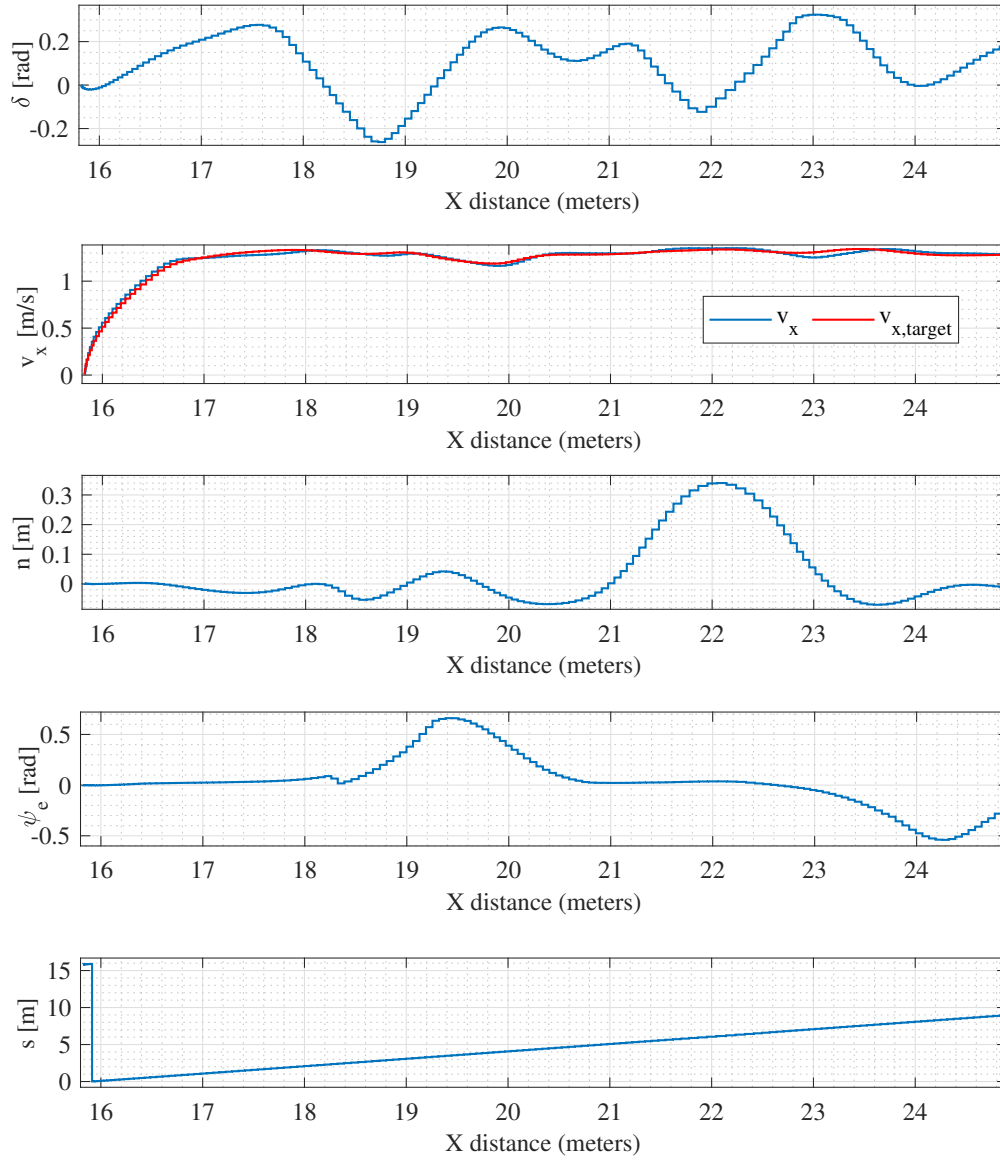


Fig. 7.16 States evolution along the maneuver. Collision avoidance with dynamic obstacle using circle safe region logic. Experimental results are reported using travelled distance on the x-axis.

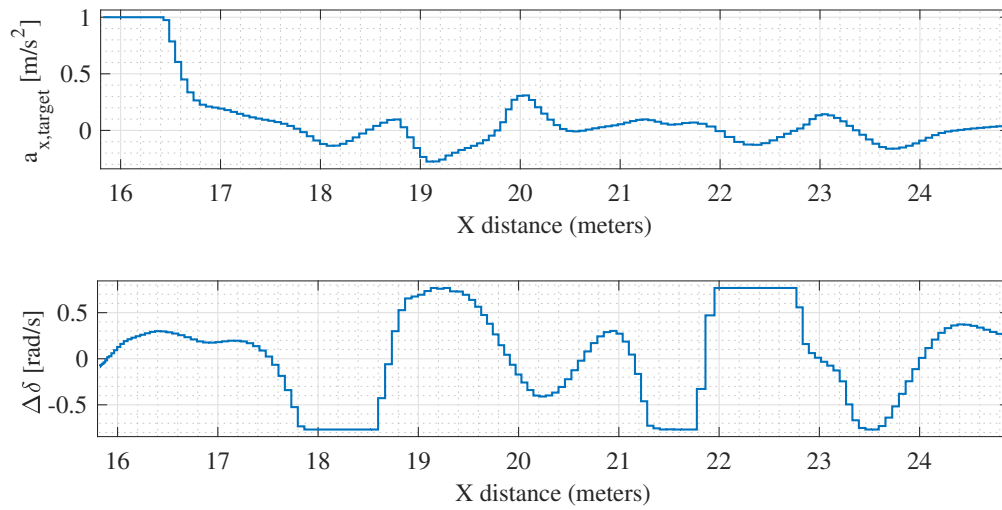


Fig. 7.17 Commands control actions evolution along the maneuver. CA with dynamic obstacle using circle safe region logic. Experimental results are reported using travelled distance on the x-axis.

7.3 Obstacle nonlinear constraint with probabilistic region

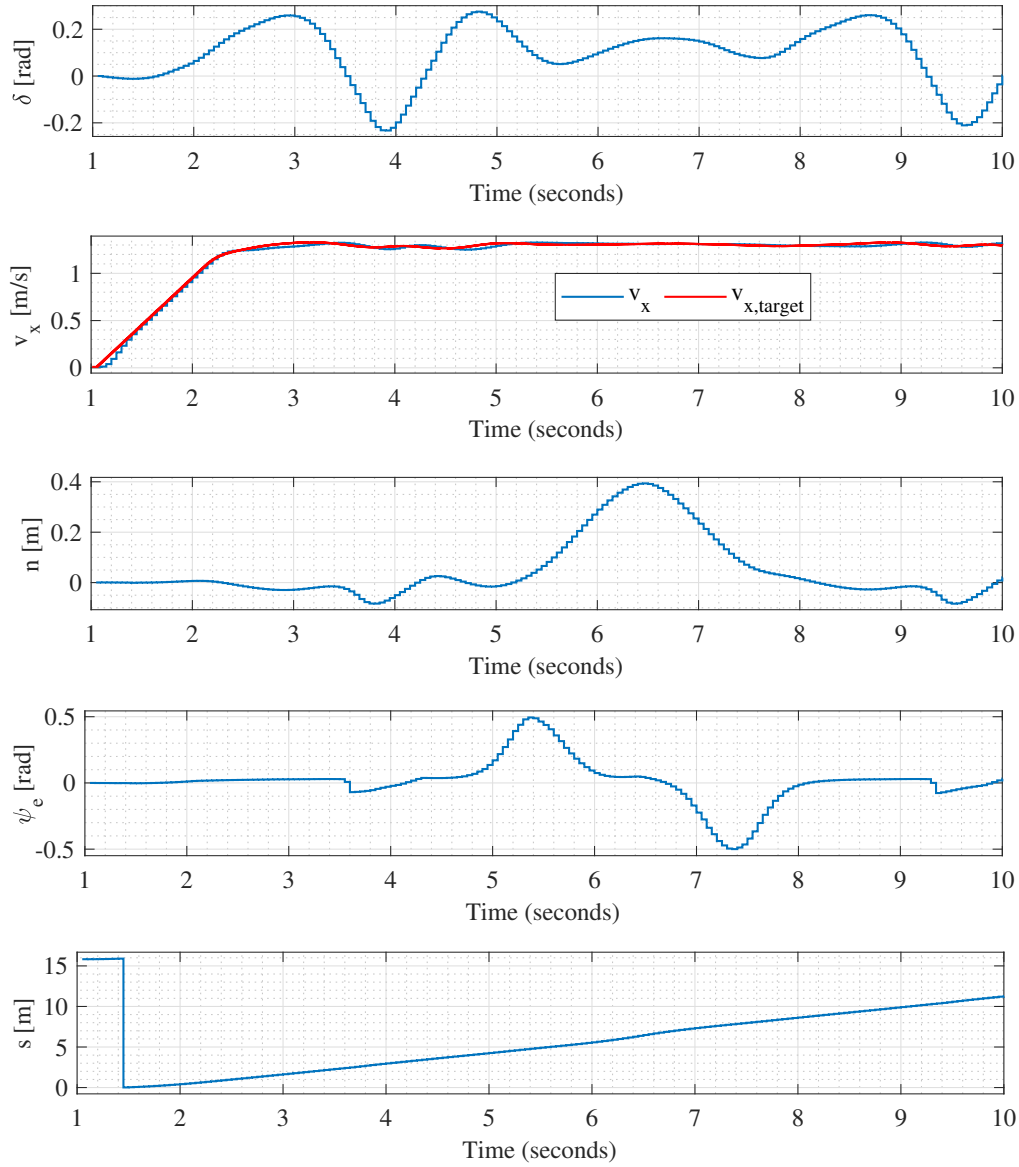
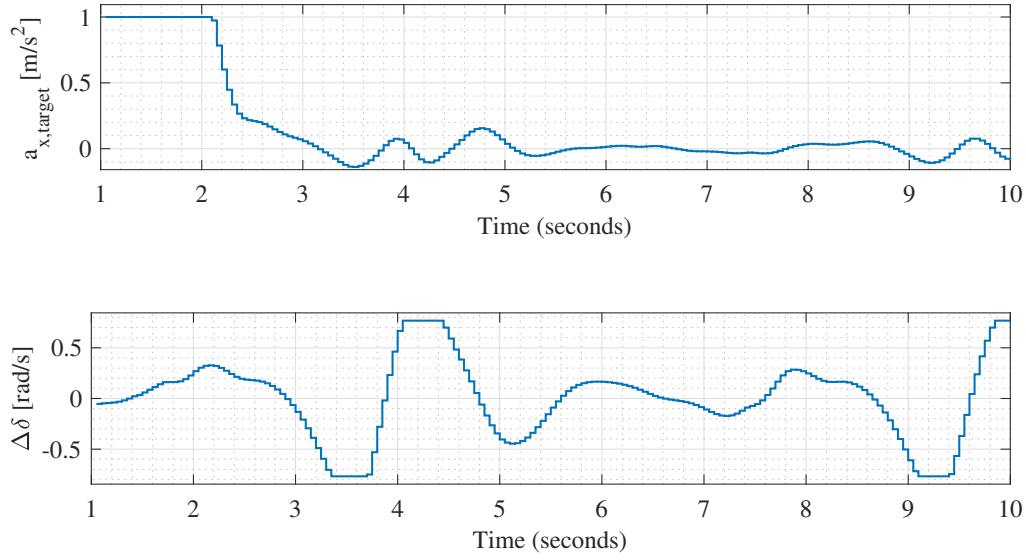
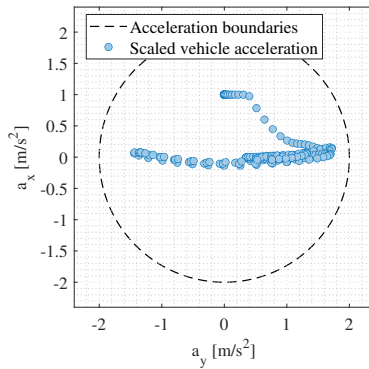


