Safe Reinforcement Learning for Automated Vehicles

R. Cornet - 4302087

Master of Science Thesis

Delft University of Technology

Department of Cognitive Robotics - Vehicle Engineering

Safe Reinforcement Learning for Automated Vehicles

Master of Science Thesis

For the degree of Master of Science in Vehicle Engineering at Delft University of Technology

R. Cornet - 4302087

August 14, 2020

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Abstract

Fully automated vehicles have the potential to increase road safety and improve traffic flow by taking the human element out of the driving loop. They can also provide mobility to people who are unable to operate a conventional vehicle. Safe automated vehicles must be able to respond in emergency situations or drive on slippery roads in bad weather conditions. Therefore it is crucial to have a safe and robust control strategy that can use the full handling capabilities of the vehicle.

This thesis presents how safe reinforcement learning can be used to design a steering policy that can drive an automated vehicle at the limit of friction.

The steering policies are trained using the Lyapunov Safe Actor-Critic (LSAC) algorithm. LSAC is a combination of the Soft Actor-Critic (SAC) algorithm and a Lyapunov stability analysis to solve constrained control problems.

The performance of LSAC is tested in a vehicle simulator against SAC and Model Predictive Control (MPC) in a series of tests that include changing lanes at different speeds, recovering from a destabilizing collision, and driving on a race track at the limit of friction.

The experiments show that LSAC outperforms MPC and SAC control strategies in terms of safety and vehicle stability. LSAC can recover from larger disturbances than MPC and SAC. A control strategy is presented that will keep the vehicle stable when driving at the limit of friction but can use the maneuverability of an unstable vehicle when is it necessary to avoid dangerous situations. Additionally, a policy is presented that can find the fastest way around a race track while staying within the track limits.

Table of Contents

List of Figures

List of Tables

Preface

Before you lies my thesis as part of my Master of Science graduation. The process that led me to write this thesis greatly widened my interests in scientific fields outside my own, in particular Reinforcement Learning.

I would like to thank my supervisor, Dr. Wei Pan, for his assistance over the last year. He introduced me to topics I was unfamiliar with and motivated me to master new skills. Together we found a research direction that combined his expertise in control theory with my background in mechanical and vehicle engineering.

I also want to thank Minghao Han of the Harbin Institute of Technology who kindly shared his code of the learning algorithms with us during his visit to Dr. Pan's lab. This allowed me to experiment with state-of-the-art learning algorithms.

I highly value everything I have learned in this time.

I would also like to thank my fellow students and the good friends I made along the way. We made great memories together and I will remember this time fondly.

I hope you enjoy reading my thesis.

Robert Cornet

Delft, University of Technology
August 14, 2020

“The world is not really stochastic. If you’ve noticed – a robot doesn’t just spontaneously start shaking. Unless you hook up an RL algorithm to it.”

— *Emanuel Todorov*

Chapter 1

Introduction

1-1 Motivation

1-1-1 Automated driving

Advanced Driver Assistance Systems (ADAS) are defined as "vehicle-based intelligent safety systems which could improve road safety in terms of crash avoidance, crash severity mitigation and protection and post-crash phases" [1]. Well known examples of ADAS are features like Adaptive Cruise Control, Blind Spot Monitoring, and Electronic Stability Control. ADAS are driving aids that assist the driver in operating the vehicle safely but can not drive the vehicle autonomously.

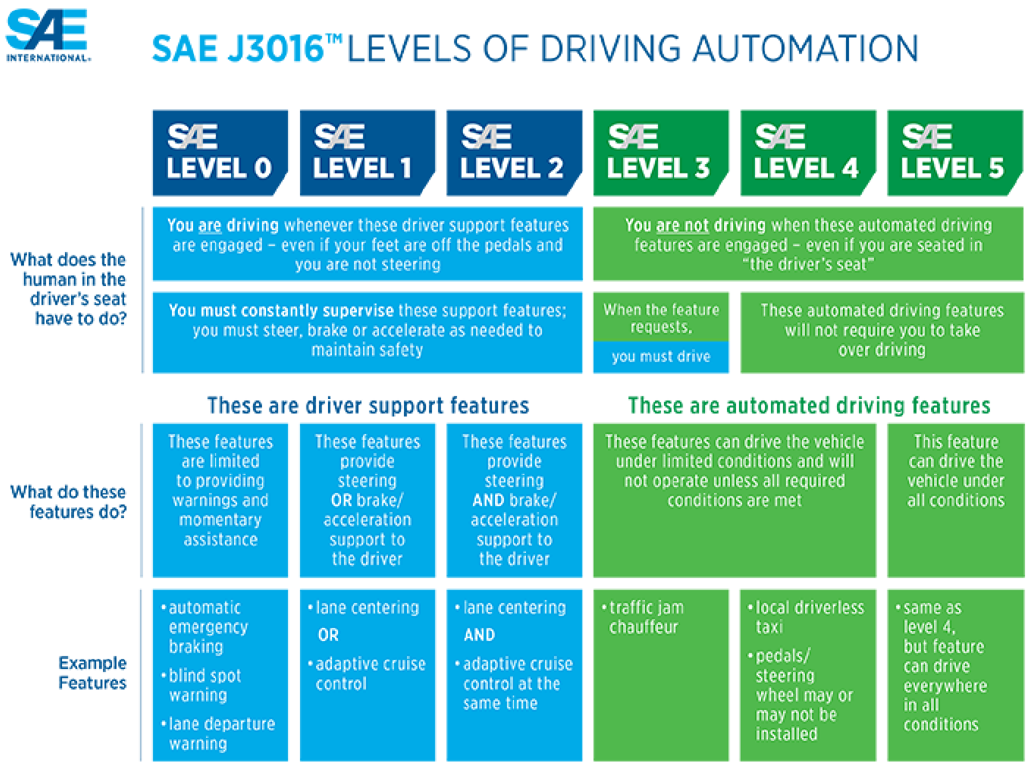
Higher levels of automation that can drive the vehicle autonomously are known as Automated Driving Systems (ADS). ADS are classified in 6 different SAE levels [2], see Figure 1-1. For SAE levels 0-2 the driver is expected to continuously supervise the car and must be ready to take over control when requested. SAE levels 3-5 are systems that can -depending on the environment- drive autonomously without supervision. Arguably the best-known ADS is Tesla's Autopilot [3], which is considered a level 2 ADS by the SAE definitions.

The application of automation and specifically high-level ADS in vehicles can have many benefits. Taking the human element out of the driving loop is expected to result in fewer (fatal) accidents and better traffic flow in congested traffic. It can also provide mobility and freedom to those who are unable to drive manually operated vehicles, and people could spend the time traveling in cars more productively [4, 5].

1-2 Background

1-2-1 Challenges of automated driving

There are still many challenges to be solved towards level 5 automated driving [6]. There are ethical and moral dilemmas to consider [7] as well as technical challenges to overcome.



SAE J3016™ LEVELS OF DRIVING AUTOMATION

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

Figure 1-1: The SAE levels of automated driving [2].

One of the technical challenges is the control problem. A robust control strategy is required that chooses the best steering, throttle, and braking actions based on an observation of the environment under all conditions. This means correctly identifying and anticipating the different cars, bicyclists, and pedestrians, as well as driving through bad weather and on slippery roads.

1-2-2 Driving near the friction limit

Steering the vehicle at low speeds such as parking or urban driving is not too difficult because the vehicle will respond nearly linear to the steering inputs. This is true for most normal driving situations where the vehicle is operating well within the limits of the maximum tire friction.

However, for safe operation under all circumstances, the vehicle must also be able to operate near the friction limit. The friction limit is the maximum amount of force that the tires can generate and is mainly determined by the friction between the tires and the road. Near the friction limit the response of the vehicle can be very non-linear, making it difficult to control.

Driving near the friction limit is often associated with racing, but it is an important aspect of safe driving as well. In an emergency stop, the Anti-lock Braking System (ABS) keeps the tires near the friction limit to generate as much braking force as possible. The friction limit is also easily reached on low friction (slippery) surfaces such as wet roads and ice, or loose surfaces like sand and snow. The maximum tire forces for different surfaces are illustrated in

Figure 1-2. Therefore it is important to have a control strategy that can operate the vehicle near the friction limit to keep the vehicle safe.

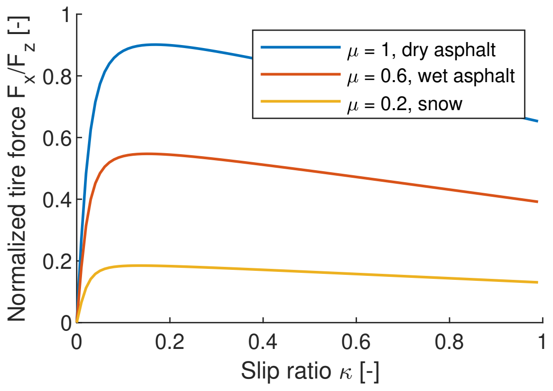


Figure 1-2: Normalized tire forces (F_x/F_z) as a function of slip ratio κ for driving on different surfaces with tire-road friction coefficient μ.

1-2-3 Vehicle stability and maneuverability

The vehicle can become unstable and difficult to control when the tires are near the friction limit. The vehicle is unstable when it is rotating to fast (spinning) or sliding sideways too fast (drifting). When the vehicle is unstable there is no guarantee that there is an action that will bring the vehicle closer to stability [8], essentially meaning a loss of control. Typically on-board safety systems like the Electronic Stability Program (ESP) will prevent the vehicle from becoming unstable.

But in some situations instability can be beneficial. Trained racing drivers can use it for more maneuverability in corners by drifting a vehicle around tight corners. This extra maneuverability can be crucial in emergency situations, for example for avoiding a collision. There may be a safe path that avoids the collision that can only be reached if the vehicle is temporarily brought in an unstable state. Then it is highly undesirable that systems like ESP will not allow it.

1-3 Related work and research gap

1-3-1 Classic control for automated vehicles

In classic control there are multiple paradigms to balance safe navigation and vehicle stability at the limit of friction.

Simple control law can be used [9, 10] but more sophisticated strategies are using Model Predictive Control (MPC) [11, 12] and Nonlinear Model Predictive Control (NMPC) [13, 14].

MPC is a model-based control technique that requires a model of the system in question. The control actions are optimized with a model of how the system responds to inputs. For automated driving, this would be a model that describes how the car responds to steering, throttle, and braking inputs and how the car interacts with the environment. Building a representative model can be difficult and time-consuming, especially for complex systems with unknown dynamics.

MPC is widely used because of its ability to solve constrained control problems. Constraints are boundaries that can be set in control problems and are often related to the safety or physical limits of a system. For example; if the objective is to accelerate a car from 0 to 100 km/h, a constraint of 1 m/s^2 on the acceleration ensures that the car accelerates comfortably and not at full power.

1-3-2 Reinforcement Learning for automated vehicles

Reinforcement Learning (RL) is a machine learning technique for training an agent by interacting with the environment or from data gathered from the system. This eliminates the need to make models of complicated systems.

RL has been proven successful on a wide range of different tasks that would be nearly impossible using classic control methods. Examples include playing intricate video games at competitive levels [15, 16, 17], intricate robot control [18, 19, 20, 21, 22, 23, 24, 25, 26] and image recognition and object detection [27, 28, 29, 30].

Using RL for driving automated vehicles has been explored before as well. Experiments show that reinforcement learning can be used to drive a car in urban environments based on camera vision [31, 32], navigate in traffic [33] and follow target trajectories [34]. Others have successfully trained policies for on-ramp and highway merging manoeuvres [35, 36, 37], highway driving [38, 39, 40], automated parking [41, 42, 43] or driving games with simplified car physics [44, 45]. Even applications of reinforcement learning in racing [46, 47] and drifting [48] environments have shown promising results.

1-3-3 Research gap

The previously mentioned work using RL for automated vehicles is focused on correctly identifying the environment, planning a path through the environment, or tracking a predefined path. For real-world applications, it is crucial to have a control strategy that can use the full handling capabilities of the vehicle when necessary. This could help avoid a collision or keep the vehicle safe on slippery surfaces.

To the best of our knowledge, there are at this moment in time no publications on using safe RL in automated vehicles with a focus on vehicle safety and stability at the limit of friction.

1-4 Research objective and approach

1-4-1 Objective

The objective of this work is to formulate a control policy that can be used to safely navigate automated vehicles at the limit of friction using safe RL.

The goal for the controller is to keep the vehicle within the limits of vehicle stability while following a target path at high speed. The stability limits may only be violated when the extra maneuverability is needed to keep the vehicle in the safe limits of the environment.

The controllers will only manipulate the steering angle of the front wheels, the throttle and brake inputs are regulated externally.

Additionally, a policy is trained to find the fastest way around a racetrack by optimizing the racing line to show the versatility of safe RL.

1-4-2 Approach

The safe RL policies for vehicle steering are trained using the Lyapunov Safe Actor-Critic (LSAC) algorithm [49][50][51]. LSAC is a combination of the Soft Actor-Critic (SAC) algorithm [52][53] with Lyapunov stability analysis and can solve constrained control problems. The constraints are used to define vehicle stability and the safe limits of the road.

SAC has outperformed most other learning algorithms in the benchmark tests of OpenAI's Gym [54] environment, but it is limited to unconstrained control problems.

The performance of the LSAC policies is compared against SAC and an MPC strategy based on the work of Funke et al. [11, 12]. The MPC is built in the MATLAB and Simulink environments.

The policies are trained and tested in a vehicle simulator that is built in OpenAI's Gym environment. An identical simulator is built in Simulink to test the MPC strategy.

The controllers will be tested in the following scenarios:

- Navigating through traffic at 50, 75 and 100 km/h
- Recovering from destabilizing impacts
- Driving on a race track and maintaining vehicle stability
- Safe navigation on a race track and maintaining vehicle stability
- Finding the racing line on a race track inside track limits

1-5 Outline

The main text of the thesis is structured as follows: Chapter 2 explains the vehicle simulation and how the environment is defined. The implementation of the MPC baseline is discussed in Chapter 3. Chapter 4 explains the basics of RL and the details of the SAC and LSAC algorithms specifically. Chapter 5 discusses the details of the training process and how the algorithms interact with the simulation environment. The experiments and results can be found in Chapter 6. Chapter 7 discusses the findings of this work and the recommendations for future work.

Chapter 2

Vehicle Simulation, Environment and Constraints

2-1 Introduction

Developing and testing new control strategies on real vehicles is generally not a viable option. It can be unsafe, time-consuming, and very costly. A good alternative is to use a vehicle simulator. In this work, a vehicle simulator is used to develop and train the control policies, as well as to evaluate the performance of the different control strategies.

For research purposes there are many simulators available, such as CARLA [55], TORCS [56], DeepDrive [57] and Constellation [58]. For this work, it is chosen to build a new simulator that is structured to be part of OpenAI's Gym [54]. Gym is an open-source toolkit for developing reinforcement learning algorithms and is compatible with the machine learning platform TensorFlow [59]. An identical copy of the simulator is built in the MATLAB environment to connect with the MPC strategy that is implemented in Simulink.

The simulator calculates the motion of the vehicle and provides an environment to drive in. The environment is a combination of a target path to follow and a target speed that is tracked by an external controller. The simulated vehicle is a rear-wheel drive passenger car. This chapter presents the properties and characteristics of the simulated car and the equations of motion used in the simulation.

The vehicle will be disturbed far outside the normal operational window in the experiments. The RL algorithms may also explore solutions outside the normal driving scope. Therefore is it crucial that the simulation is guaranteed to return valid and finite states at all times. Novel definitions of the tire slip ratios (Section 2-4-2) and a modified tire model (Section 2-4-3) ensure that this is the case.

2-2 Vehicle description and dimensions

The vehicle is a rear-wheel drive passenger car with a mass of 1600 kg, a wheelbase of 2.7 m, and a track width of 1.52 m. These dimensions are comparable to a modern family sedan. All other relevant vehicle dimensions are given in Table 2-1, with the corresponding assignment in Figure 2-1.

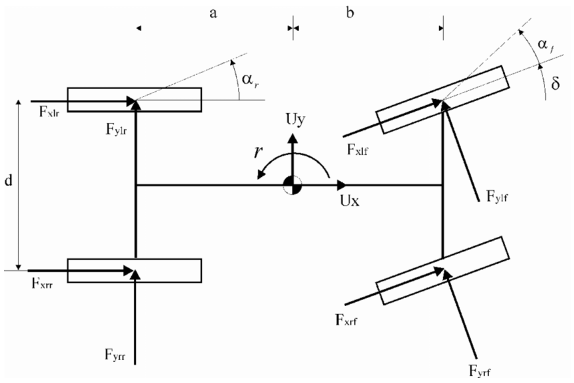


Figure 2-1: Schematic representation of the planar vehicle model and the definitions of the main forces, velocities and angles.

Table 2-1: Vehicle dimensions, weights and parameters

Parameter	Value	Unit	Description
m	1600	[kg]	mass of vehicle
l_f (b)	1.1	[m]	distance from CoG to front axle
l_r (a)	1.6	[m]	distance from CoG to rear axle
L	2.7	[m]	wheelbase
B (d)	1.52	[m]	track width
h_s	0.51	[m]	height CoG
h_f	0.08	[m]	height front roll center
h_r	0.13	[m]	height rear roll center
I_zz	2100	[kg m^2]	rotational inertia around Z-axis
C_k	105000	[N/-]	longitudinal tire stiffness
$C_{\alpha,\text{f}}$	57000	[N/rad]	front tire cornering stiffness
$C_{\alpha,\text{r}}$	36000	[N/rad]	rear tire cornering stiffness
e_r	0.35	[-]	friction reduction coefficient
r_w	0.3	[m]	wheel rolling radius
J_w	1	[kg m^2]	wheel rotational inertia
g	9.81	[m/s^2]	gravitational constant
δ_max	0.75	[rad]	max steering angle
$\dot{\delta}_\text{max}$	2π	[rad/s]	max steering rate

2-3 Vehicle handling

2-3-1 Neutral steer, understeer, and oversteer

The handling of a vehicle is defined as how it responds to steering inputs at different speeds. Different vehicles can show very different handling characteristics. A vehicle can be understeered, neutral steered, or oversteered.

A neutral steering vehicle will -with a fixed steering angle- keep the same turning radius when the speed is increased.

An understeered vehicle will underrotate in corners. This is because the front tires reach their friction limit before the rear tires, $\alpha_f > \alpha_r$. For a constant steering angle and increasing speed, the cornering radius will get larger and larger. Although an understeering car is slipping, it remains stable.

An oversteered vehicle will overrotate because the rear tires are saturated before the front tires, $\alpha_f < \alpha_r$. The driver has to lower the steering angle or even steer in the opposite direction of the corner to prevent the vehicle from spinning around. An oversteering vehicle is unstable and can be difficult to control. This is challenging behavior and that is why in this work an oversteered vehicle is used to test the controllers.

2-3-2 Understeer gradient K_{us}

Whether a vehicle will understeer or oversteer is captured in the understeer gradient K_{us}. For $K_{\text{us}} > 0$ the vehicle is understeered and for $K_{\text{us}} < 0$ oversteered. The front and rear tire cornering stiffnesses are the largest contributions to K_{us}, but other factors such as load transfer and suspension geometry have an effect as well.

A good approximation of K_{us} for the simulated vehicle in this work can be made by evaluating the front and rear cornering stiffnesses:

$$K_{\text{us}} = \frac{m \cdot g}{l_{\text{f}} + l_{\text{r}}} \left(\frac{l_{\text{r}}}{C_{\alpha,\text{f}}} - \frac{l_{\text{f}}}{C_{\alpha,\text{r}}} \right) = -0.414 \text{ [deg]} \tag{2-1}$$

Thus, the simulated vehicle is oversteered and will tend to overrotate in corners when the vehicle is pushed to its limits.

2-3-3 Critical speed v_{crit}

Oversteered vehicles have a critical speed v_{crit}. For v_{crit} the required steering angle for any corner becomes zero. The critical velocity for the simulated vehicle:

$$v_{\text{crit}} = \sqrt{\frac{g \cdot L}{-K_{\text{us}}}} = 60.55 \text{ m/s} \tag{2-2}$$

2-4 Equations of Motion

In this section the equations of motion of the simulated vehicle are listed. Each subsection details the specific equations and assumptions of a subsystem of the vehicle. All equations are integrated with explicit first order Euler integration with a step size of $h = 0.001$ seconds:

$$y_{n+1} = y_n + hf\left(t_n, y_n\right) \tag{2-3}$$

with $f\left(t_n, y_n\right)$ the derivative of y_n at $t = t_n$.

2-4-1 Wheel dynamics

The rotation speed ω of every wheel is is determined by:

$$\dot{\omega} = \frac{F_{\mathrm{x}} r_{\mathrm{w}} - T_{\mathrm{net}}}{J_{\mathrm{w}}} \tag{2-4}$$

With F_{x} the longitudinal force on the wheel, r_{w} the effective rolling radius, T_{net} the sum of the applied braking and driving torque, and J_{w} the rotational inertia of the wheel.

2-4-2 Tire slip definitions

The tires generate forces in the longitudinal direction (F_{x}) and the lateral direction (F_{y}) when the car is moving. F_{x} and F_{y} are the result of the relative motion between the tires and the road, called slip. Longitudinal slip is expressed as slip ratio κ, lateral slip is expressed as slip angle α.

Original Slip Ratio (OSR)

The tire states are calculated from the rotation of the wheels and the planar motion of the vehicle. The conventional definitions for tire slip are referred to as the Original Slip Ratio (OSR) [60].

The longitudinal tire slip κ is defined as the speed difference at the tire-road interface scaled by the longitudinal velocity:

$$\kappa^{\mathrm{OSR}} = \frac{\omega r_{\mathrm{w}} - u}{|u|} \tag{2-5}$$

The lateral slip angle α is the angle between the lateral and longitudinal velocity vectors v and u:

$$\alpha^{\mathrm{OSR}} = \delta - \tan^{-1}\left(\frac{v}{|u|}\right) \tag{2-6}$$

with δ steering angle of the wheel.

The Original Slip Ratio (OSR) is valid for general handling simulations at medium to high speeds but at low speeds this definition is numerically unstable. For $|u| \to 0$ Equations 2-5 and 2-6 are undefined.

Integration stability

Explicit integration of the wheel dynamics is unstable at low speeds when using the OSR slip definitions. The linearized system of the wheel and tires becomes increasingly stiff when $|u| \to 0$ or $|v| \to 0$.

The stability of the integration can be determined from the eigenvalues of the linearized system. For explicit Euler integration with step size h the wheel dynamics (Equation 2-4) are stable for u and v greater than the marginal velocities $\bar{u}_\mathrm{m}$ and $\bar{v}_\mathrm{m}$ [61]:

$$|\bar{u}_\mathrm{m}| = \frac{h}{2} C_\mathrm{x} \left(\frac{r_\mathrm{w}^2}{J_\mathrm{w}} + \frac{1}{m_\mathrm{e}} \right) \tag{2-7}$$

$$|\bar{v}_\mathrm{m}| = \frac{h}{2} \frac{C_\mathrm{y}}{m_\mathrm{e}} \tag{2-8}$$

with C_x and C_x the longitudinal and lateral tire stiffness and m_e the effective corner mass. Different slip definitions are needed to run a valid simulation for u and v smaller than the marginal velocities $\bar{u}_\mathrm{m}$ and $\bar{v}_\mathrm{m}$.