Fig. 7.18 States evolution along the maneuver. Collision avoidance with static obstacle using Gaussian safe region

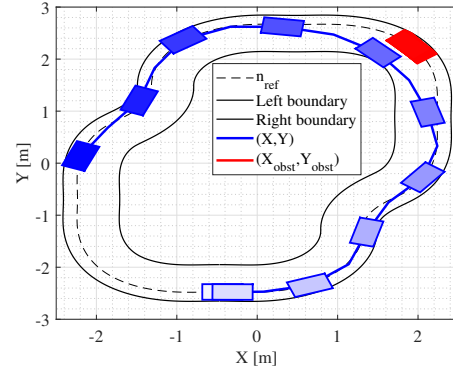
This section discusses the outcomes of the collision avoidance (CA) strategy that employs a non-linear constraint on the obstacle, formulated through a probabilistic safe region with a stage-varying upper bound on the Gaussian confidence



(a) Command control actions evolution along the circuit. CA with static obstacle using Gaussian safe region



(b) Longitudinal and lateral acceleration plot



(c) Vehicle (blue) and static obstacle (red) poses

Fig. 7.19 Commands, vehicle accelerations and poses evolution along the maneuver. Collision avoidance with Gaussian safe region logic behaviour with static obstacle

level. Figures 7.18 and 7.19a present the evolution of the predicted states and the corresponding control inputs. The overtaking maneuver occurs around 6 seconds, as evidenced by the lateral deviation increasing to approximately 0.4 m before returning to track the reference n_{ref} .

In comparison to the previous approach, no oscillations are observed in the control actions, thus avoiding the onset of instabilities in the scaled vehicle. The vehicle's accelerations, depicted in Fig. 7.19b, consistently respect the boundaries defined by the Kamm's circle constraint. It is particularly noteworthy that this

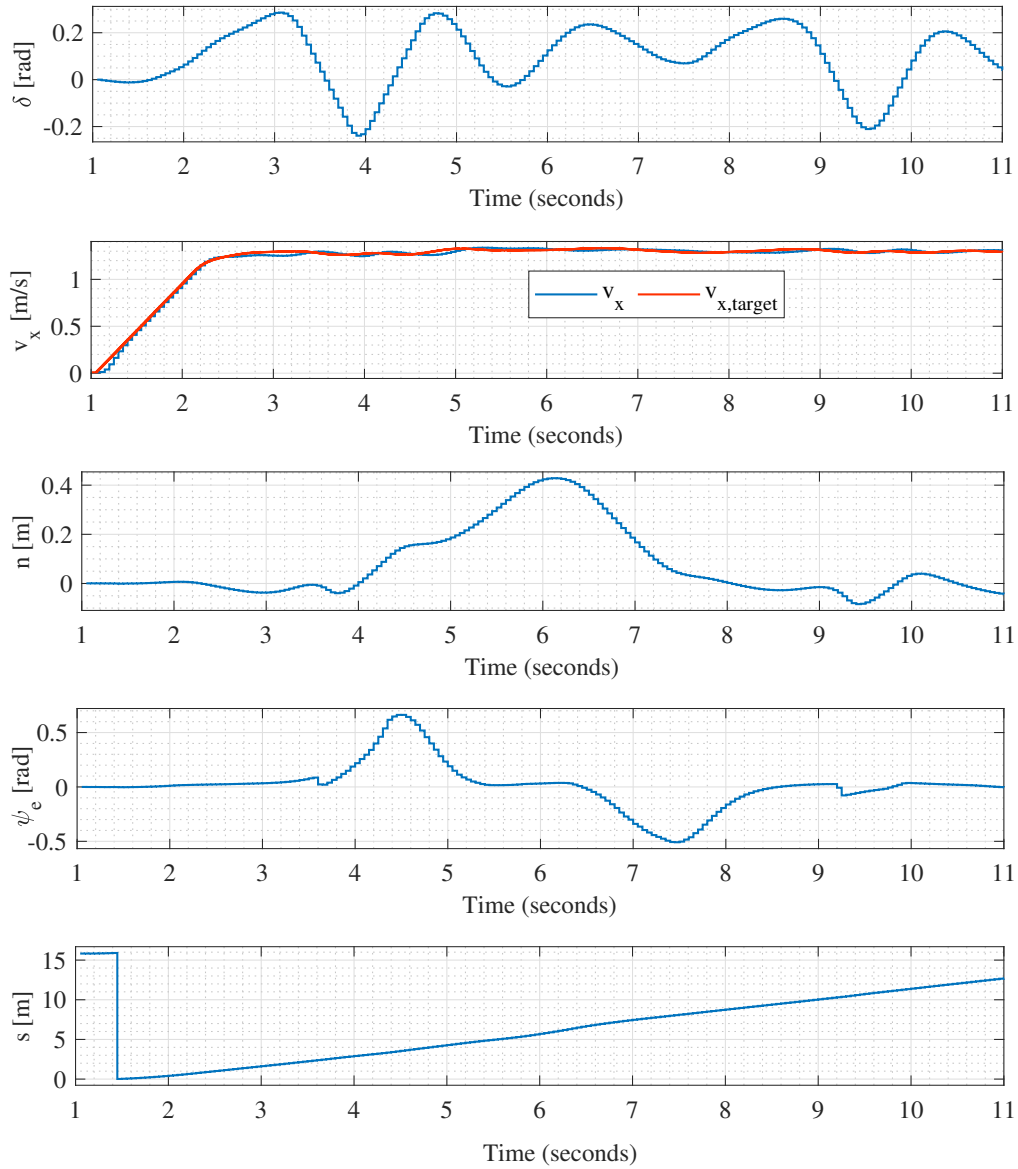
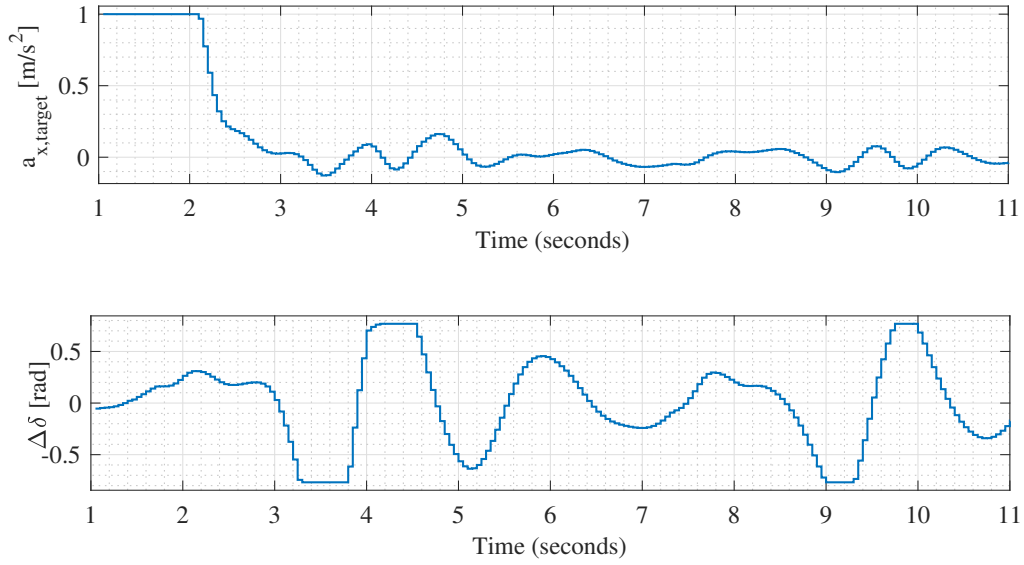


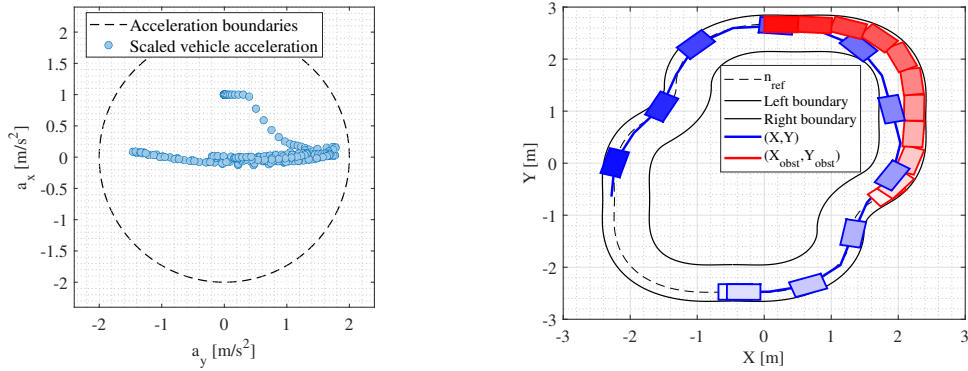
Fig. 7.20 States evolution along the maneuver. Collision avoidance with dynamic obstacle using Gaussian safe region

method achieves a significant reduction in peak lateral acceleration relative to the previous method, despite maintaining the same vehicle operating speed.

Figure 7.19c illustrates the trajectories of both the vehicle and the static obstacle (represented by blue and red rectangles, respectively). In this case, the vehicle follows the trajectory dictated by the evolution of the Gaussian confidence bound



(a) Command control actions evolution along the circuit. CA with dynamic obstacle using Gaussian safe region



(b) Longitudinal and lateral acceleration plot (c) Vehicle (blue) and static obstacle (red) poses

Fig. 7.21 Commands, vehicle accelerations and poses evolution along the maneuver. Collision avoidance with Gaussian safe region logic behaviour with dynamic obstacle

over the prediction horizon, resulting in a smoother and more efficient collision avoidance maneuver compared to the circular safe region approach.

Figures 7.12, 7.13a, 7.13b, and 7.21c report the results when the same strategy is extended to a scenario involving a moving obstacle. Here too, the absence of oscillations in the control inputs ensures a smooth and stable maneuver, demonstrating a clear improvement over the previous CA approach based on reference lateral deviation tracking. Furthermore, vehicle accelerations are consistently lower than those observed with the earlier CA strategies.

The experimental results are also reported using the travelled distance on the x-axis for both static (Figures 7.22 and 7.23) and dynamic obstacles (Figures 7.24 and 7.25).

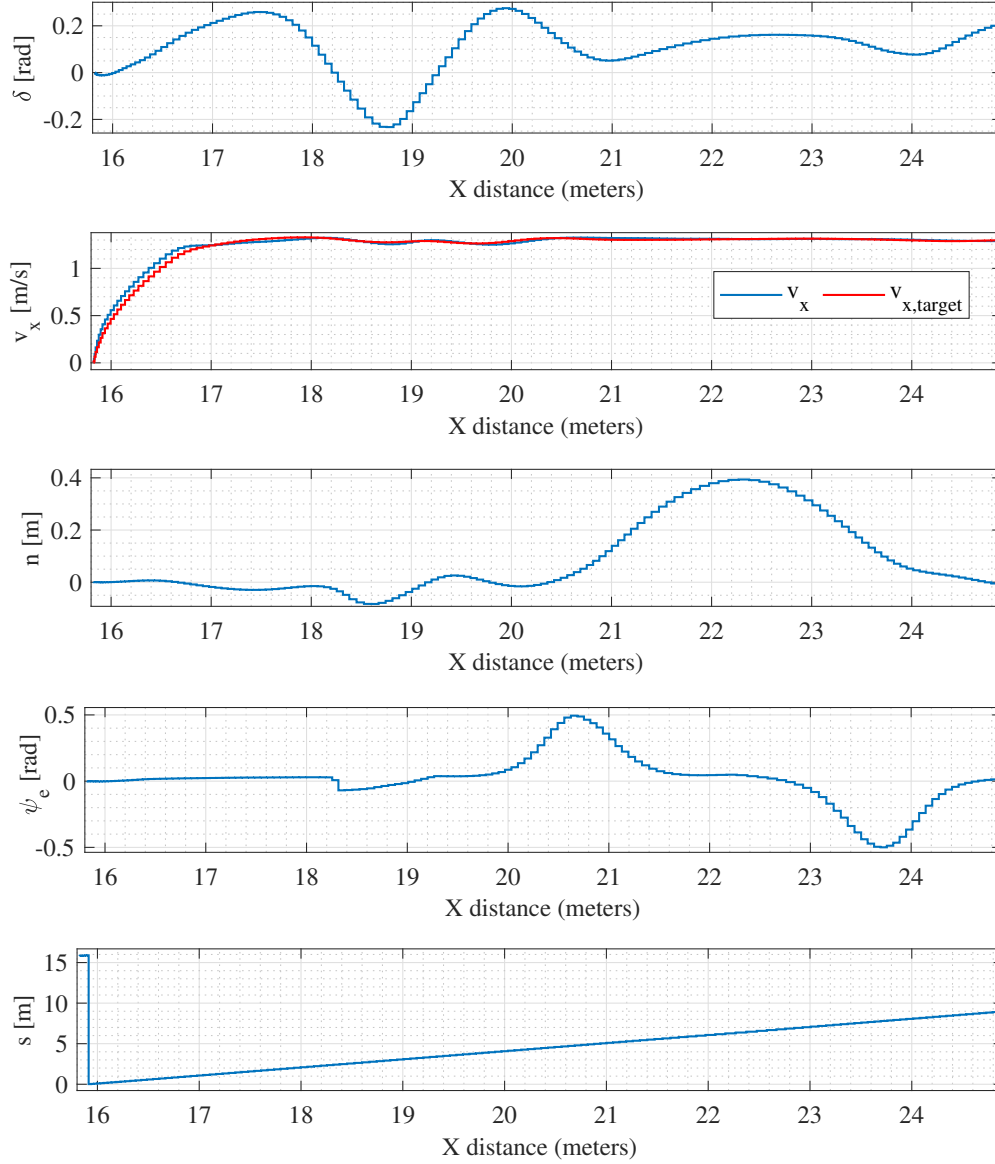


Fig. 7.22 States evolution along the maneuver. Collision avoidance with static obstacle using Gaussian safe region logic. Experimental results are reported using travelled distance on the x-axis.

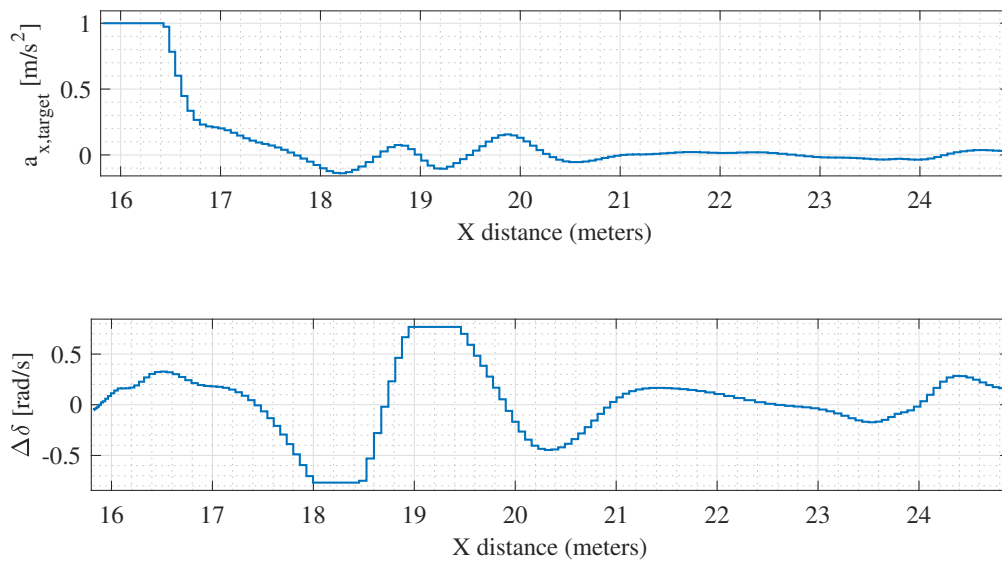


Fig. 7.23 Commands control actions evolution along the maneuver. CA with static obstacle using Gaussian safe region logic. Experimental results are reported using travelled distance on the x-axis.

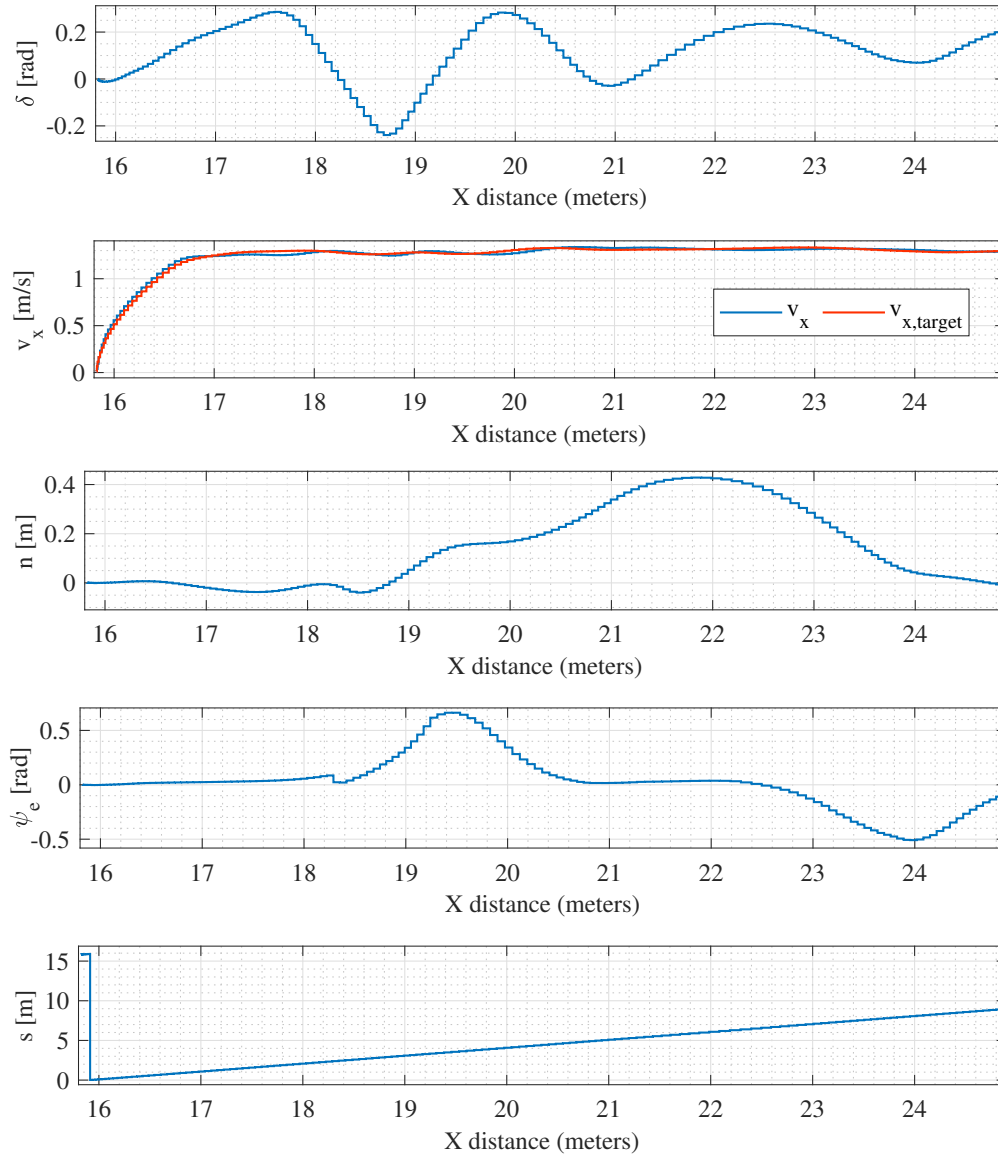


Fig. 7.24 States evolution along the maneuver. Collision avoidance with dynamic obstacle using Gaussian safe region logic. Experimental results are reported using travelled distance on the x-axis.

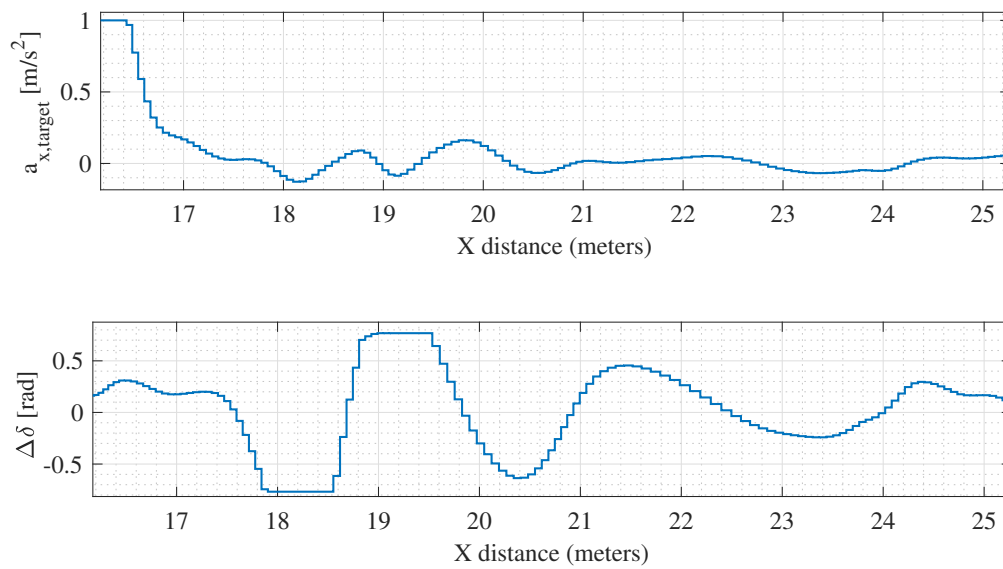


Fig. 7.25 Commands control actions evolution along the maneuver. CA with dynamic obstacle using Gaussian safe region logic. Experimental results are reported using travelled distance on the x-axis.