Lagged Slip Ratio (LSR)

A common solution is an alternative slip definition called the Lagged Slip Ratio (LSR) [62]. LSR expresses κ and α as differential equations:

$$\dot{\kappa^\mathrm{LSR}} = -\frac{|u|}{L_\mathrm{x}} \kappa^\mathrm{LSR} + \frac{\omega r_\mathrm{w} - u}{L_\mathrm{x}} \tag{2-9}$$

$$\dot{\alpha^\mathrm{LSR}} = -\frac{|u|}{L_\mathrm{y}} \alpha^\mathrm{LSR} + \frac{v}{L_\mathrm{y}} \tag{2-10}$$

This is the slip definition used in commercially available vehicle simulation software such as CarSim and ADAMS/Car [63]. However, the LSR definition is still unstable for explicit integration methods.

Advanced Slip Ratio (ASR)

In this work the Advanced Slip Ratio (ASR) [64, 65] definitions will be used for κ and α. ASR is a new slip definition that is stable for explicit Euler integration and has proven to produce accurate results, even compared with implicit integration for LSR.

ASR is defined as:

$$\kappa^\mathrm{ASR} = \frac{\omega r_\mathrm{w} - u}{\max(|u|, \eta \bar{u}_\mathrm{m})} \tag{2-11}$$

$$\alpha^\mathrm{ASR} = \delta - \tan^{-1}\left(\frac{v}{\max\left(|u|, \eta \bar{v}_\mathrm{m}\right)} \right) \tag{2-12}$$

where η is a safety coefficient, set to 1.1 as suggested in [64].

The experiment in Figure 2-2 shows the stability of ASR and the instability of OSR for speeds under the marginal velocity $\bar{u}_{\mathrm{m}}$. $\bar{u}_{\mathrm{m}}$ for a single corner model with $m_{\mathrm{e}} = 400$ kg, $J_{\mathrm{w}} = 1$ kg m^2, r = 0.3 m, $C_x = 105000$ N/-, $h = 0.001$ s:

$$\bar{u_{\mathrm{m}}} = 4.856 \text{ m/s} \tag{2-13}$$

For an initial velocity of -10 m/s and an applied driving torque of 50 Nm:

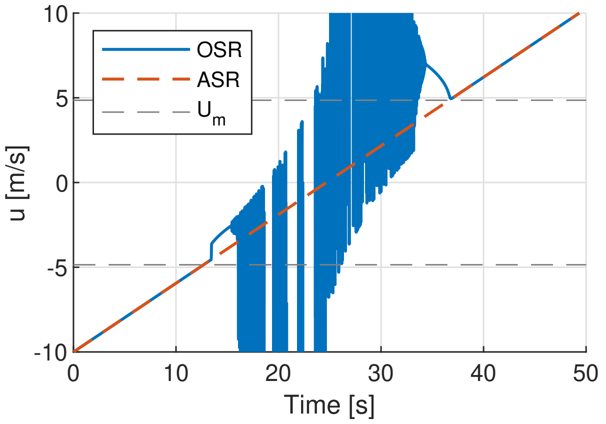


Figure 2-2: Vehicle velocity u over time from a simulation using the OSR and ASR slip definitions for an vehicle accelerating from -10 m/s with an applied torque of 50 Nm. Experiment ran with explicit Euler integration with a step size of 0.001 s where the marginal velocity $\bar{u}_{\mathrm{m}}$ is 4.856 m/s. OSR = Original Slip Ratio, ASR = Advanced Slip Ratio. [64, 65]

From the experiment in Figure 2-2 it is clear that OSR suffers large instability for $|u| < \bar{u}_{\mathrm{m}}$ while ASR remains completely stable for $|u| \to 0$.

2-4-3 Tire modelling

Simulating the behavior of tires accurately is still one of the biggest challenges in vehicle simulation. A tire model calculates the forces generated by the tires (F_{x} and F_{y}) as a result of tire slip κ and α. There are many tire models, each for specific purposes [66]. For example, tire models can be focused on analyzing ride quality or on simulating vehicle handling.

Well known models suitable for simulating vehicle handling include the Magic Formula [60], Delft-Tyre and Dugoff [67]. The Magic Formula is not based on physical modeling of the tire but it is an empirical model based on test data and function fitting. It is regarded as one of the best models for racing and vehicle handling simulators and the basis of MF-Tyre and MF-SWIFT [68] used in simulation software as ADAMS, Virtual.lab Motion and CarSim. With 20+ parameters to tune it is also quite unwieldy.

Instead in this work an adaptation of the Modified Dugoff model [69] is used. It strikes a good balance between model accuracy and mathematical simplicity [70].

Modified Dugoff

The Modified Dugoff tire model [69] is a simple set of equations to calculate the longitudinal (F_x) and lateral (F_y) forces generated by the tire . Modified Dugoff is an adaptation of the original Dugoff model [67] to give more realistic behavior at larger slip angles.

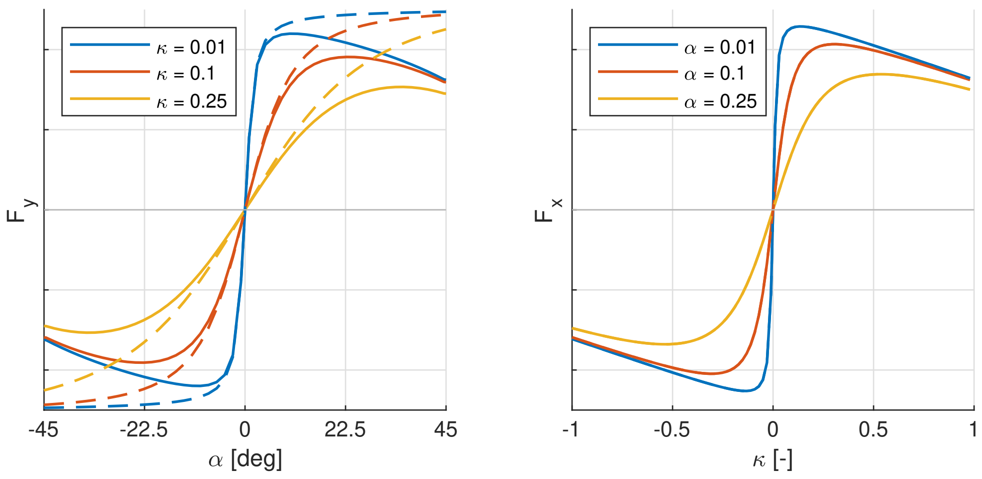


Figure 2-3: Lateral tire force F_y vs. slip angle α (left) and longitudinal tire force F_x vs. wheel slip κ (right) for the Dugoff and Modified Dugoff tire models. The dashed lines are the original Dugoff model, the solid lines are the Modified Dugoff model.

The longitudinal tire forces F_x and lateral tire forces F_y in the Modified Dugoff tire model:

$$F_\mathrm{x} = C_\mathrm{x}\sigma_\mathrm{x} f(\lambda) \tag{2-14}$$

$$F_\mathrm{y} = C_\mathrm{y}\sigma_\mathrm{y} f(\lambda) \tag{2-15}$$

where

$$\sigma_x = \frac{\kappa}{1-\kappa} \tag{2-16}$$

$$\sigma_y = \frac{\tan\alpha}{1-\kappa} \tag{2-17}$$

$$f(\lambda) = \begin{cases} \lambda(2-\lambda), & \text{if } \lambda < 1 \\ 1, & \text{if } \lambda \geq 1 \end{cases} \tag{2-18}$$

$$\lambda = \frac{\mu F_z(1-\kappa)\left(1 - e_\mathrm{r}\sqrt{\kappa^2 + \tan^2\alpha}\right)}{2\sqrt{(C_x\sigma_x)^2 + (C_y\tan\alpha)^2}} \tag{2-19}$$

Stable Modified Dugoff

The experiments in this work will include extreme cases of vehicle handling. The vehicle will be disturbed in violent maneuvers and the RL algorithms may explore vehicle states far outside the valid range of the Modified Dugoff tire model. It is important that the tire model returns valid forces at all times and the simulations do not crash due to infinite of undefined values.

To guarantee valid behavior at all times this work introduces the Stable Modified Dugoff model. The modifications do not influence normal operation but will keep the simulation from crashing during collisions, fast vehicle rotation, or pure lateral motion.

The following limitations are made to the Modified Dugoff tire model:

- The longitudinal slip ratio κ is limited to [-0.99, 0.99] because for $\kappa \in [-1, 1]$, σ_x is not finite (divide by 0).
- $|\tan(\alpha)|$ is limited to < 1, because for $\alpha \in [-90, 90]$, $\tan(\alpha)$ is not finite. This only affects the force generation for slip angles $\alpha > 45$ deg.

The Stable Modified Dugoff model is plotted against the Modified Dugoff model for all possible tire states in Figure 2-4.

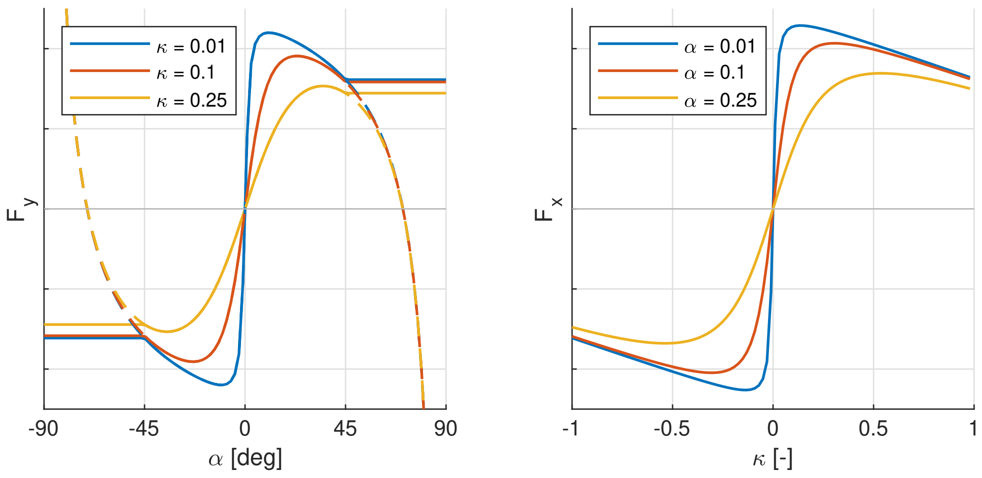


Figure 2-4: Lateral tire force F_y vs. slip angle α (left) and longitudinal tire force F_x vs. wheel slip κ (right). The solid lines are the Stable Modified Dugoff tire model,the dashed lines represent the Modified Dugoff tire model.

2-4-4 Vehicle body motion

The motion of the vehicle body is a result of the forces generated by the tires. Any aerodynamic effects are ignored.

Normal load and load transfer

The vertical load on a wheel F_z changes as the vehicle accelerates, decelerates or moves through corners. This effect is called load transfer. F_z is a function of the longitudinal and lateral accelerations; as well as the height of the center of gravity and height of the front and rear roll centers. The subscripts indicate the location of the wheel; fl = front left, rr = rear right.

$$F_{z,fl} = \frac{m \cdot g}{2L} \cdot l_r - a_x \cdot F_{z,\text{long}} - a_y \cdot F_{z,\text{lat},f} \tag{2-20}$$

$$F_{z,fr} = \frac{m \cdot g}{2L} \cdot l_r - a_x \cdot F_{z,\text{long}} + a_y \cdot F_{z,\text{lat},f} \tag{2-21}$$

$$F_{z,rl} = \frac{m \cdot g}{2L} \cdot l_f + a_x \cdot F_{z,\text{long}} - a_y \cdot F_{z,\text{lat},r} \tag{2-22}$$

$$F_{z,rr} = \frac{m \cdot g}{2L} \cdot l_f + a_x \cdot F_{z,\text{long}} + a_y \cdot F_{z,\text{lat},r} \tag{2-23}$$

with

$$F_{z,\text{long}} = m \cdot \frac{h_\text{s}}{2 \cdot L} \tag{2-24}$$

$$F_{z,\text{lat},f} = m \cdot \frac{l_r}{L} \cdot \frac{h_f}{B} \tag{2-25}$$

$$F_{z,lat,f} = m \cdot \frac{l_f}{L} \cdot \frac{h_r}{B} \tag{2-26}$$

Force conversion from wheels to body

The forces from the tire model are oriented in the wheel reference frame and need to be converted to the vehicle reference frame to account for the steering angles δ:

$$F_{x,i} = F_{xw,i} \cos \delta_i - F_{yw,i} \sin \delta_i \tag{2-27}$$

$$F_{y,i} = F_{xw,i} \sin \delta_i + F_{yw,i} \cos \delta_i \tag{2-28}$$

with δ_i the respective steering angle of each wheel. For the front wheels, left and right δ are assumed equal and Ackermann geometry is not included. The steering angle δ for the rear wheels is assumed 0 and any steering from body roll is ignored.

Individual wheel contributions to planar motion

$$F_x = F_{x,fl} + F_{x,fr} + F_{x,rl} + F_{x,rr} \tag{2-29}$$

$$F_y = F_{y,fl} + F_{y,fr} + F_{y,rl} + F_{y,rr} \tag{2-30}$$

$$T = (F_{y,fl} + F_{y,fr})\, l_f - (F_{y,rl} + F_{y,rr}+) l_r + (F_{x,fl} - F_{x,fr} + F_{x,rl} - F_{x,rr}) \frac{1}{2}\text{B} \tag{2-31}$$

Accelerations

$$\dot{u} = \frac{F_x}{m} + (r \cdot v) \tag{2-32}$$

$$\dot{v} = \frac{F_y}{m} - (r \cdot u) \tag{2-33}$$

$$\dot{r} = \frac{T}{I_{zz}} \tag{2-34}$$

2-5 Simulation environment

The environment is the virtual world of the simulated vehicle. Specifically, the environment creates different roads for the vehicle to drive on in the experiments. It also defines a target path (Section 2-5-1) and target velocity (2-5-3) for the vehicle to follow. The environment also provides the location and orientation of the vehicle with respect to the target, called the path states (Section 2-5-2).

2-5-1 Target path

A target path for the vehicle to follow is created in the environment. The path is defined as a function of $K(s)$ where K is the curvature of the path at distance s on the path. The curvature K is the inverse of the corner radius R, $K = \frac{1}{R}$.

E(s) is the position in east (X) direction, N(s) the position in the north (Y) direction, and ϕ the orientation. Given starting point E_0, N_0 and starting orientation ϕ_0 the path can be expressed in the Cartesian coordinate system:

$$\phi(s) = \int_0^s K(x)\,dx + \phi_0 \tag{2-35}$$

$$E(s) = \int_0^s \cos\phi(x)dx + E_0 \tag{2-36}$$

$$N(s) = \int_0^s \sin\phi(x)dx + N_0 \tag{2-37}$$

A sample corner with target path, path curvature and target speed is given in Figure 2-5.

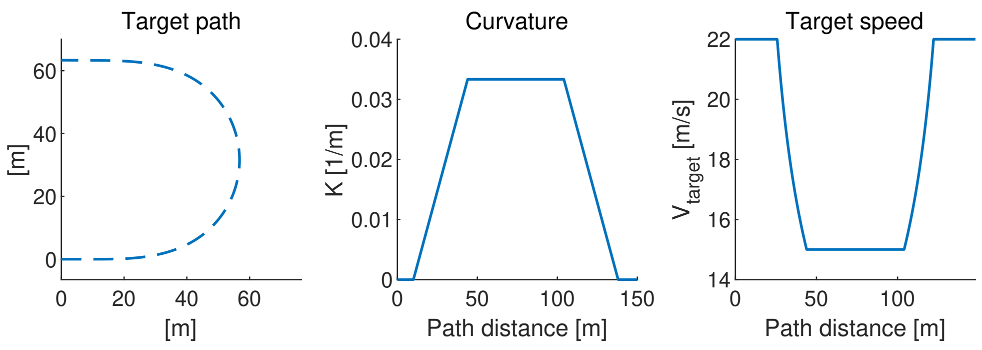


Figure 2-5: A sample path (left), the curvature of this path (middle) and the target velocity (right).

2-5-2 Path states

The position of the vehicle relative to the target path is defined with three variables. All distances are with respect to the center of gravity (CoG) of the vehicle.

The distance from the CoG to the nearest point on the target path is the lateral error e. The location on the path that is closest to the vehicle is path distance s. The heading angle between the vehicle and the path is the heading error $\Delta\phi$. The schematics are shown in Figure 2-6:

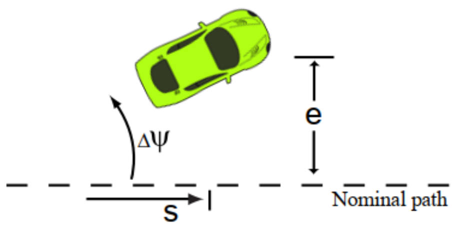


Figure 2-6: Path states e (lateral error), $\Delta\phi$ (heading error) and s (distance on path) are defined from the center of gravity (CoG) of the vehicle to the nearest point on the target path.

2-5-3 Limit velocity V_{max} and target velocity V_{target}

The maximum possible speed of a vehicle through a corner depends on many factors such as road surface and vehicle handling. V_{max} is an approximation of the theoretical longitudinal speed limit for each point on the target path, based only on the tire-road friction μ and corner curvature K. It is derived from simplified tire theory and the assumption of steady-state cornering. The target velocity V_{target} is then set as a percentage of V_{max}.

V_{max}

The maximum force that a tire can generate be approximated by the friction circle. The vector of the longitudinal tire forces F_x and lateral tire forces F_y can never be greater than the normal force F_z times the tire-road friction μ:

$$\sqrt{F_x^2 + F_y^2} \leq \mu F_z \tag{2-38}$$

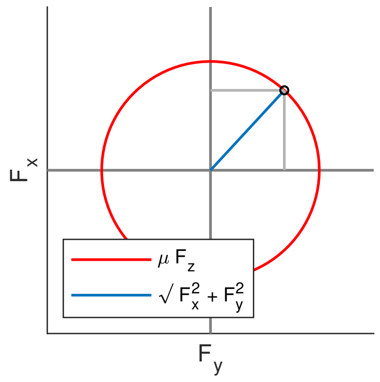


Figure 2-7: The tire friction circle from Equation 2-38 approximates the maximum tire forces F_x and F_y for a given tire-road friction coefficient μ.

Dividing out the vehicle mass gives the limit of accelerations:

$$\sqrt{a_x^2 + a_y^2} \leq \mu g \tag{2-39}$$

with g the gravitational acceleration. In steady state cornering the longitudinal acceleration $a_x = 0$, so the lateral acceleration a_y must satisfy:

$$a_y \leq \mu g \tag{2-40}$$

For steady state angular motion the lateral acceleration a_y is a function of corner curvature K and the longitudinal velocity V_x:

$$a_y = V_x^2 K \tag{2-41}$$

Combining Equations 2-40 and 2-41 and setting $V_x = V_{\mathrm{max}}$ gives the theoretical limit V_{max}:

$$V_{\text{max}} = \sqrt{\frac{\mu g}{K}} \tag{2-42}$$

V_{target}

At V_{max} the tires are completely saturated according to the friction circle theory in Equation 2-38; $F_y = F_{y,\text{max}}$. The desired friction utilisation μ_{des} in that case is 100%.

$\mu_{\text{des}} \in [0,1]$ is used to scale V_{max} to V_{target}. For $\mu_{\text{des}} = 0.75$, the target velocity V_{target} is such that the tires will operate at 75% of the maximum tire force generation. The relation between V_{target}, V_{max} and μ_{des}:

$$V_{\text{target}} = \sqrt{\frac{\mu_{\text{des}}\mu g}{K}} = \sqrt{\mu_{\text{des}}} \cdot V_{\text{max}} \tag{2-43}$$

Defining V_{target} as a percentage of V_{max} with μ_{des} rather than in absolute speed makes it convenient to compare vehicle performance in different corners or on different surfaces. The target velocity V_{target} for the sample path in Figure 2-5 for different μ_{des} can be found in Figure 2-8.

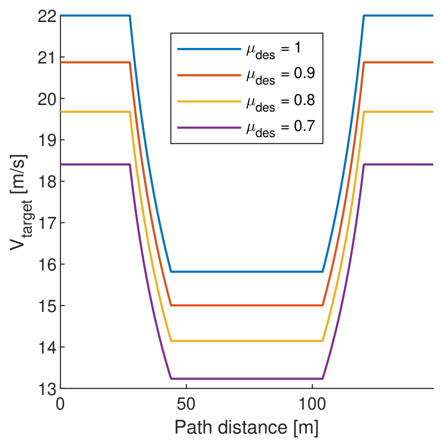


Figure 2-8: Target velocity V_{target} for different values of desired friction utilisation μ_{des} for the sample path from Figure 2-5.

The notation of $\sqrt{\mu_{\text{des}}} \cdot V_{\text{max}}$ is used in the experiments in Chapter 6 to denote V_{target} in different scenarios.

2-5-4 Relating V_{max} to normal driving

The maximum velocity V_{max} is quite an abstract concept. This section explains how V_{max} and V_{target} relate to normal driving speeds.

Studies on driving comfort suggest that the lateral acceleration a_y in corners should be kept below 2 m/s^2, and the range for normal driving does generally not exceed 4 m/s^2 [71].

For driving on dry asphalt with $\mu = 1$ and $g = 9.81$ m/s^2, the maximum lateral acceleration is 9.81 m/s^2 (Equation 2-39):

$$a_y \leq \mu g = 9.81 \text{ m/s}^2 \tag{2-44}$$

If instead a_y is limited to 2 m/s^2 for comfort the friction utilization μ_{des} is limited to 20% of the maximum:

$$V_{\text{comfort}} = \sqrt{\frac{2}{9.81}} \cdot V_{\max} = \sqrt{0.204} \cdot V_{\max} \tag{2-45}$$

For regular driving where a_y is limited to 4 m/s^2 the friction utilization μ_{des} is limited to 41%:

$$V_{\text{regular}} = \sqrt{\frac{4}{9.81}} \cdot V_{\max} = \sqrt{0.408} \cdot V_{\max} \tag{2-46}$$

In the experiments for driving near the friction limit values for μ_{des} are between 0.80 and 0.90. This is in the same range as tested by [12].

2-6 Longitudinal speed control

The target velocity of the vehicle V_{target} is provided by the environment. The longitudinal speed control is done externally because the tested control strategies are only set up to manipulate the steering of the vehicle.

The vehicle tracks V_{target} with a simple proportional control law. Positive engine torque is applied to the rear wheels, the braking torque is applied to all wheels.

The controller is limited to 400 Nm of engine torque to each of the rear wheels, for a total of 800 Nm of engine torque. Braking torque is limited to 600 Nm per wheel on the front axle and 400 Nm per wheel on the rear axle for a total braking torque of 2000 Nm.

For reference, this is enough power to accelerate the vehicle from 0 to 100 km/h in 5.8 seconds and brake from 100 to 0 km/h in 3.7 seconds.

The braking force is distributed between the front and rear according to the static weight distribution (60/40)%.

The desired torque T_{des} is set as:

$$T_{\text{des}} = \text{clip}\left(1000 \cdot \left(V_{\text{target}} - V\right), -1000, 400\right) \ \text{[Nm]} \tag{2-47}$$

The throttle and braking system apply the desired torque T_{des} to the wheels:

$$T_{\text{net(fl,fr)}} = \begin{cases} 0 & \text{if } T_{\text{des}} \geq 0 \text{ (accelerating)} \\ 0.6 \cdot T_{\text{des}} \cdot \text{sign}\left(\omega\right) & \text{if } T_{\text{des}} < 0 \text{ (braking)} \end{cases} \tag{2-48}$$

$$T_{\text{net(rl,rr)}} = \begin{cases} T_{\text{des}} & \text{if } T_{\text{des}} \geq 0 \text{ (accelerating)} \\ 0.4 \cdot T_{\text{des}} \cdot \text{sign}\left(\omega\right) & \text{if } T_{\text{des}} < 0 \text{ (braking)} \end{cases} \tag{2-49}$$

2-7 Constraints

Two sets of constraints are used to describe the safe limits of the vehicle with respect to vehicle stability and the environment. The constraint on vehicle stability is explained in Section 2-7-1, the environmental constraint in Section 2-7-2. These constraints are based on the work of [72].