Table 7.1 CA strategy comparison for static obstacle scenario

	Collision Avoidance with static obstacle					
	$a_{x,max}[m/s^2]$	$a_{x,min}[m/s^2]$	$a_{y,max}[m/s^2]$	$a_{y,min}[m/s^2]$	$x_{rel}[m]$	$y_{rel}[m]$
Sigmoid n_{ref} generator CA	0.96	-1.5	0.95	-0.89	2.5	0.38
Circle safe-region CA	0.91	-1.4	0.82	-0.73	2.3	0.32
Prob. safe-region CA + stage-varying upper constraint on conf. bound level	0.85	-1.3	0.68	-0.61	2.0	0.3

Table 7.2 CA strategy comparison for dynamic obstacle scenario

	Collision Avoidance with dynamic obstacle					
	$a_{x,max}[m/s^2]$	$a_{x,min}[m/s^2]$	$a_{y,max}[m/s^2]$	$a_{y,min}[m/s^2]$	$x_{rel}[m]$	$y_{rel}[m]$
Sigmoid n_{ref} generator CA	1.0	-1.6	1.1	-1.0	2.7	0.41
Circle safe-region CA	0.95	-1.5	0.93	-0.88	2.4	0.35
Prob. safe-region CA + stage-varying upper constraint on conf. bound level	0.88	-1.4	0.72	-0.66	2.1	0.33

Tables 7.1 and 7.2 report the comparative results for the three collision avoidance strategies in scenarios involving static and dynamic obstacles, respectively. The evaluation metrics include the minimum and maximum longitudinal and lateral accelerations ($a_{x,min}$, $a_{x,max}$, $a_{y,min}$, $a_{y,max}$), as well as the relative longitudinal and lateral distances between the vehicle and the obstacle (x_{rel} , y_{rel}). Among the three methods, the approach based on dual-logistic lateral reference generation exhibits the poorest performance, whereas the Gaussian-based probabilistic safe region with stage-varying upper constraints along the prediction horizon yields the best results across all metrics.

Chapter 8

Conclusions

This dissertation presented the development and validation of a scaled-vehicle testbed for the testing of autonomous driving control algorithms, with a particular focus on collision avoidance (CA) strategies. The framework enables both single- and multi-agent experiments, offering a bridge between simulation environments and real-world deployments.

After a thorough review of existing small-scale platforms, the proposed testbed was introduced, emphasizing the preservation of the scaled vehicle's dynamic characteristics through careful design choices, including the integration of custom electronics and a real-time control system based on automotive communication standards.

A dynamic vehicle model was identified through experimental testing, serving as the foundation for the development of advanced control strategies. Subsequently, several motion control problems were formulated, addressing both racing-oriented and passenger-car-oriented applications, with a particular focus on Model Predictive Control (MPC) formulations.

Moreover, three different collision avoidance strategies have been implemented:

- A reference lateral deviation generator logic,
- A non-linear obstacle constraint using a circular safe region,
- A non-linear obstacle constraint based on a probabilistic Gaussian safe region with a dynamic constraint bound.

Comparative experimental results demonstrated that the probabilistic safe region approach yielded the best performance in both static and dynamic obstacle scenarios. Specifically, this strategy provided smoother maneuvers, lower lateral accelerations, and reduced oscillations in the control inputs, thereby enhancing the stability and safety of the vehicle, even in dynamic environments.

The proposed approach highlights the potential of scaled-vehicle infrastructures as effective tools for the early-stage development and validation of autonomous vehicle algorithms. By enabling rapid prototyping, reproducibility, and cost efficiency, such platforms represent a critical step toward bridging the gap between academic research and full-scale industrial applications.

However, the proposed platform exhibits several limitations at both the modeling and hardware levels, which justify the choice of longitudinal acceleration and steering angle as control inputs. Firstly, the scaled vehicle is equipped with a single electric motor that handles traction, braking, and coasting via the ESC. This motor is mechanically connected to the rear wheels through a single-stage reduction gear and an open differential. Due to this configuration, advanced actuation strategies such as torque vectoring, which require independent control of individual wheel torques, are not feasible and thus fall outside the scope of this work. Moreover, the vehicle model employed in the NMPC formulation is purely kinematic and defined in the Frenet reference frame. It does not account for complex tire dynamics such as lateral slip, load transfer, or combined forces. The model assumes negligible lateral velocity at the rear axle (which is non-steerable) and small steering angles, allowing for trigonometric simplifications that yield a computationally efficient structure. While this abstraction is suitable for many trajectory tracking and obstacle avoidance tasks, it limits the controller's capability to manage highly dynamic or near-limit maneuvers where tire forces play a dominant role. For this reason, the focus of the work is not on executing obstacle avoidance at the handling limits, but rather on incorporating the obstacle's motion into the control problem, enabling predictive strategies that avoid such extreme scenarios altogether.

Furthermore, the proposed scaled vehicle does not feature wheel encoders. Although some scaled-vehicle platforms include these kinds of sensors, as the CRS framework [59, 92], they are used solely for odometry and localization. The rapid dynamics of wheel slip in the scaled vehicle cannot be effectively regulated with the

traction motor; thus, no low-level torque or wheel slip control is performed at the individual wheel level.

Beyond these modeling and actuation constraints, additional limitations stem from the experimental setup itself. The control loop experiences a total latency of up to 89 ms due to communication and processing delays. While acceptable for many tasks, this delay restricts the system's ability to perform fast, reactive maneuvers in highly dynamic environments. Moreover, the platform relies exclusively on an external vision-based localization system and lacks onboard perception modules such as cameras, LiDAR, or radar. As a result, the current framework focuses solely on trajectory planning and control, without addressing perception robustness or sensor fusion strategies.

Although the architecture is compatible with multi-agent scenarios, the implementation and experimental validation presented in this work are limited to single-vehicle applications. Cooperative strategies and vehicle-to-vehicle communication have not yet been explored.

These limitations are inherent to the chosen modeling simplifications and experimental setup. However, they also represent opportunities for future work, particularly in expanding the control framework to incorporate richer actuation models, real-time onboard perception, and coordinated multi-agent strategies, as well as the integration of perception and decision-making modules to further approach the complexity of real-world autonomous systems.

8.1 Outlook

There are several theoretical and computational research questions that merit further investigation in future work. First, an important challenge arises when the precise pose information of the vehicle, currently provided by the vision system acting as a GPS substitute within the control loop, is no longer available or reliable. In such a case, it would be necessary to explore mapping and localization algorithms, potentially based on virtualized or onboard sensors, to ensure accurate state estimation. Second, it would be crucial to develop a more general optimization framework capable of handling a variable number of obstacles. The current formulation, while successfully applied to a path-tracking problem on a closed circuit, considers only a

single moving obstacle. It also assumes the presence of a high-level logic external to the MPC that identifies and provides the pose of the nearest obstacle to update the references for the collision avoidance strategy. In real-world scenarios, however, such an assumption is unrealistic, as multiple dynamic obstacles must be managed simultaneously and in real-time.

References

- [1] Hao Ju, Rongshun Juan, Randy Gomez, Keisuke Nakamura, and Guangliang Li. Transferring policy of deep reinforcement learning from simulation to reality for robotics. *Nature Machine Intelligence*, 4(12):1077–1087, 2022.
- [2] Budi Padmaja, CH VKNSN Moorthy, N Venkateswarulu, and Myneni Madhu Bala. Exploration of issues, challenges and latest developments in autonomous cars. *Journal of Big Data*, 10(1):61, 2023.
- [3] Rajesh Rajamani. *Vehicle dynamics and control*. Springer Science & Business Media, 2011.
- [4] Robin Verschueren, Gianluca Frison, Dimitris Kouzoupis, Jonathan Frey, Niels van Duijkeren, Andrea Zanelli, Branimir Novoselnik, Thivaharan Albin, Rien Quirynen, and Moritz Diehl. acados—a modular open-source framework for fast embedded optimal control. *Mathematical Programming Computation*, 14(1):147–183, 2022.
- [5] Espressif Systems. ESP32-WROOM-32D Datasheet, 2023. Accessed: 2025-03-25.
- [6] Felipe Caleffi, Lauren da Silva Rodrigues, Joice da Silva Stamboroski, Brian Vargas Rorig, Maria Manoela Cardoso dos Santos, Vanessa Zuchetto, and Ítalo Brum Raguzzoni. A systematic review of hardware technologies for small-scale self-driving cars. *Ciência e Natura*, 45(esp. 1):84071–84071, 2023.
- [7] Francesco Mondada, Edoardo Franzini, and Paolo Ienne. Mobile robot miniaturisation: A tool for investigation in control algorithms. In *Experimental Robotics III: The 3rd International Symposium, Kyoto, Japan, October 28–30, 1993*, pages 501–513. Springer, 1994.
- [8] Paul FMJ Verschure, Thomas Voegtlin, and Rodney J Douglas. Environmentally mediated synergy between perception and behaviour in mobile robots. *Nature*, 425(6958):620–624, 2003.
- [9] LEGO. Lego Mindstorms. <https://www.lego.com/mindstorms>. Accessed: 2025-04-30.
- [10] iRobot Create. iRobot Create. <https://www.irobot.com/>. Accessed: 2025-04-30.

- [11] Branislav Thurskỳ and Gabriel Gašpar. Using pololu's 3pi robot in the education process. 2010.
- [12] Serge Kernbach. Swarmrobot. org-open-hardware microrobotic project for large-scale artificial swarms. *arXiv preprint arXiv:1110.5762*, 2011.
- [13] Michael Rubenstein, Bo Cimini, Radhika Nagpal, and Justin Werfel. Aerobot: An affordable one-robot-per-student system for early robotics education. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6107–6113. IEEE, 2015.
- [14] Daniel Pickem, Myron Lee, and Magnus Egerstedt. The gritsbot in its natural habitat-a multi-robot testbed. In *2015 IEEE International conference on robotics and automation (ICRA)*, pages 4062–4067. IEEE, 2015.
- [15] Sean Wilson, Ruben Gameros, Michael Sheely, Matthew Lin, Kathryn Dover, Robert Gevorkyan, Matt Haberland, Andrea Bertozzi, and Spring Berman. Pheeno, a versatile swarm robotic research and education platform. *IEEE Robotics and Automation Letters*, 1(2):884–891, 2016.
- [16] Liam Paull, Jacopo Tani, Heejin Ahn, Javier Alonso-Mora, Luca Carlone, Michal Cap, Yu Fan Chen, Changhyun Choi, Jeff Dusek, Yajun Fang, et al. Duckietown: an open, inexpensive and flexible platform for autonomy education and research. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1497–1504. IEEE, 2017.
- [17] A René Geist, Jonathan Fiene, Naomi Tashiro, Zheng Jia, and Sebastian Trimpe. The wheelbot: A jumping reaction wheel unicycle. *IEEE Robotics and Automation Letters*, 7(4):9683–9690, 2022.
- [18] David Johnson, Tim Stack, Russ Fish, Daniel Montralio Flickinger, Leigh Stoller, Robert Ricci, and Jay Lepreau. Mobile emulab: A robotic wireless and sensor network testbed. In *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*, pages 1–12. IEEE, 2006.
- [19] Andrew Stubbs, Vladimeros Vladimerou, Adam Thomas Fulford, Derek King, Jeffrey Strick, and Geir E Dullerud. Multivehicle systems control over networks: a hovercraft testbed for networked and decentralized control. *IEEE Control Systems Magazine*, 26(3):56–69, 2006.
- [20] Daniel Pickem, Paul Glotfelter, Li Wang, Mark Mote, Aaron Ames, Eric Feron, and Magnus Egerstedt. The robotarium: A remotely accessible swarm robotics research testbed. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1699–1706. IEEE, 2017.
- [21] Adrián Jiménez-González, Jose Ramiro Martinez-de Dios, and Anibal Ollero. Testbeds for ubiquitous robotics: A survey. *Robotics and Autonomous Systems*, 61(12):1487–1501, 2013.