2-7-1 Stability constraints

Limits on the lateral velocity U_y and yaw rate r are introduced to act as a stability boundary. If the vehicle goes outside these boundaries, there is no guarantee that there exists a control input in the next time step that will get the states back inside or closer to the boundaries. Given longitudinal velocity U_x the stability envelope is defined as:

$$U_{y,\text{max}} = U_x \alpha_{r,\text{sat}} + l_r r \tag{2-50}$$

$$r_{\text{max}} = \frac{g\mu}{U_x} \tag{2-51}$$

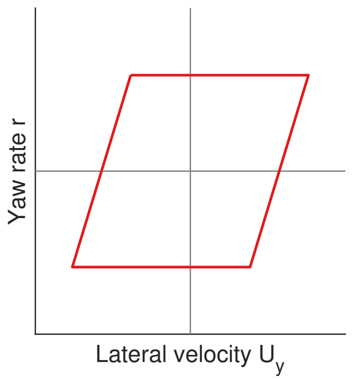


Figure 2-9: Stability constraints [72] with limits on the lateral velocity U_y and yaw rate r. The lateral velocity limit prevents sliding of the vehicle sideways, the yaw rate limit prevents the vehicle from spinning around.

$\alpha_{r,\text{sat}}$ is the saturation angle of the rear tire where the maximum lateral force $F_{y,\text{max}}$ is generated. For $|\alpha| > \alpha_{\text{sat}}$, $\Delta F_y \leq 0$. α_{sat} is approximated as [60, 73]:

$$\alpha_{\text{sat}} = \tan^{-1}\left(\frac{3\eta\mu F_z}{C_\alpha}\right) \tag{2-52}$$

where η is a derating factor to account for combined slip conditions:

$$\eta = \frac{\sqrt{\mu^2 F_z^2 - F_x^2}}{\mu F_z} \tag{2-53}$$

2-7-2 Environmental constraint

The environmental envelope is a set of safe path states for the vehicle. It describes where the vehicle can be without leaving the road or colliding with objects. In this work, only the lateral tracking error e of the CoG is considered, and not the orientation and outside dimensions of the vehicle. For every point s in the environment, e must satisfy:

$$e_{\min}(s) \leq e(s) \leq e_{\max}(s) \tag{2-54}$$

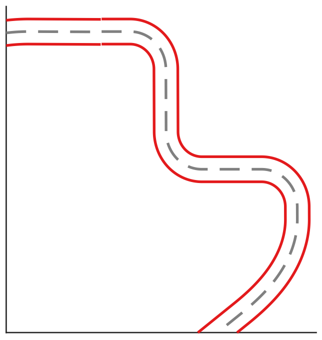


Figure 2-10: Environmental constraint to limit the vehicle to stay in the boundaries of the road.

Chapter 3

Model Predictive Control

3-1 Introduction

This chapter discusses the implementation of Model Predictive Control (MPC) as a performance baseline for the experiments in this work. The MPC in this work is an adaptation from the safe driving strategy presented in [11] and [12], with environmental constraints from [72] and stability constraints from [8]. The MPC is built in the Simulink environment with the Model Predictive Control Toolbox$^{\text{TM}}$.

Model predictive control is a model-based control method. A model of the system is used to predict future states and optimize the planned control inputs. The control inputs are optimized for a short time in the future called the prediction horizon. After the first action is executed the optimization starts again for the next prediction horizon.

MPC is often used for its ability to solve constrained control problems. Constraints are custom limits on a system that are often used to ensure safe operation. For example, constraints can be used to limit the rate of change in temperature in a chemical plant. In this work constraints are used to limit the car to be safe and stable; see Sections 2-7-1 and 2-7-2.

MPC is a transparent and predictable control method but it requires a representative model of the system. Creating an accurate model of the system can be a difficult and time-consuming task, especially for complex systems or systems with unknown dynamics.

3-2 Overview of operation

A nominal path is set up in the environment as defined in Section 2-5-1. This gives the controller information about the current path states (e, $\Delta\phi$, s) and how the path will evolve over the prediction horizon. The primary objective of the controller is to minimize the path tracking errors e and $\Delta\phi$.

The controller is given 2 constraints. The first constraint is on vehicle stability (Section 2-7-1). This prioritizes vehicle stability over path tracking. The second constraint is on the environment (Section 2-7-2). This constraint can force the vehicle in unstable states if it is necessary to avoid driving off the road.

3-3 MPC prediction model

3-3-1 Bicycle model

The vehicle motion is captured in the kinematic bicycle model. A bicycle model is a two-wheel representation of a car used to capture the most essential vehicle dynamics in a simple set of equations. The relative simplicity of this model makes it viable to use in real-time MPC applications without the need for excessive computational power [74, 75].

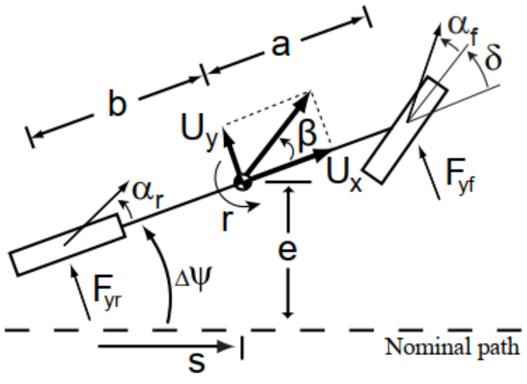


Figure 3-1: A schematic overview of the bicycle model and the path states that are used to predict the vehicle motion in the MPC. Schematic from [12].

Vehicle states

The following equations of motion describe the lateral velocity U_y and yaw rate r:

$$\dot{U}_y = \frac{F_{yf} + F_{yr}}{m} - rU_x \tag{3-1}$$

$$\dot{r} = \frac{aF_{yf} - bF_{yr}}{I_{zz}} \tag{3-2}$$

Here F_{yf} and F_{yr} are the front and rear lateral tire forces, m is the mass of the vehicle, a and b are the distances from the front and rear wheels to the center of gravity (CoG), and I_{zz} is the rotational moment of inertia of the vehicle.

F_{yf} is the variable manipulated by the MPC. How F_{yf} is related to steering angle δ can be found in Section 3-3-4.

Path states

The path states are defined as the heading error $\Delta\phi$, lateral error e and the location on the path s, see Section 2-5-2. The corresponding equations of motion for the path states are:

$$\Delta\dot{\phi} = r - \dot{s}K(s) \tag{3-3}$$

$$\dot{e} = U_x \sin\Delta\phi + U_y \cos\Delta\phi \approx U_x\Delta\phi + U_y \tag{3-4}$$

$$\dot{s} = \frac{U_x \cos\Delta\phi - U_y \sin\Delta\phi}{1 - K(s)e} \approx \frac{U_x}{1 - K(s)e} \tag{3-5}$$

3-3-2 State Space model

The equations of the bicycle model can be captured in a state space formulation to use in MPC. In the notation $\dot{x} = Ax + BF_{yf} + C$ as in [12]:

$$\begin{bmatrix} \dot{U}_y \\ \dot{r} \\ \Delta\dot{\phi} \\ \dot{s} \\ \dot{e} \end{bmatrix} = \begin{bmatrix} 0 & -U_x & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & U_x & 0 & 0 \end{bmatrix} \begin{bmatrix} U_y \\ r \\ \Delta\phi \\ s \\ e \end{bmatrix} + \begin{bmatrix} 1/m \\ a/I_{zz} \\ 0 \\ 0 \\ 0 \end{bmatrix} F_{yf} + \begin{bmatrix} F_{yr}/m \\ -F_{yr} \cdot b/I_{zz} \\ -\dot{s}K(s) \\ U_x/1 - K(s)e \\ 0 \end{bmatrix} \tag{3-6}$$

This notation shows that the only manipulated variable is F_yf, while the C matrix essentially represents external disturbances in the system. Converted to the more common $\dot{x} = Ax + Bu$ notation:

$$\begin{bmatrix} \dot{U}_y \\ \dot{r} \\ \Delta\dot{\phi} \\ \dot{s} \\ \dot{e} \end{bmatrix} = \begin{bmatrix} 0 & -U_x & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & U_x & 0 & 0 \end{bmatrix} \begin{bmatrix} U_y \\ r \\ \Delta\phi \\ s \\ e \end{bmatrix} + \begin{bmatrix} 1/m & 1 & 0 & 0 & 0 & 0 \\ a/I_{zz} & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} F_{yf} \\ F_{yr}/m \\ -F_{yr} \cdot b/I_{zz} \\ -\dot{s}K(s) \\ U_x/1 - K(s)e \\ 0 \end{bmatrix} \tag{3-7}$$

For the implementation in MATLAB this is modelled as a system with 1 manipulated variable (F_{yf}) and three measured disturbances (F_{yr}, $-\dot{s}K(s)$, $U_x/1 - K(s)e$):

$$\begin{bmatrix} \dot{U}_y \\ \dot{r} \\ \Delta\dot{\phi} \\ \dot{s} \\ \dot{e} \end{bmatrix} = \begin{bmatrix} 0 & -U_x & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & U_x & 0 \end{bmatrix} \begin{bmatrix} U_y \\ r \\ \Delta\phi \\ s \\ e \end{bmatrix} + \begin{bmatrix} 1/m & 1/m & 0 & 0 \\ a/I_{zz} & -b/I_{zz} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} F_{yf} \\ F_{yr} \\ -\dot{s}K(s) \\ U_x/1 - K(s)e \end{bmatrix} \tag{3-8}$$

Matrix A is updated every time step with Adaptive Model Predictive Control to reflect the changing velocity U_x.

3-3-3 Predicting F_{yf} and F_{yr}

A tire model is needed to predict F_{yf} and F_{yr} from vehicle states U_x, U_y and r for every step in the prediction horizon. The vehicle motion is converted to slip angle α and normal force F_z, then the tire model converts this to F_{yf} and F_{yr}.

To give MPC the full advantage its internal tire model is the same as the tire model of the simulation vehicle. This is the Stable Modified Dugoff tire model and can be found in Section 2-4-3.

Slip angle α can be estimated from longitudinal velocity U_x, lateral velocity U_y, yaw rate r and steering angle δ. In practice some of these states can be difficult to measure, but in this work they are assumed to be known or observable. The slip angles for the front and rear wheels for the bicycle model are defined as:

$$\alpha_f = \tan^{-1}\left(\frac{U_y + ar}{U_x}\right) - \delta \tag{3-9}$$

$$\alpha_r = \tan^{-1}\left(\frac{U_y - br}{U_x}\right) \tag{3-10}$$

F_z is the normal force on the tires, which changes due to weight transfer during acceleration or deceleration. In the bicycle model, the lumped weight on the entire axle is used.

For the front axle:

$$F_{zf} = \frac{1}{L}(mbg - hF_x) \tag{3-11}$$

For the rear axle:

$$F_{zr} = \frac{1}{L}(mag + hF_x) \tag{3-12}$$

For each step k in the prediction horizon F_{yf} and F_{yr} are then given by the tire model:

$$F^k_{y,f} = f_{\text{tire}}\left(F^k_{z,f}, \alpha^k_f\right) \tag{3-13}$$

$$F^k_{y,r} = f_{\text{tire}}\left(F^k_{z,r}, \alpha^k_r\right) \tag{3-14}$$

3-3-4 F_{yf} to δ conversion

The output from the MPC controller is $F_{y,f}$; the desired lateral force on the front tires. However the input on the car is the steering angle δ. To translate $F_{y,f}$ to the steering angle for the front wheels δ is expressed as:

$$\delta = \tan^{-1}\left(\frac{U_y + ar}{U_x}\right) - f^{-1}_{\text{tire}}(F_{y,f}) \tag{3-15}$$

In this expression f^{-1}_{tire} is the inverse of the tire model.