- [22] Joseph Funke, Paul Theodosis, Rami Hindiyeh, Ganymed Stanek, Krisada Kritatakirana, Chris Gerdes, Dirk Langer, Marcial Hernandez, Bernhard Müller-Bessler, and Burkhard Huhnke. Up to the limits: Autonomous audi tts. In *2012 IEEE intelligent vehicles symposium*, pages 541–547. IEEE, 2012.
- [23] Sebastian Thrun. Toward robotic cars. *Communications of the ACM*, 53(4):99–106, 2010.
- [24] Jonathan Y Goh, Tushar Goel, and J Christian Gerdes. Toward automated vehicle control beyond the stability limits: drifting along a general path. *Journal of Dynamic Systems, Measurement, and Control*, 142(2):021004, 2020.
- [25] Juraj Kabzan, Miguel I Valls, Victor JF Reijgwart, Hubertus FC Hendriks, Claas Ehmke, Manish Prajapat, Andreas Bühler, Nikhil Gosala, Mehak Gupta, Ramya Sivanesan, et al. Amz driverless: The full autonomous racing system. *Journal of Field Robotics*, 37(7):1267–1294, 2020.
- [26] Roborace. Roborace. Online. Accessed: 2025-04-30.
- [27] Alexander Wischnewski, Maximilian Geisslinger, Johannes Betz, Tobias Betz, Felix Fent, Alexander Heilmeyer, Leonhard Hermansdorfer, Thomas Herrmann, Sebastian Huch, Phillip Karle, et al. Indy autonomous challenge-autonomous race cars at the handling limits. In *12th International Munich Chassis Symposium 2021: chassis. tech plus*, pages 163–182. Springer, 2022.
- [28] Brian Goldfain, Paul Drews, Changxi You, Matthew Barulic, Orlin Velez, Panagiotis Tsiotras, and James M Rehg. Autorally: An open platform for aggressive autonomous driving. *IEEE Control Systems Magazine*, 39(1):26–55, 2019.
- [29] Donkey Car. Donkey car. Online. Accessed: 2022-07-30.
- [30] Matthew O’Kelly, Hongrui Zheng, Dhruv Karthik, and Rahul Mangharam. Fltenth: An open-source evaluation environment for continuous control and reinforcement learning. *Proceedings of Machine Learning Research*, 123, 2020.
- [31] AWS DeepRacer. Aws deepracer. Online. Accessed: 2022-07-30.
- [32] Chung H Hsieh, Yao-Li Chuang, Yuan Huang, Kevin K Leung, Andrea L Bertozzi, and Emilio Frazzoli. An economical micro-car testbed for validation of cooperative control strategies. In *2006 American Control Conference*, pages 6–pp. IEEE, 2006.
- [33] Kevin K Leung, Chung H Hsieh, Yuan R Huang, Abhijeet Joshi, Vlad Voroninski, and Andrea L Bertozzi. A second generation micro-vehicle testbed for cooperative control and sensing strategies. In *2007 American Control Conference*, pages 1900–1907. IEEE, 2007.

- [34] Alexander Liniger, Alexander Domahidi, and Manfred Morari. Optimization-based autonomous racing of 1: 43 scale rc cars. *Optimal Control Applications and Methods*, 36(5):628–647, 2015.
- [35] R. Tedrake and the Drake Development Team. Drake: Model-based design and verification for robotics. Technical report, 2019.
- [36] Markus Gifftthaler, Michael Neunert, Markus Stäuble, and Jonas Buchli. The control toolbox—an open-source c++ library for robotics, optimal and model predictive control. In *2018 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAN)*, pages 123–129. IEEE, 2018.
- [37] R. S. L. E. Zurich. Ocs2: Optimal control for switched systems. Technical report, 2022.
- [38] Armin Mokhtarian, Jianye Xu, Patrick Scheffe, Maximilian Kloock, Simon Schäfer, Heeseung Bang, Viet-Anh Le, Sangeet Ulhas, Johannes Betz, Sean Wilson, et al. A survey on small-scale testbeds for connected and automated vehicles and robot swarms: A guide for creating a new testbed. *IEEE Robotics & Automation Magazine*, 2024.
- [39] Tanmay Samak, Chinmay Samak, Sivanathan Kandhasamy, Venkat Krovi, and Ming Xie. Autodrive: A comprehensive, flexible and integrated digital twin ecosystem for autonomous driving research & education. *Robotics*, 12(3):77, 2023.
- [40] Juraj Kabzan, Boaz Cornelis Floor, and Srinivasan Sirish. Vehicle path planning, 2023. GB2602677A, Motional AD LLC. Also published as US, CN, KR, DE, and GB patents.
- [41] Juraj Kabzan, Hans Andersen, Lixun Lin, Ning Wu, Yiming Zhao, Xiyuan Liu, Qian Wang, Zachary Batts, Adam Jesse Miller, Boaz Cornelis Floor, and Marc Dominik Heim. Control architectures for autonomous vehicles, 2021. US20210276588A1, Motional AD LLC. Also published as KR, DE, and GB patents.
- [42] Alexander Schwab, Lisa-Marie Reichelt, Philipp Welz, and Jan Lunze. Experimental evaluation of an adaptive cruise control and cooperative merging concept. In *2020 IEEE Conference on Control Technology and Applications (CCTA)*, pages 318–325. IEEE, 2020.
- [43] Matthias Althoff, Markus Koschi, and Stefanie Manzing. Commonroad: Composable benchmarks for motion planning on roads. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 719–726. IEEE, 2017.
- [44] Miltos Kyriakidis, Riender Happee, and Joost CF De Winter. Public opinion on automated driving: Results of an international questionnaire among 5000 respondents. *Transportation research part F: traffic psychology and behaviour*, 32:127–140, 2015.

- [45] Mark A Geistfeld. A roadmap for autonomous vehicles: State tort liability, automobile insurance, and federal safety regulation. *Calif. L. Rev.*, 105:1611, 2017.
- [46] Philip Koopman and Michael Wagner. Autonomous vehicle safety: An interdisciplinary challenge. *IEEE Intelligent Transportation Systems Magazine*, 9(1):90–96, 2017.
- [47] Nidhi Kalra and Susan M Paddock. Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? *Transportation research part A: policy and practice*, 94:182–193, 2016.
- [48] Politecnico di Torino. Primo incontro operativo per l’infrastruttura infuturo, che realizzerà una pista prove per veicoli elettrici autonomi, 2023. Accessed: 2025-05-27.
- [49] ASAM e.V. ASAM OpenDRIVE® 1.7.0 Standard. <https://www.asam.net/standards/detail/opendrive/>, 2023. Accessed: 2025-03-24.
- [50] Eclipse Foundation. SUMO – Simulation of Urban MObility. <https://www.eclipse.dev/sumo/>, 2023. Accessed: 2025-03-04.
- [51] AVL List GmbH. AVL Testbed.CONNECT, 2023. Accessed: 2025-03-24.
- [52] AVL List GmbH. AVL PUMA – Automation for Testbeds, 2023. Accessed: 2025-03-24.
- [53] Giancarlo Genta. *Motor vehicle dynamics: modeling and simulation*, volume 43. World scientific, 1997.
- [54] TDK InvenSense. *MPU-9250 Nine-Axis (Gyro + Accelerometer + Compass) MEMS MotionTracking Device*. InvenSense, Inc., 2016. Accessed: 2025-03-25.
- [55] Vicon Motion Systems Ltd. *VICON Vero v2.2 Motion Capture Camera*. Vicon Motion Systems, 2023. Accessed: 2025-03-25.
- [56] Vicon Motion Systems Ltd. *Vicon Tracker Software*. Vicon Motion Systems, 2023. Accessed: 2025-03-25.
- [57] Giancarlo Genta and Cristiana Delprete. Some considerations on the experimental determination of moments of inertia. *Meccanica*, 29:125–141, 1994.
- [58] Alexander Liniger. *Path planning and control for autonomous racing*. ETH Zurich, 2018.
- [59] Andrea Carron, Sabrina Bodmer, Lukas Vogel, René Zurbrügg, David Helm, Rahel Rickenbach, Simon Muntwiler, Jerome Sieber, and Melanie N Zeilinger. Chronos and crs: Design of a miniature car-like robot and a software framework for single and multi-agent robotics and control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1371–1378. IEEE, 2023.

- [60] Jorge J Moré. The levenberg-marquardt algorithm: implementation and theory. In *Numerical analysis: proceedings of the biennial Conference held at Dundee, June 28–July 1, 1977*, pages 105–116. Springer, 2006.
- [61] Carmine Caponio, Pietro Stano, Raffaele Carli, Ignazio Olivieri, Daniele Ragone, Aldo Sorniotti, Patrick Gruber, and Umberto Montanaro. Modeling, positioning, and deep reinforcement learning path following control of scaled robotic vehicles: Design and experimental validation. *IEEE transactions on automation science and engineering*, 2024.
- [62] Torsten Bohlin and Stefan F Graebe. Issues in nonlinear stochastic grey box identification. *International journal of adaptive control and signal processing*, 9(6):465–490, 1995.
- [63] Efsthios Velenis, Emilio Frazzoli, and Panagiotis Tsotras. On steady-state cornering equilibria for wheeled vehicles with drift. In *Proceedings of the 48th IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 3545–3550. IEEE, 2009.
- [64] Christoph Voser, Rami Y Hindiyeh, and J Christian Gerdes. Analysis and control of high sideslip manoeuvres. *Vehicle System Dynamics*, 48(S1):317–336, 2010.
- [65] Egbert Bakker, Lars Nyborg, and Hans B Pacejka. Tyre modelling for use in vehicle dynamics studies. *SAE transactions*, pages 190–204, 1987.
- [66] Moritz Werling, Philip Reinisch, and Klaus Gresser. Kombinierte bremsausweich-assistenz mittels nichtlinearer modellprädiktiver trajektorienplanung für den aktiven fußgängerschutz. In *8. Workshop Fahrerassistenzsysteme, FAS2012: Walting im Altmühltal, 26.-28. September 2012*, pages 77–86, 2012.
- [67] dSPACE GmbH. *SCALEXIO - Modular Real-Time System for Hardware-in-the-Loop Simulation*. dSPACE GmbH, 2023. https://www.dspace.com/en/inc/home/products/hw/simulator_hardware/scalexio/scalexio_autobox.cfm.
- [68] Lukas P Fröhlich, Christian Küttel, Elena Arcari, Lukas Hewing, Melanie N Zeilinger, and Andrea Carron. Model learning and contextual controller tuning for autonomous racing. *arXiv preprint arXiv:2110.02710*, 2021.
- [69] Daniel Kloeser, Tobias Schoels, Tommaso Sartor, Andrea Zanelli, Gianluca Prison, and Moritz Diehl. Nmpc for racing using a singularity-free path-parametric model with obstacle avoidance. *IFAC-PapersOnLine*, 53(2):14324–14329, 2020.
- [70] Jason Kong, Mark Pfeiffer, Georg Schildbach, and Francesco Borrelli. Kinematic and dynamic vehicle models for autonomous driving control design. In *2015 IEEE intelligent vehicles symposium (IV)*, pages 1094–1099. IEEE, 2015.

- [71] Pietro Stano, Umberto Montanaro, Davide Tavernini, Manuela Tufo, Giovanni Fiengo, Luigi Novella, and Aldo Sorniotti. Model predictive path tracking control for automated road vehicles: A review. *Annual reviews in control*, 55:194–236, 2023.
- [72] Andres Alvarez, Nico Denner, Zhe Feng, David Fischer, Yang Gao, Lukas Harsch, Sebastian Herz, Nick Le Large, Bach Nguyen, Carlos Rosero, et al. The software stack that won the formula student driverless competition. *arXiv preprint arXiv:2210.10933*, 2022.
- [73] Dmitri Dolgov, Sebastian Thrun, Michael Montemerlo, and James Diebel. Path planning for autonomous vehicles in unknown semi-structured environments. *The international journal of robotics research*, 29(5):485–501, 2010.
- [74] David González, Joshué Pérez, Vicente Milanés, and Fawzi Nashashibi. A review of motion planning techniques for automated vehicles. *IEEE Transactions on intelligent transportation systems*, 17(4):1135–1145, 2015.
- [75] Christos Katrakazas, Mohammed Quddus, Wen-Hua Chen, and Lipika Deka. Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions. *Transportation Research Part C: Emerging Technologies*, 60:416–442, 2015.
- [76] Moritz Werling, Julius Ziegler, Sören Kammel, and Sebastian Thrun. Optimal trajectory generation for dynamic street scenarios in a frenet frame. In *2010 IEEE international conference on robotics and automation*, pages 987–993. IEEE, 2010.
- [77] Julius Ziegler, Philipp Bender, Thao Dang, and Christoph Stiller. Trajectory planning for bertha—a local, continuous method. In *2014 IEEE intelligent vehicles symposium proceedings*, pages 450–457. IEEE, 2014.
- [78] Brian Paden, Michal Čáp, Sze Zheng Yong, Dmitry Yershov, and Emilio Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on intelligent vehicles*, 1(1):33–55, 2016.
- [79] International Organization for Standardization (ISO). Iso 3888-2:2011. passenger cars – test track for a severe lane-change manoeuvre – part 2: Obstacle avoidance, March 2011. Prepared by ISO/TC 22/SC 33; confirmed in 2016; to be revised.
- [80] Alexander Winkler, Jonathan Frey, Timm Fahrbach, Gianluca Frison, René Scheer, Moritz Diehl, and Jakob Andert. Embedded real-time nonlinear model predictive control for the thermal torque derating of an electric vehicle. *IFAC-PapersOnLine*, 54(6):359–364, 2021.
- [81] Gianluca Frison, Dimitris Kouzoupis, Tommaso Sartor, Andrea Zanelli, and Moritz Diehl. Blasfeo: Basic linear algebra subroutines for embedded optimization. *ACM Transactions on Mathematical Software (TOMS)*, 44(4):1–30, 2018.

- [82] Joel AE Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. Casadi: a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11:1–36, 2019.
- [83] Boris Houska, Hans Joachim Ferreau, and Moritz Diehl. Acado toolkit—an open-source framework for automatic control and dynamic optimization. *Optimal control applications and methods*, 32(3):298–312, 2011.
- [84] Stephen Wolfram. *The mathematica book*. Wolfram Research, Inc., 2003.
- [85] Aaron Meurer, Christopher P Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K Moore, Sartaj Singh, et al. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103, 2017.
- [86] Symbolic Math Toolbox et al. Matlab. *Mathworks Inc*, page 20, 1993.
- [87] Sébastien Gros, Mario Zanon, Rien Quirynen, Alberto Bemporad, and Moritz Diehl. From linear to nonlinear mpc: bridging the gap via the real-time iteration. *International Journal of Control*, 93(1):62–80, 2020.
- [88] Gabriel M Hoffmann, Claire J Tomlin, Michael Montemerlo, and Sebastian Thrun. Autonomous automobile trajectory tracking for off-road driving: Controller design, experimental validation and racing. In *2007 American control conference*, pages 2296–2301. IEEE, 2007.
- [89] Martin Buehler, Karl Iagnemma, and Sanjiv Singh. *The DARPA urban challenge: autonomous vehicles in city traffic*, volume 56. Springer Science & Business Media, 2009.
- [90] Dong Hun Shin. *High-performance tracking of explicit paths by roadworthy mobile robots*. Carnegie Mellon University, 1990.
- [91] Eugenio Tramacere, Sara Luciani, Stefano Feraco, Angelo Bonfitto, and Nicola Amati. Processor-in-the-loop architecture design and experimental validation for an autonomous racing vehicle. *Applied Sciences*, 11(16):7225, 2021.
- [92] Sabrina Bodmer, Lukas Vogel, Simon Muntwiler, Alexander Hansson, Tobias Bodewig, Jonas Wahlen, Melanie N Zeilinger, and Andrea Carron. Optimization-based system identification and moving horizon estimation using low-cost sensors for a miniature car-like robot. *arXiv preprint arXiv:2404.08362*, 2024.

Appendix A

Survey of small testbed

The following is a list of small-scale testbed infrastructures for CAVs/RSs research found in the available literature. Fifteen small testbeds have been identified as relevant. The specific characteristics of each testbed have been divided into 4 categories: general, sense, plan, and act characteristics.

Table A.1 CARS Autonomous Scaled-Vehicle Testbed: General, Sense and Plan characteristics

General Characteristics	
Testbeds	CARS Autonomous Scaled-Vehicle Testbed
General information	
Main references	
Year of the first publication introducing the test bed	2025
Focus	Multi-agent decision-making trajectory planning and control for Connected and Automated Vehicles (CAVs)
CAV or RS	CAV
Software Information	Based on MATLAB/SIMULINK
Programming language	Matlab
Scenario formats	Customizable
Sense Characteristics	
Sensors	Ceiling mounted Vicon camera WROOM ESP32 and IMU sensor onboard vehicle
Global positioning system	Indoor Positioning System (IPS) using ceiling camera
Accuracy	< 1 mm
Traffic management infrastructure	No
Surroundings	Racing Track, cities, moving obstacles
Plan Characteristics	
Lab infrastructure	A combination of high-level controller on the computation units, and low-level controller on the vehicle.
Computation unit/ Computation scheme	Centralized computer for vision system Real-time electronic control unit for vehicles motion planning. Ground Station sends throttle, brake and steering commands through CAN over WIFI communication
Human-Robot-Interaction (Mixed traffic support)	Yes
Different kind of agents	No
Agent count	1+

Table A.2 CARS Autonomous Scaled-Vehicle Testbed: Act characteristics

Act characteristics	
Longitudinal dynamics	
Actuator hardware	Brushless DC motor Electronic speed controller (ESC)
Battery	Gens Ace 3500 mAh 7.4 V 2S LiPo Battery pack
Battery-to-agent weight ratio	<24%
Speed min [m/s]	0
Speed max [m/s]	17.8 (limited to 3 by software)
Lateral dynamics	
Steering	Ackermann steering
Steering angle max [deg]	34
Geometry	
Wheelbase [m]	0.260
Length [m]	0.416
Width [m]	0.190
Height [m]	0.146
Weight [kg]	1.6

Table A.3 CPM Lab Testbed: General, Sense and Plan characteristics

General Characteristics	
Testbeds	CPM Lab
General information	https://cpm.embedded.rwth-aachen.de/
Main references	https://doi.org/10.23919/ECC54610.2021.9654986 https://doi.org/10.1109/CoDIT55151.2022.9804088
Year of the first publication introducing the test bed	2021
Focus	Multi-agent decision-making trajectory planning and control for Connected and Automated Vehicles (CAVs)
CAV or RS	CAV
Software Information	Based on Data Distribution Service (DDS)
Programming language	C++ Matlab
Scenario formats	Lanlet1, Lanlet2
Sense Characteristics	
Sensors	Ceiling mounted Basler camera Sensitive surface layer
Global positioning system	Indoor Positioning System (IPS) using ceiling camera
Accuracy	1 - 3 cm
Traffic management infrastructure	No
Surroundings	
Plan Characteristics	
Lab infrastructure	A combination of high-level controller on the computation units, and mid-level and low-level controller on the vehicle. Collaborating via a middleware on both sides using Data Distribution Service (DDS) via WiFi
Computation unit/ Computation scheme	Each vehicle is assigned an Intel NUC for high-level computation/ Parallel, sequential and hybrid
Human-Robot-Interaction (Mixed traffic support)	Yes
Different kind of agents	No
Agent count	20

Table A.4 CPM Lab Testbed: Act characteristics

Act Characteristics	
Testbeds	CPM Lab
Longitudinal dynamics	
Actuator hardware	Brushless DC motor Electronic speed controller (ESC)
Battery	Gens Ace 3500 mAh 7.4 V 2S LiPo Battery pack
Battery-to-agent weight ratio	34%
Speed min [m/s]	0
Speed max [m/s]	3.7 (limited to 2.7 by software)
Lateral dynamics	
Steering	Ackermann steering
Steering angle max [deg]	35
Geometry	
Wheelbase [m]	0.150
Length [m]	0.220
Width [m]	0.107
Height [m]	0.070
Weight [kg]	0.5

Table A.5 F1TENTH: General, Sense and Plan characteristics

General Characteristics	
Testbeds	F1TENTH
General information	https://f1tenth.org/
Main references	https://doi.org/10.48550/arXiv.1901.08567 https://par.nsf.gov/biblio/10221872 https://core.ac.uk/download/pdf/84832234.pdf https://doi.org/10.1109/CASE48305.2020.9216949
Year of the first publication introducing the test bed	2020
Focus	Autonomous racing, Computer Vision (CV)
CAV or RS	CAV
Software Information	Based on ROS
Programming language	C++ Python
Scenario formats	Customizable
Sense Characteristics	
Sensors	None
Global positioning system	With onboard sensors
Accuracy	A few mm
Traffic management infrastructure	No
Surroundings	Racing Track, moving obstacles
Plan Characteristics	
Lab infrastructure	ROS-over-network WiFi can be supported
Computation unit/ Computation scheme	Jetson TX2 GPU computer/ Onboard computation
Human-Robot-Interaction (Mixed traffic support)	No
Different kind of agents	No
Agent count	1+