3-3-5 Prediction horizon

The controller optimizes the inputs over the prediction horizon. The prediction of the states in the prediction horizon during a cornering maneuver is shown in Figure 3-2. It shows that the bicycle model is a valid approximation for the reality of the real (simulated) vehicle.

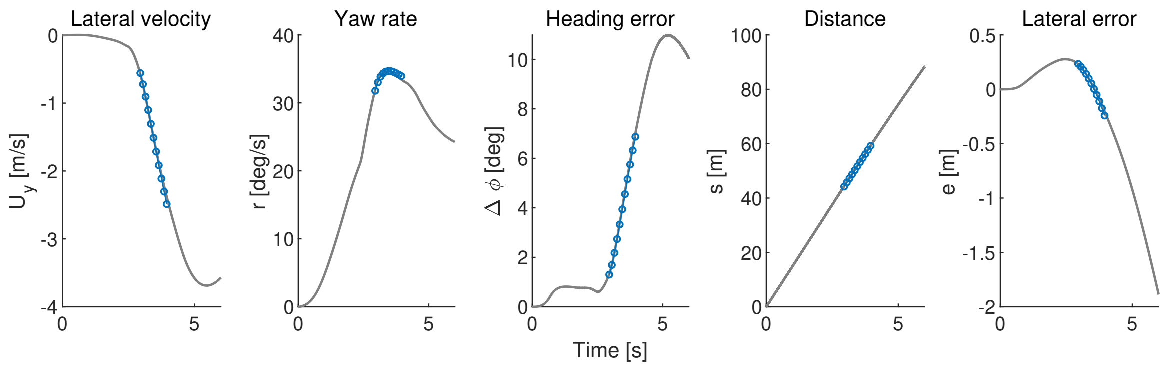


Figure 3-2: The MPC prediction (dotted, blue) during a cornering maneuver vs. the actual evolution of the states (solid, black).

3-4 Constraints

Constraints are limits on model states or outputs. They can represent physical limits such as the maximum steering angle, or desired behavior such as speed limits or minimum distances to the edges of the road. Here the constraints are used to limit lateral velocity U_y and yaw rate r, as well as limit the vehicle to the safe boundaries of the road (Figure 3-3). These constraints are to keep the car stable and safe.

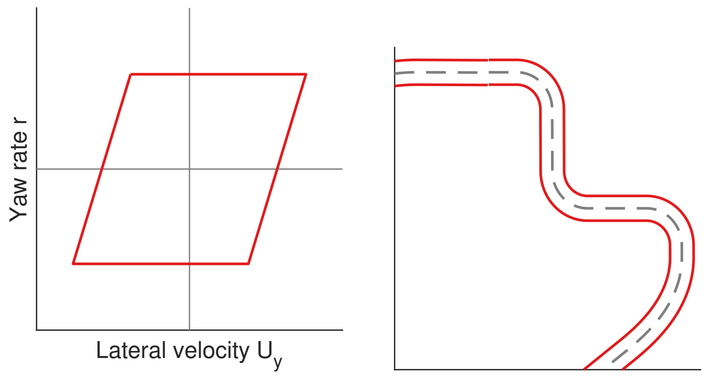


Figure 3-3: The constraints that are applied on the vehicle stability (left) and the environment (right). See Sections 2-7-1 and 2-7-2.

For each step k in the prediction horizon the constraints must be satisfied as follows:

Lateral velocity:

$$U_y^k \leq U_x^k \alpha_{r,\text{sat}}^k + l_r r^k \tag{3-16}$$

Yaw rate:

$$r^k \leq \frac{g\mu}{U_x^k} \tag{3-17}$$

Lateral path tracking error:

$$e_{\text{min}}^k(s) \leq e^k(s) \leq e_{\text{max}}^k(s) \tag{3-18}$$

3-4-1 Constraint softening

It is not always possible to satisfy all constraints. An outside disturbance can force a violation of one of the constraints. For example; the yaw rate of the car can be constrained but a collision from the side can rotate the car faster than the constraint prescribes.

If the constraints are set as 'hard' constraints it means that strictly no violation is allowed. When a violation occurs or a violation is predicted, the controller freezes its action because the problem is infeasible; there is no mathematical solution.

To solve this the constraints are 'softened'. Every constraint has an Equal Concern for Relaxation (ECR) value. Higher values of ECR mean that a larger violation of the constraint is allowed and the constraint is 'softer'. Low values of ECR make the constraint 'harder' and mean that only small violations are allowed. Constraints are hard for ECR equal to zero. The ECR values are tuned to prioritize the environmental constraint over vehicle stability. The ECR values for each constraint are in Table 3-1.

3-5 Controller settings

The settings and tuning weights used for the Adaptive Model Predictive Controller in MATLAB are listed in Table 3-1.

The controller weights are optimized on the car without constrains until the car converges to a target from $e = 3$ meters quickly and without too much overshoot.

Then the constraints on vehicle stability are added and the ECR is decreased until the car shows little to no stability violations when driving near the friction limit. Lastly, the environmental constraint is added and its ECR is lowered until it can force stability violations near the edge of the road.

Table 3-1: Controller settings used for the adaptive MPC controller in MATLAB/Simulink.

Parameter	Value	Unit	Description
Prediction horizon	1	[s]	Length of the prediction horizon
Controller time step	0.02	[s]	-
$\Delta\phi$ weight	0.2	[-]	-
e weight	1	[-]	-
$\Delta\phi$ scale	0.15	[-]	Max expected $\Delta\phi$
e scale	5	[-]	Max expected e
MV_{max}	10000	[N]	Max for manipulated variable ($F_{y,f}$)
$MV_{\text{rate,max}}$	10000	[N/s]	Max rate of change manipulated variable
ECR $MV_{\text{rate,max}}$	10	[-]	Softening on MV_{rate}
ECR U_y	50	[-]	Softening on U_y constraint
ECR r	50	[-]	Softening on r constraint
ECR e	0.5	[-]	Softening on e constraint

Chapter 4

Reinforcement Learning

4-1 Introduction

Reinforcement learning (RL) is a machine learning process to form a control policy for a control problem. One of the main benefits of RL with respect to classical control methods such as MPC is that there is no need to manually capture the important dynamics of a system in a model. Instead, a control policy can be 'trained' by interacting with the environment or from data gathered previously. The desired behavior of the control policy is determined by feedback from the environment. Good behavior is rewarded and bad behavior is penalized.

There are many examples of control problems that have been solved by RL that would be difficult to solve with classic control methods, ranging from image recognition to walking robots. For more examples and a list of publications, the reader is referred to the main introduction of this thesis (Chapter 1).

Training a control policy typically takes a lot of interactions with the environment to explore the possible solutions. That is why it is often not feasible to train directly on a physical system. Instead, the agent is trained in a virtual environment where lots of actions can be tried relatively quickly.

There are many different RL algorithms, with quite a few that are combinations or variations of others. There are a few factors to consider when choosing an algorithm to use but the main distinction is the compatibility with discrete or continuous action spaces. For example, algorithms like Rainbow [76], ACER [77] and DQN [15] can only solve problems with a discrete action space. For continuous action spaces there are -among others- DDPG [26], TD3 [78] or SAC [52, 53]. Other algorithms like Policy Gradient [79], PPO [80] and A2C/A3C [81] are compatible with both discrete and continuous action spaces.

In this work the Soft Actor-Critic (SAC) [52][53] and the Lyapunov Safe Actor-Critic (LSAC) [49][50][51] algorithms are used. SAC is among the most popular RL algorithms for continuous control tasks and has outperformed most others in the benchmark tests of OpenAI's Gym [54]. LSAC is a new algorithm that combines SAC with a stability analysis to solve constrained control problems.

The following topics will be discussed in the remainder of this chapter. In Section 4-2 the most important principles and definitions used in RL are explained. Section 4-3 builds on this basis and introduces the key RL algorithms with important aspects for SAC and LSAC. Section 4-4 and 4-5 go further into the details of the SAC and LSAC algorithms.

4-2 Principles of RL

In this section, the basic principles of RL are explained. These principles are the building blocks of most relevant RL algorithms and in particular the SAC and LSAC algorithms that are used later in this work.

4-2-1 Markov Decision Process

The control problems in RL are typically modeled as Markov Decision Processes (MDP). The 'Markov property' for a control problem means that the future state only depends on the current state and the current action. Previous states or actions do not influence future states.

A MDP consists of 4 components that form a tuple of $\langle S, A, P, r \rangle$. Here S is the set of valid states, and A is the set of possible actions. P is the transition probability to the next state s' given current state s and action a:

$$P = \mathbb{P}\left(s' \mid s, a\right) \tag{4-1}$$

r is the reward from state s and action a:

$$r = R\left(s, a\right) \tag{4-2}$$

If a control problem has constraints it is modelled as a Constrained Markov Decision Process (CMDP). A CMDP tuple has 5 components, namely $\langle S, A, P, r, b \rangle$. $b(s, a) \in [0, \infty\rangle$ is the cost of the constraint.

4-2-2 State and action space

State

The set of states S contains all valid states s for a given system. State s describes the entire environment in a real valued vector or matrix.

It practice it not always possible to make a complete description of the system and its environment. Instead, the state is only an observation of the environment that is gathered from cameras or other sensors.

Action space

The action space set A contains all possible actions for a given state and can be discrete or continuous. A system with a discrete action space has a finite amount of valid actions for each state. In practice, it is more common for a system to have a continuous action space wfere -even if the action space has upper and lower bounds- there are an infinite amount of valid actions in between for each state. Think of the steering wheel of a car; it can not just be turned left or right but it can be turned at every angle within the steering limits.

4-2-3 Reward, return and discount

The agent learns from the feedback provided by the environment. The feedback from the environment is called the reward r and determines how good or bad the current state and action is. The reward as a function of the state s and the action a:

$$r = R(s, a) \tag{4-3}$$

There are multiple ways to define the best performance given the reward r. The most common objective is to maximize the discounted return:

$$R = \sum_{t=0}^{\infty} \gamma^t r_{t+1} = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots \tag{4-4}$$

Discount rate $\gamma \in [0, 1]$ provides a measure of how the agent should try to maximize the return. If $\gamma = 0$, the agent will only care about the reward of the current state and behave very short-sighted. If $\gamma = 1$, the agent will try to maximize all future rewards, even if it means receiving a lower reward in the current state. Typically values close to 1 are used.

4-2-4 Agent and policy

In RL the actions a are executed by an agent. The agent selects actions based on the policy π. The terms policy and agent are often used interchangeably. The policy defines an action a for every state s:

$$\pi = \pi(a \mid s) \tag{4-5}$$

4-2-5 Episodes

RL agents are often trained in episodes. Episodes are started in the initial states and ended when a terminal state or time limit is reached. Randomness in the initial state helps to explore new solutions and create a more generalized policy. Terminal states can be used to prevent the agent from exploring states that are not useful for the learning process, for example when it is too far from the target. A time limit can be used to avoid wasting time when the agent is stuck in the environment.

Episodes are also called trajectories or rollouts and are denoted with τ:

$$\tau = s_0, a_0, s_1, a_1, s_2, a_2 \cdots \tag{4-6}$$

4-2-6 Objective and optimal policy

The goal of the policy is to pick the actions that result in the highest return. The objective is described as maximizing the expected return $J(\pi)$:

$$J(\pi) = \underset{\tau \sim \pi}{\mathrm{E}} \left[R(\tau) \right] \tag{4-7}$$

The optimal policy π^* is the policy that gives the highest $J(\pi)$:

$$\pi^* = \arg \max_{\pi} J(\pi) \tag{4-8}$$

4-2-7 Value

Almost all RL algorithms try to approximate the value of the current state and the available actions. These are the value V and action-value Q.