Table A.6 F1TENTH: Act characteristics

Act characteristics	
Longitudinal dynamics	
Actuator hardware	Brushless DC motor Electronic speed controller (ESC)
Battery	Traxxas LiPo or NiMH battery (5000 mAh)
Battery-to-agent weight ratio	<24%
Speed min [m/s]	0
Speed max [m/s]	17.8
Lateral dynamics	
Steering	Ackermann steering
Steering angle max [deg]	45
Geometry	
Wheelbase [m]	No information found
Length [m]	No information found
Width [m]	No information found
Height [m]	No information found
Weight [kg]	> 3

Table A.7 Robotarium: General, Sense and Plan characteristics

General Characteristics	
Testbeds	Robotarium
General information	https://www.robotarium.gatech.edu/
Main references	https://ieeexplore.ieee.org/document/7989200 https://ieeexplore.ieee.org/abstract/document/8960572 https://doi.org/10.1109/LRA.2021.3062796
Year of the first publication introducing the test bed	2017
Focus	Autonomous racing, Computer Vision (CV)
CAV or RS	RS
Software Information	Consists of components divided between MATLAB/Python, simulation API, Custom MQTT pub/sub interaction with testbed, and Docker containerized applications coordinating server applications.
Programming language	Python Matlab
Scenario formats	No information found
Sense Characteristics	
Sensors	Vicon Camera
Global positioning system	Vicon system
Accuracy	< 1 mm
Traffic management infrastructure	-
Surroundings	None
Plan Characteristics	
Lab infrastructure	Tracking through overhead camera and communicating via WiFi
Computation unit/ Computation scheme	Central server/ User code executed by a central server Onboard distributed computation for velocity control
Human-Robot-Interaction (Mixed traffic support)	No
Different kind of agents	Yes (different robots with similar specifications)
Agent count	20

Table A.8 Robotarium: Act characteristics

Act characteristics	
Longitudinal dynamics	
Actuator hardware	100:1 Micro Metal Gearmotor LP 6V with Extended Motor Shaft
Battery	Lithium Ion Polymer Battery - 3.7v 2500mAh
Battery-to-agent weight ratio	No information found
Speed min [m/s]	0.03
Speed max [m/s]	0.2
Lateral dynamics	
Steering	Differential drive
Steering angle max [deg]	No information found
Geometry	
Wheelbase [m]	0.110
Length [m]	0.095
Width [m]	0.110
Height [m]	0.095
Weight [kg]	No information found

Table A.9 IDS3C(UDSSC): General, Sense and Plan characteristics

General Characteristics	
Testbeds	IDS3C(UDSSC)
General information	https://ids-lab.net/facilities/
Main references	https://doi.org/10.1016/j.ifacol.2018.07.022 https://doi.org/10.1109/MCS.2022.3209056
Year of the first publication introducing the test bed	2018
Focus	Provide a remotely accessible platform to study Robot Swarms (RS) for research and education
CAV or RS	CAV
Software Information	Central computer: ROS Vehicle: Ros/Arduino
Programming language	Mainframe C++ Vehicle ROS/C++/Python
Scenario formats	(4way/T-shape) intersections Merging and roundabout
Sense Characteristics	
Sensors	Vicon Camera Raspberry Pi camera
Global positioning system	Vicon system
Accuracy	< 1 mm
Traffic management infrastructure	Users can place (stop/yield) signs
Surroundings	Buildings
Plan Characteristics	
Lab infrastructure	ROS/WiFi
Computation unit/ Computation scheme	Centralized computer Vehicle: (Raspberry Pi & ATmega32U)/ Parallel, sequential and hybrid
Human-Robot-Interaction (Mixed traffic support)	Yes
Different kind of agents	Yes (CAVs and drones)
Agent count	~20

Table A.10 IDS3C(UDSSC): Act characteristics

Act characteristics	
Longitudinal dynamics	
Actuator hardware	One geared DC motor
Battery	1000mAh LiPo Battery (7.4VDC)
Battery-to-agent weight ratio	No information found
Speed min [m/s]	0.2
Speed max [m/s]	0.5
Lateral dynamics	
Steering	Ackermann steering
Steering angle max [deg]	45
Geometry	
Wheelbase [m]	No information found
Length [m]	No information found
Width [m]	No information found
Height [m]	No information found
Weight [kg]	No information found

Table A.11 Cambridge Minicar: General, Sense and Plan characteristics

General Characteristics	
Testbeds	Cambridge Minicar
General information	https://proroklab.org/wp/
Main references	https://doi.org/10.1109/ICRA.2019.8794445
Year of the first publication introducing the test bed	2019
Focus	Multi-agent decision-making for CAVs
CAV or RS	CAV
Software Information	No information found
Programming language	Python
Scenario formats	No information found
Sense Characteristics	
Sensors	OptiTrack cameras
Global positioning system	OptiTrack motion capture system
Accuracy	0.2 mm
Traffic management infrastructure	-
Surroundings	None
Plan Characteristics	
Lab infrastructure	Radio
Computation unit/ Computation scheme	External computation via workstation/ Mixed (offboard and onboard)
Human-Robot-Interaction (Mixed traffic support)	Partial
Different kind of agents	No
Agent count	~ 20

Table A.12 Cambridge Minicar: Act characteristics

Act characteristics	
Longitudinal dynamics	
Actuator hardware	No information found
Battery	Three AA Batteries
Battery-to-agent weight ratio	20%
Speed min [m/s]	No information found
Speed max [m/s]	1.5
Lateral dynamics	
Steering	Ackermann steering
Steering angle max [deg]	18
Geometry	
Wheelbase [m]	0.122
Length [m]	0.197
Width [m]	0.081
Height [m]	0.075
Weight [kg]	0.450

Table A.13 Duckietown: General, Sense and Plan characteristics

General Characteristics	
Testbeds	Duckietown
General information	https://www.duckietown.org/
Main references	https://ieeexplore.ieee.org/abstract/document/7989179 https://doi.org/10.1007/978-3-319-55553-9_8
Year of the first publication introducing the test bed	2017
Focus	Perception, navigation and collision/obstacle avoidance
CAV or RS	CAV
Software Information	A containerized architecture, where every container is responsible for a part of the main functionalities.
Programming language	Python ROS
Scenario formats	Scenarios are customized with tiles
Sense Characteristics	
Sensors	Watchtowers with Raspberry Pi cameras
Global positioning system	OptiTrack motion capture system
Accuracy	Watchtower: 1 cm Sign tracking: Depends on the users algorithm
Traffic management infrastructure	Users can place signs and traffic lights
Surroundings	Buildings, obstacles, traffic signs and lights
Plan Characteristics	
Lab infrastructure	Vehicles can communicate with LEDs which are detected by other vehicles
Computation unit/ Computation scheme	Raspberry Pi 2 B on each vehicle/ Both onboard and offboard are supported through the api, parallel and sequential computation is achieved also.
Human-Robot-Interaction (Mixed traffic support)	No
Different kind of agents	Yes (CAVs and drones)
Agent count	Depends on the user setup

Table A.14 Duckietown: Act characteristics

Act characteristics	
Longitudinal dynamics	
Actuator hardware	2DC motors
Battery	10 Ah at 3.7V (37Wh)
Battery-to-agent weight ratio	10.4%
Speed min [m/s]	0
Speed max [m/s]	No information found
Lateral dynamics	
Steering	Differential drive
Steering angle max [deg]	90
Geometry	
Wheelbase [m]	0.100
Length [m]	0.340
Width [m]	0.150
Height [m]	0.230
Weight [kg]	1.8

Table A.15 CHARTOPOLIS: General, Sense and Plan characteristics

General Characteristics	
Testbeds	CHARTOPOLIS
General information	https://doi.org/10.48550/arXiv.2210.00377
Main references	https://doi.org/10.48550/arXiv.2210.00377
Year of the first publication introducing the test bed	2022
Focus	Human-robot interaction, art installation
CAV or RS	RS
Software Information	ROS, DonkeyCar stack
Programming language	Python C++
Scenario formats	ASAM OpenDrive
Sense Characteristics	
Sensors	Photo-interrupt encoders on drive shaft Inertial Measurement Unit (IMU)
Global positioning system	Overhead camera OptiTrack
Accuracy	No information found
Traffic management infrastructure	No information found
Surroundings	Buildings, traffic lights and signs Adjustable lighting
Plan Characteristics	
Lab infrastructure	Dual-component laboratory architecture that includes a reconfigurable physical driving environment with agents and traffic elements, complemented by a high-fidelity virtual simulator
Computation unit/ Computation scheme	Computation on each vehicle/ No information found
Human-Robot-Interaction (Mixed traffic support)	Yes
Different kind of agents	No
Agent count	No information found

Table A.16 CHARTOPOLIS: Act characteristics

Act characteristics	
Longitudinal dynamics	
Actuator hardware	RC380 high speed carbon brushed motor
Battery	8.4 V, 18650 battery $\times$ 4 (two in parallel, two in series)
Battery-to-agent weight ratio	6.9%
Speed min [m/s]	No information found
Speed max [m/s]	No information found
Lateral dynamics	
Steering	No information found
Steering angle max [deg]	No information found
Geometry	
Wheelbase [m]	0.178
Length [m]	No information found
Width [m]	No information found
Height [m]	0.133
Weight [kg]	$\sim$ 2.6

Table A.17 Cambridge RoboMaster: General, Sense and Plan characteristics

General Characteristics	
Testbeds	Cambridge RoboMaster
General information	https://arxiv.org/pdf/2405.02198
Main references	http://arxiv.org/abs/2405.02198
Year of the first publication introducing the test bed	2024
Focus	An enhanced and cost-effective multi-robot research platform based on the DJI RoboMaster S1, designed to enable versatile, reliable, and agile multi-agent experiments through integrated hardware, control systems, and simulation tools
CAV or RS	CAV
Software Information	https://proroklab.github.io/cambridge-robomaster/software_setup.html
Programming language	Python C++
Scenario formats	No information found
Sense Characteristics	
Sensors	Raspberry Pi HQ camera Physical bumper sensors Wheel odometry
Global positioning system	Motion Capture System
Accuracy	No information found
Traffic management infrastructure	-
Surroundings	Configurable static obstacles
Plan Characteristics	
Lab infrastructure	Modular ROS2-based optimal estimation and control stack for full onboard autonomy, with ad-hoc peer-to-peer communication infrastructure
Computation unit/ Computation scheme	Nvidia Jetson Orin NX 16GB (Raspberry Pi also available)/ On board computation
Human-Robot-Interaction (Mixed traffic support)	No
Different kind of agents	No
Agent count	~ 9

Table A.18 Cambridge RoboMaster: Act characteristics

Act characteristics	
Longitudinal dynamics	
Actuator hardware	Four DJI 3508I BLDC motors with integrated gear and controller
Battery	25.91Wh, 2.4 Ah
Battery-to-agent weight ratio	No information found
Speed min [m/s]	0
Speed max [m/s]	4.5
Lateral dynamics	
Steering	Four-wheel mecanum (omnidirectional)
Steering angle max [deg]	360
Geometry	
Wheelbase [m]	No information found
Length [m]	0.320
Width [m]	0.240
Height [m]	0.200
Weight [kg]	3