Value V^π is the expected return from state s and all following states s' if the actions are taken from policy π:

$$V^\pi(s) = \underset{\tau \sim \pi}{\mathrm{E}} \left[R(\tau) \mid s \right] \tag{4-9}$$

The optimal value V^* is the expected return from state s and all following states s' if the actions are taken from the optimal policy π^*:

$$V^*(s) = \max_{\pi} \underset{\tau \sim \pi}{\mathrm{E}} \left[R(\tau) \mid s \right] \tag{4-10}$$

The action-value Q^π is the expected return from state s and all following states s' for any action in s and actions from policy π in all following states s':

$$Q^\pi(s, a) = \underset{\tau \sim \pi}{\mathrm{E}} \left[R(\tau) \mid s, a \right] \tag{4-11}$$

The optimal action-value Q^* is the expected return from state s and all following states s' for any action in s and actions from the optimal policy π^* in all following states s':

$$Q^*(s, a) = \max_{\pi} \underset{\tau \sim \pi}{\mathrm{E}} \left[R(\tau) \mid s, a \right] \tag{4-12}$$

The optimal policy will choose the action in state s that results in the highest return; which is the action with the highest Q value:

$$a^* = \arg \max_{a} Q^*(s, a) \tag{4-13}$$

4-2-8 Deep RL

In deep RL the different functions are approximated by neural networks with parameters θ or ϕ. θ and ϕ are the weights of the nodes in the network and are initialized randomly and optimized in the training process.

The policy network with parameters ϕ:

$$\pi(s) \rightarrow \pi_{\phi}(s) \tag{4-14}$$

Value networks V and Q with parameters θ:

$$V(s) \rightarrow V_{\theta}(s) \tag{4-15}$$

$$Q(s,a) \rightarrow Q_{\theta}(s,a) \tag{4-16}$$

4-2-9 Target networks and moving averages

When the values θ of network $Q_{\theta}(s,a)$ are updated with the information from $Q_{\theta}(s',a')$ it is likely that the two will be very similar. If the network is updated after every interaction it can lead to catastrophic interference, where the network can not properly relate the actions to good or bad results. In practice the learning is more stable is the network is kept constant for a little longer before updating the parameters.

This can be done by storing the updates in a separate network called the target network $\bar{Q}_{\theta}(s,a)$. θ then is copied from the target network every n steps:

$$Q_{\theta}(s,a) \leftarrow \bar{Q}_{\theta}(s,a) \tag{4-17}$$

Alternatively a moving average $\bar{\theta}$ can stabilize the learning.

4-2-10 Experience replay

Consecutive experiences $\langle s, a, s', r \rangle$ in the environment can be highly correlated. The correlation can cause a bias in the learned policy to a certain order of events. Ideally, the training data should be Independent and Identically Distributed (IID). To create IID sets the $\langle s, a, s', r \rangle$ experiences are stored in a replay buffer $\mathcal{D}$ and then sampled in random order from the buffer later.

4-3 Key RL algorithms

For a good understanding of the Soft Actor-Critic (SAC) and Lyapunov Safe Actor-Critic (LSAC) algorithms that are used in this work, it is beneficial to go through some of the key preceding RL algorithms. RL algorithms can be divided into two main categories; Q-learning and Policy optimization. Some combine the strengths of both methods in a single algorithm.

4-3-1 Q-learning

The optimal action at state s is defined as the action with the highest Q value (Equation 4-13). Q-learning methods [15, 82, 83] focus on learning $Q(s, a)$ to determine the policy.

In the basic versions of Q-learning $Q_{(s,a)}$ is a lookup table [82] with a row for every state and a column for every action. The values in the table are updated with the update rule:

$$Q^{new}(s, a) \leftarrow Q(s, a) + \alpha \left(r + \gamma \max_a Q(s_{t+1}, a) - Q(s, a) \right) \tag{4-18}$$

Here learning rate α determines the relative weight of the new values. Representing Q as a table is only feasible for control problems with small discrete state and action spaces. It can not be scaled up for very large or continuous state and action spaces.

In Deep Q-learning [15, 83] the value of $Q(s, a)$ is approximated with a neural network to solve problems with large or continuous state and action spaces:

$$Q(s, a) \rightarrow Q_\theta(s, a) \tag{4-19}$$

$Q_\theta(s, a)$ is initialized randomly and θ is updated by gradient descent minimizing a loss function:

$$L(\theta) = \left(Q(s, a) - \left(r + \gamma \max_a Q(s', a') \right) \right)^2 \tag{4-20}$$

4-3-2 Policy optimization

Instead of approximating the action-value function $Q(s, a)$, policy optimization methods [79, 80, 81] focus on optimizing the policy directly. The policy is approximated by a neural network with parameters ϕ:

$$\pi(s) \rightarrow \pi_\phi(s) \tag{4-21}$$

The values of ϕ are optimized via gradient ascent on the objective $J(\pi_\phi)$:

$$J(\pi_\phi) = \mathbb{E}[R(s, a)] \tag{4-22}$$

The gradient of $J(\pi_\phi)$ can be written as:

$$\nabla_\phi \mathbb{E}[R(s, a)] = \mathbb{E}[\nabla_\phi \log \pi_\phi(a \mid s) R(s, a)] \tag{4-23}$$

The stochastic policy gradient update is then:

$$\phi_{t+1} = \phi_t + \alpha R_{t+1} \nabla_\phi \log \pi_{\phi_t}(a_t \mid s_t) \tag{4-24}$$

Equation 4-24 shows that the gradient ∇_ϕ is scaled with the reward R_{t+1}. This means if the reward is high, the likelihood of those actions is increased. Likewise, if the return is low, the likelihood of those actions is decreased.

4-4 Soft Actor-Critic (SAC)

4-4-1 Introduction

Soft Actor-Critic (SAC) is a state-of-the-art off-policy learning algorithm for continuous control problems. Off-policy means that the policy can be updated with experiences from previous iterations of the policy. Off-policy algorithms are considered sample efficient because all available experiences $\langle s, a, s', r \rangle$ can be used to improve the policy. In contrast, on-policy methods can only use experiences from the current iteration of the policy.

There are several iterations of SAC, each with slight differences in its implementation [52, 53]. In this work SAC as presented in [53] is used. SAC is a combination of policy optimization (Actor) and deep Q-learning (Critic).

4-4-2 Entropy regularization

SAC introduces the maximum entropy principle for RL by adding an entropy term to the standard objective function. Maximizing the entropy at each state encourages exploration of new states and actions. This addresses a downside of policy gradient methods that can get stuck on a local maximum by finding one promising solution and not exploring outside that solution. In practice, this leads to more exploration and in turn faster learning [84, 85, 86].

The standard objective for RL is to find a policy that maximizes the expected return (Equations 4-7 and 4-8):

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{(s,a)\sim\rho_\pi} \left[R\left(\tau\right)\right] \tag{4-25}$$

A policy with the maximum entropy objective simultaneously maximizes the expected return and the entropy $\mathcal{H}$:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{(s,a)\sim\rho_\pi} [r(s,a) + \alpha\mathcal{H}(\pi(\cdot \mid s))] \tag{4-26}$$

Here α is called the temperature parameter and it is used to tune the balance between exploring (entropy) and exploiting the current policy (expected return).

4-4-3 Soft Actor-Critic

SAC uses neural networks to approximate the policy and the Q-values. The policy network models the policy with the mean and covariance of a multivariate Gaussian distribution.

The policy network with parameters ϕ:

$$\pi\left(s\right) \to \pi_\phi\left(a \mid s\right) \tag{4-27}$$

The approximations of the Q-functions tend to overestimate the actual Q-values. To overcome this SAC uses two Q-functions Q_1 and Q_2 and at every step, the lowest value of both is used.

Value networks Q_1 and Q_2 with parameters θ_1 and θ_2:

$$Q_1\left(s,a\right) \rightarrow Q_{\theta_1}\left(s,a\right) \tag{4-28}$$

$$Q_2\left(s,a\right) \rightarrow Q_{\theta_2}\left(s,a\right) \tag{4-29}$$

The objective for the parameters of the Q-function is defined as $J_Q(\theta)$:

$$J_Q(\theta) = \mathbb{E}_{(s_t,a_t)\sim\mathcal{D}}\left[\frac{1}{2}\left(Q_\theta\left(s_t,a_t\right) - \left(r\left(s_t,a_t\right) + \gamma\mathbb{E}_{s_{t+1}\sim p}\left[V_{\bar{\theta}}\left(s_{t+1}\right)\right]\right)\right)^2\right] \tag{4-30}$$

where the value function $V(s_t)$ is given by:

$$V(s_t) = \mathbb{E}_{a_t\sim\pi}\left[Q(s_t,a_t) - \alpha\log\pi(a_t \mid s_t)\right] \tag{4-31}$$

θ is optimized with stochastic gradient ascent. The gradient of $J_Q(\theta)$ is approximated by $\hat{\nabla}_\theta J_Q(\theta)$:

$$\hat{\nabla}_\theta J_Q(\theta) = \nabla_\theta Q_\theta(a_t,s_t)\left(Q_\theta\left(s_t,a_t\right) - \left(r\left(s_t,a_t\right) + \gamma\left(Q_{\bar{\theta}}\left(s_{t+1},a_{t+1}\right) - \alpha\log\left(\pi_\theta\left(a_{t+1},a_{t+1}\right)\right)\right)\right)\right) \tag{4-32}$$

$\bar{\theta}$ is an exponential moving average of θ to stabilize the learning process, see Section 4-2-9.

The policy is updated by minimizing the Kullback–Leibler divergence:

$$J_\pi(\phi) = \mathbb{E}_{s_t\sim\mathcal{D}}\left[\mathbb{E}_{a_t\sim\pi_{phi}}\left[\alpha\log\left(\pi_\phi\left(a_t \mid s_t\right)\right) - Q_\theta(s_t,a_t)\right]\right] \tag{4-33}$$

The gradient of $J_\pi(\phi)$ can be approximated by $\hat{\nabla}_\phi J_\pi(\phi)$:

$$\hat{\nabla}_\phi J_\pi(\phi) = \nabla_\phi\alpha\log\left(\pi_\theta\left(a_t \mid s_t\right)\right) + \left(\nabla_{a_t}\alpha\log\pi_\theta\left(a_t \mid s_t\right)\right) - \nabla_{a_t}Q\left(s_t,a_t\right)\right)\nabla_\phi f_\phi\left(\epsilon_t;s_t\right) \tag{4-34}$$

In Equation 4-34 the policy π_ϕ is expressed as $f_\phi\left(\epsilon_t;s_t\right)$ for lower variance, see [53] for more details:

$$a_t = f_\phi\left(\epsilon_t;s_t\right) \tag{4-35}$$

with ϵ is input noise.

The algorithm is summarized in Table 4-1.

Table 4-1: SAC algorithm overview

```
SAC
Input: $\theta_1, \theta_2, \phi$                                   ▷ Initial parameters
  $\bar{\theta_1} \leftarrow \theta_1, \bar{\theta_2} \leftarrow \theta_2$        ▷ Initialize target network weights
  $\mathcal{D} \leftarrow 0$                                      ▷ Initialize empty replay pool
    for each iteration do
        for each environment step do
            $a_t \sim \pi_\phi(a_t \mid s_t)$                      ▷ Sample action from the policy
            $s_{t+1} \sim p(s_{t+1} \mid s_t, a_t)$              ▷ Sample transition from the environment
            $\mathcal{D} \leftarrow \mathcal{D} \cup (s_t, a_t, r(s_t, a_t), s_{t+1})$   ▷ Store the transition in the replay pool
        end for
        for each gradient step do
            $\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)$ for $i \in 1, 2$   ▷ Update Q parameters
            $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_{\phi_i} J_\pi(\phi)$          ▷ Update policy weights
            $\alpha \leftarrow \alpha - \lambda \hat{\nabla}_\alpha J(\alpha)$              ▷ Adjust temperature
            $\bar{\theta_i} \leftarrow \tau\theta_i + (1-\tau)\bar{\theta_i}$ for $i \in 1, 2$   ▷ Update target network weights
        end for
    end for
Output: $\theta_1, \theta_2, \phi$                                  ▷ Optimized parameters
```

4-5 Lyapunov Safe Actor-Critic (LSAC)

4-5-1 Introduction

It is difficult to predict the behavior of a (trained) RL policy. The main objective for a policy is typically defined as maximizing the discounted return, which means optimizing the behavior for the long term. As a consequence, it may take bad decisions now because it expects a benefit from it in the future. For safety-critical tasks such as automated driving, this may not be the right approach. A single bad decision can have fatal consequences.