Table A.19 Go-Chart: General, Sense and Plan characteristics

General Characteristics	
Testbeds	Go-Chart
General information	https://doi.org/10.1109/IROS45743.2020.9341770
Main references	https://doi.org/10.1109/IROS45743.2020.9341770
Year of the first publication introducing the test bed	2020
Focus	Testing interaction with HDVs, CV
CAV or RS	CAV
Software Information	External GPU for video and image processing and onboard computation for vehicle control
Programming language	No information found
Scenario formats	No information found
Sense Characteristics	
Sensors	2D LiDAR Camera Ultrasonic sensor IMU
Global positioning system	2D LiDAR Wide angle camera IMU
Accuracy	No information found
Traffic management infrastructure	Yes
Surroundings	Buildings, traffic lights, LEDs in the car
Plan Characteristics	
Lab infrastructure	No communication
Computation unit/ Computation scheme	Video streaming: 1.2GHz ARM Cortex-A53 with 1GB LPDDR2 Vehicle Control: Teensy 3.2 (ARM Cortex-M4 core running at 96MHz with 64kB RAM)/ Onboard computation with external GPU for deep learning
Human-Robot-Interaction (Mixed traffic support)	Yes
Different kind of agents	No
Agent count	No information found

Table A.20 Go-Chart: Act characteristics

Act characteristics	
Longitudinal dynamics	
Actuator hardware	Powered by four standard micro metal gear motors
Battery	Four 18650 batteries (7.4V at 7000 mAh)
Battery-to-agent weight ratio	No information found
Speed min [m/s]	0.04
Speed max [m/s]	0.42
Lateral dynamics	
Steering	Four-wheel skid steering system
Steering angle max [deg]	No information found
Geometry	
Wheelbase [m]	No information found
Length [m]	0.170
Width [m]	0.110
Height [m]	0.150
Weight [kg]	No information found

Table A.21 CRS: General, Sense and Plan characteristics

General Characteristics	
Testbeds	CRS
General information	https://gitlab.ethz.ch/ics/crs
Main references	https://doi.org/10.1109/ICRA48891.2023.10161434
Year of the first publication introducing the test bed	2023
Focus	Autonomous racing, multi-agent decision-making, learning-based control
CAV or RS	CAV
Software Information	ROS, but packages are ROS independent
Programming language	C++ Python ROS
Scenario formats	Customizable
Sense Characteristics	
Sensors	Steer encoder Motion Capture Wheel encoders Voltage and current measurements IMU
Global positioning system	Motion capture (qualysis)
Accuracy	< 1 mm
Traffic management infrastructure	No
Surroundings	Racing track, customizable mats
Plan Characteristics	
Lab infrastructure	Low-level controller for steer and torque control on the vehicle. Sensor measurements and control commands are exchanged with a computer via WiFi. Controller, planner, estimator, and safety filter are run on a laptop using a custom C++ (open source) software where the communication among packages is managed by ROS.
Computation unit/ Computation scheme	Each vehicle has a ESP32 onboard. External computation can be done by an arbitrary number of laptops (currently only 1 is used)/ No information found
Human-Robot-Interaction (Mixed traffic support)	Yes
Different kind of agents	Yes
Agent count	~ 10

Table A.22 CRS: Act characteristics

Act characteristics	
Longitudinal dynamics	
Actuator hardware	Brush motor with gearbox
Battery	4AAA batteries
Battery-to-agent weight ratio	No information found
Speed min [m/s]	-4 (limited by the controller)
Speed max [m/s]	4 (limited by the controller)
Lateral dynamics	
Steering	Ackermann steering
Steering angle max [deg]	35 - 42 (depends on the hardware)
Geometry	
Wheelbase [m]	0.98 (sizes can depend on the vehicle model)
Length [m]	0.166
Width [m]	0.075
Height [m]	0.044
Weight [kg]	0.180

Table A.23 MiniCity: General, Sense and Plan characteristics

General Characteristics	
Testbeds	MiniCity
General information	https://doi.org/10.3390/s22186793
Main references	https://doi.org/10.3390/s22186793 https://motion-planning-workshop.kavrakilab.org/papers/minicity.pdf
Year of the first publication introducing the test bed	2022
Focus	Testing perception hardware and software for autonomous vehicles
CAV or RS	CAV
Software Information	Uses ROS Melodic for interprocess communication
Programming language	ROS
Scenario formats	Customizable
Sense Characteristics	
Sensors	OptiTrack cameras
Global positioning system	OptiTrack motion capture system
Accuracy	< 2 mm
Traffic management infrastructure	Yes
Surroundings	Buildings, traffic signs and lights
Plan Characteristics	
Lab infrastructure	ROS V2X
Computation unit/ Computation scheme	Onboard computation/ Onboard computation
Human-Robot-Interaction (Mixed traffic support)	Yes
Different kind of agents	No
Agent count	No information found

Table A.24 MiniCity: Act characteristics

Act characteristics	
Longitudinal dynamics	
Actuator hardware	No information found
Battery	No information found
Battery-to-agent weight ratio	No information found
Speed min [m/s]	No information found
Speed max [m/s]	No information found
Lateral dynamics	
Steering	Ackermann steering
Steering angle max [deg]	No information found
Geometry	
Wheelbase [m]	No information found
Length [m]	0.530
Width [m]	0.300
Height [m]	0.330
Weight [kg]	5.4

Table A.25 Optimal RC racing (ORCA): General, Sense and Plan characteristics

General Characteristics	
Testbeds	Optimal RC racing (ORCA)
General information	https://control.ee.ethz.ch/research/team-projects/autonomous-rc-car-racing.html
Main references	https://doi.org/10.1002/oca.2123
Year of the first publication introducing the test bed	2014
Focus	Autonomous racing, research and education in automatic control
CAV or RS	CAV
Software Information	The positioning system communicates via USB with a computer. Relevant information gets then distributed via Bluetooth
Programming language	C++ Matlab
Scenario formats	No information found
Sense Characteristics	
Sensors	IMU Gyroscope Accelerometer
Global positioning system	PointGrey Flea3 camera Infrared spot Wide angle lens with infrared filter
Accuracy	< 4 mm
Traffic management infrastructure	-
Surroundings	Walls, stationary cars
Plan Characteristics	
Lab infrastructure	Ethernet for vision data From computer to vehicle via Bluetooth
Computation unit/ Computation scheme	ARM A9 chip Exynos 4412 or ARM Cortex A15 Exynos 5410 chip/ No information found
Human-Robot-Interaction (Mixed traffic support)	No
Different kind of agents	No
Agent count	1+

Table A.26 Optimal RC racing (ORCA): Act characteristics

Act characteristics	
Longitudinal dynamics	
Actuator hardware	Depends on chosen model car
Battery	3.7V 130mAh LiPO battery
Battery-to-agent weight ratio	No information found
Speed min [m/s]	Depends on the chosen model car
Speed max [m/s]	> 3
Lateral dynamics	
Steering	Ackermann steering
Steering angle max [deg]	Depends on the chosen model car
Geometry	
Wheelbase [m]	Depends on the chosen model car
Length [m]	Depends on the chosen model car
Width [m]	Depends on the chosen model car
Height [m]	Depends on the chosen model car
Weight [kg]	Depends on the chosen model car

Table A.27 Miniature Autonomy: General, Sense and Plan characteristics

General Characteristics	
Testbeds	Miniature Autonomy
General information	http://dx.doi.org/10.3389/fnbot.2022.846355
Main references	https://doi.org/10.5220%2F0007955704830488 http://dx.doi.org/10.3389/fnbot.2022.846355
Year of the first publication introducing the test bed	2019
Focus	Testing of autonomous vehicles and aerial vehicles, CV
CAV or RS	CAV
Software Information	All computations on vehicle (ROS Stack) TensorFlow Lite on machine learning tasks
Programming language	No information found
Scenario formats	Customizable
Sense Characteristics	
Sensors	One front facing Ultra-wide-angle camera Absolute encoder UWB Indoor Positioning System (IPS)
Global positioning system	Top view camera with markers on vehicles Ultra wide band radio with onboard module
Accuracy	No information found
Traffic management infrastructure	Yes
Surroundings	Buildings, intersections, traffic lights
Plan Characteristics	
Lab infrastructure	ROS UWB Radio
Computation unit/ Computation scheme	None/ No information found
Human-Robot-Interaction (Mixed traffic support)	No information found
Different kind of agents	No
Agent count	No information found

Table A.28 Miniature Autonomy: Act characteristics

Act characteristics	
Longitudinal dynamics	
Actuator hardware	No information found
Battery	3500 mAh Li-Ion battery
Battery-to-agent weight ratio	No information found
Speed min [m/s]	No information found
Speed max [m/s]	No information found
Lateral dynamics	
Steering	Ackermann steering
Steering angle max [deg]	No information found
Geometry	
Wheelbase [m]	No information found
Length [m]	No information found
Width [m]	No information found
Height [m]	No information found
Weight [kg]	No information found

Table A.29 Self-Driving Car Studio: General, Sense and Plan characteristics

General Characteristics	
Testbeds	Self-Driving Car Studio at University of Surrey
General information	https://www.surrey.ac.uk/centre-automotive-engineering/research/connected-and-autonomous-vehicles-lab https://www.quanser.com/products/self-driving-car-studio/ https://arxiv.org/pdf/2401.05194v1
Main references	
Year of the first publication introducing the test bed	2024
Focus	Multi-agent decision-making trajectory planning and control for Connected and Automated Vehicles (CAVs)
CAV or RS	CAV
Software Information	Central computer: ROS
Programming language	Python Matlab C/C++
Scenario formats	Customizable
Sense Characteristics	
Sensors	Onboard vehicle: LiDAR, camera, encoder, IMU
Global positioning system	With onboard sensors
Accuracy	A few mm
Traffic management infrastructure	Yes
Surroundings	City with moving obstacles
Plan Characteristics	
Lab infrastructure	A combination of high-level controller on the computation units, and mid-level and low-level controller on the vehicle. Collaborating via a middleware on both sides using Data Distribution Service (DDS) via WiFi
Computation unit/ Computation scheme	Each vehicle is powered by the NVIDIA Jetson Orin AGX running Ubuntu and supporting the latest Jetpack
Human-Robot-Interaction (Mixed traffic support)	No
Different kind of agents	No
Agent count	1+

Table A.30 Self-Driving Car Studio: Act characteristics

Act characteristics	
Longitudinal dynamics	
Actuator hardware	Brushless DC motor Electronic speed controller (ESC)
Battery	3S 11.1 V LiPo (3300 mAh) with XT60 connector
Battery-to-agent weight ratio	24%
Speed min [m/s]	0
Speed max [m/s]	2
Lateral dynamics	
Steering	Ackermann steering
Steering angle max [deg]	35
Geometry	
Wheelbase [m]	0.280
Length [m]	0.390
Width [m]	0.200
Height [m]	0.290
Weight [kg]	2.7