Lyapunov Safe Actor-Critic (LSAC) [49][50] introduces a new way to address the safety of RL policies with a theoretical guarantee of stability. LSAC is a combination of Lyapunov stability analysis with the SAC algorithm. The proposed Lyapunov analysis can also be combined with other algorithms as CPO (LCPO) and PPO (LPPO).

This chapter provides a broad overview of the principles of the LSAC algorithm, for the proof and derivations of the stability theory the reader is referred to the work of Han et al. in [49].

4-5-2 Stability and Lyapunov analysis

Definition of stability

The stability of a system is its ability to recover to an equilibrium position from (external) disturbances [87]. The definition of stability addressed by LSAC is called *ultimately uniformly bounded* (UUB) stability.

A UUB system will stay inside a set of safe states. If the system is disturbed or started outside the safe set it will move back to the safe set in finite time. The safe set of states must have a non-zero size; if the size of the safe set approaches 0 the system must always return to the origin. In that case, the system is said to have *mean square stability* (MSS). For an automated car, the UUB limits can be the edges of the road; a UUB vehicle will not leave the road and if it is pushed off the road it will drive back on in finite time T.

The UUB definition is valid for systems bounded with limit $d > 0$, and $\forall\ d \in (0, c)$ there is a finite time $T = T(d, \underline{d})$ independent of t_0 [49]:

$$\mathbb{E}_{s_0} b_\pi(s_0) \leq d \rightarrow \mathbb{E}_{s_t} b_\pi(s_t) \leq \underline{d} \quad \forall t \geq t_0 + T \tag{4-36}$$

Here b_π is the expected undiscounted safety cost from the constraint under policy π:

$$b_\pi(s) = \mathbb{E}_{a\sim\pi} b(s, a) \tag{4-37}$$

The UUB principle from Equation 4-36 is visualized in Figure 4-1.

Lyapunov analysis

Lyapunov functions are used in control theory to analyze the stability of dynamical systems around the equilibrium position. A Lyapunov function L must be a continuously differentiable scalar semi-positive definite function; $L : \mathcal{S} \rightarrow \mathbb{R}^+$.

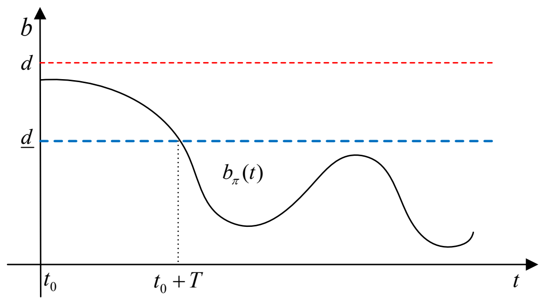


Figure 4-1: Visualization of the UUB principle presented in Equation 4-36.

Consider the following system:

$$\dot{x} = f(x) \tag{4-38}$$

with Lyapunov function $L(x)$ and its time derivative $\dot{L}(x)$:

$$\dot{L}(x) = \frac{d}{dt} L\left(x(t)\right) = \frac{\partial L}{\partial x} \cdot \frac{dx}{dt} = \nabla L \cdot \dot{x} = \nabla L \cdot f(x) \tag{4-39}$$

If $\dot{L}(x)$ for a state trajectory τ is semi-negative definite, the system will move down the slope of $L(x)$ in finite time reach a state where $L = 0$ and thus be stable.

UUB guarantee for RL

Consider a system with the safe set of states Ω constrained by $\bar{d}$:

$$\Omega = \left\{ s \in \mathcal{S} \mid b_\pi(s) < \bar{d} \right\} \tag{4-40}$$

If the system is UUB there must exist a set of Lyapunov functions $L(s)$ and positive constants α_1, α_2 and α_3 for all states in safe set Ω:

$$\alpha_1 b_\pi(s) \leq L(s) \leq \alpha_2 b_\pi(s) \tag{4-41}$$

For the states outside the safe set Ω in the edge set Δ

$$\Delta = \left\{ s \in \mathcal{S} \mid b_\pi(s) \geq \eta, \eta \in [0, \alpha_2^{-1} \alpha_1 \bar{d}] \right\} \tag{4-42}$$

must hold:

$$\mathbb{E}_{s \sim \tau} \left(\mathbb{E}_{s' \sim \pi} L(s') - L(s) \right) \leq -\alpha_3 \mathbb{E}_{s \sim \tau} b_\pi(s) \tag{4-43}$$

Since the constraint is only active in the edge set Δ the agent is free to explore to the edges of safe set Ω.

4-5-3 Lyapunov Safe Actor-Critic

The objective function for the LSAC policy is:

$$\begin{aligned} J(\pi) = \mathbb{E}_{(s,a,s')\sim\mathcal{D}}[\beta[\log(\pi_\theta(f_\theta(\epsilon,s) \mid s))] - Q(s, f_\theta(\epsilon,s))] \\ + \mathbb{E}_{(s,a,s',b)\sim\mathcal{D}_e}[\lambda(L_c(s', f_\theta(\epsilon,s')) - L_c(s,a) + \alpha_3 b(s,a))] \end{aligned} \quad (4\text{-}44)$$

where $\mathcal{D}$ stores the environment transitions in safe set Ω and $\mathcal{D}_e$ stores the transitions in the edge set Δ.

Two Q functions (Q_1, Q_2) with target networks (Q'_1, Q'_2) are used (as in SAC) to stabilize the learning process, see Section 4-2-9.

The gradient of the objective function $J(\pi)$ is then:

$$\begin{aligned} \nabla_\theta J(\pi) = \mathbb{E}_{\mathcal{D}}[\nabla_\theta \beta \log(\pi_\theta(a \mid s)) + \nabla_a(\beta \log(\pi_\theta(a \mid s)) - \min_i Q_i(s,a))\nabla_\theta f_\theta(\epsilon', s)] \\ + \mathbb{E}_{\mathcal{D}e}[\lambda \nabla_{a'} L_c(s', a')\nabla_\theta f_\theta(\epsilon, s')] \end{aligned} \quad (4\text{-}45)$$

Again, like SAC the value function is augmented with the entropy to improve exploration:

$$V(s) = \mathbb{E}_{a\sim\pi}(Q(s,a) - \beta \log(\pi(a \mid s))) \quad (4\text{-}46)$$

The objective for the Q-functions J_{Q_1} and J_{Q_2} is to minimize:

$$J_Q(\theta) = \mathbb{E}_{(s_t,a_t)\sim\mathcal{D}}[\frac{1}{2}(Q_\theta(s_t,a_t) - (r(s_t,a_t) + \gamma\mathbb{E}_{s_{t+1}\sim\rho}[V_{\bar{\theta}}(s_{t+1})]))^2] \quad (4\text{-}47)$$

The Lyapunov critic L_c:

$$L_c(s) = \mathbb{E}_{a\sim\pi} L_c(s,a) \quad (4\text{-}48)$$

The objective function for the lyapunov critic is given by:

$$J(L_c) = \mathbb{E}_{(s,a)\sim D}[\frac{1}{2}(L_c(s,a) - L_{\text{target}}(s,a))^2] \quad (4\text{-}49)$$

Here L_{target} is the target for L_c defined by the cost of constraint:

$$L_{target}(s,a) = c(s,a) \quad (4\text{-}50)$$

The tuning parameters β (temperature) and λ (lagrangian) have the following objectives:

$$J(\beta) = \mathbb{E}_{(s,a)\ \ D)} - \beta[\log(\pi_\theta a \mid s) + H_t] \quad (4\text{-}51)$$

$$J(\lambda) = \mathbb{E}_{(s,a)\sim\mathcal{D}} - \lambda \left[L_c\left(s', f_\theta(\epsilon, s')\right) - L_c(s,a) + \alpha_3 c(s,a)\right] \quad (4\text{-}52)$$

The LSAC algorithm is summarized in Table 4-2.

Table 4-2: LSAC algorithm overview

```
LSAC
Input: $\phi_Q$, $\phi_{L_c}$, $\theta$                                    ▷ Initial parameters
  $\bar{\phi}_Q \leftarrow \phi_Q, \bar{\phi}_{L_c} \leftarrow \phi_{L_c}, \bar{\theta} \leftarrow \theta$        ▷ Initialize target network weights
  $\Delta$                                                      ▷ Initialize edge set
  $\lambda$                                                      ▷ Initialize Lagrangian multiplier
  $\mathcal{R}$, $\mathcal{R}_c$                                              ▷ Initialize replay and edge replay pool
    for each iteration do
      for each environment step do
        $a_t \sim \pi_\phi(a_t \mid s_t)$                          ▷ Sample action from the policy
        $s_{t+1} \sim p(s_{t+1} \mid s_t, a_t)$                    ▷ Sample transition from environment
        $\mathcal{R} \leftarrow \mathcal{R} \cup (s_t, a_t, r(s_t, a_t), b, s_{t+1})$      ▷ Store transition in replay pool $\mathcal{R}$
        if $s_t \in \Delta$:
        $\mathcal{R}_c \leftarrow \mathcal{R}_c \cup (s_t, a_t, r(s_t, a_t), b, s_{t+1})$    ▷ Store transition in replay pool $\mathcal{R}_c$
      end for
      for each gradient step do
        $\phi_Q \leftarrow \phi_Q + \alpha_{\phi_Q} \nabla_{\phi_Q} J(Q)$                 ▷ Update Q parameters
        $\phi_{L_c} \leftarrow \phi_{L_c} + \alpha_{\phi_{L_c}} \nabla_{\phi_{L_c}} J(L_c)$           ▷ Update $L_c$ parameters
        $\theta \leftarrow \theta + \alpha_\theta \nabla_\theta J(\pi)$                     ▷ Update policy weights
        $\lambda \leftarrow \max(0, \lambda + \alpha_\lambda L_\lambda)$                   ▷ Update Lagrange multiplier
        $\beta \leftarrow \beta + \alpha_\beta \nabla_\beta J(\beta)$                      ▷ Adjust temperature

        $\bar{\phi}_Q \leftarrow \tau \phi_Q + (1 - \tau) \bar{\phi}_Q$                   ▷ Update the target networks
        $\bar{\phi}_{L_c} \leftarrow \tau \phi_{L_c} + (1 - \tau) \bar{\phi}_{L_c}$
        $\bar{\theta} \leftarrow \tau \theta + (1 - \tau) \bar{\theta}$
      end for
    end for
Output: $\phi_Q$, $\phi_{L_c}$, $\theta$                                   ▷ Optimized parameters
```

Chapter 5

Training Setup and Interaction with the Environment

5-1 Introduction

This chapter discusses the general settings of the RL algorithms such as the state observations, network structures, and hyperparameters. It also covers how rewards can be designed to get the desired behavior. Details that are specific to a single experiment are discussed in Chapter 6.

5-2 OpenAI GYM

The vehicle simulator for the agent to train in is built as a part of OpenAI's Gym [54]. Gym is an open-source toolkit to test and compare learning algorithms on a library of standard tasks. These tasks include balancing cart poles and pendulums, and a collection of Atari video games. After each action A, the environment returns the next state S' and reward r back to the learning algorithm. The environment is reset after a terminal state is reached or a time limit per episode is exceeded.

5-3 Hyperparameters

The hyperparameters for SAC are set as suggested by the authors of SAC in [53]. The hyperparameters for LSAC are identical to SAC, with the addition of learning rate for the Lyapunov critic and α_3. The hyperparameters are the same for all experiments. The values are summarized in Table 5-1.

Table 5-1: Hyperparameters used for the SAC and LSAC algorithms. The hyperparameters are set as suggested by the authors of SAC in [53].

Parameter	Value
learning rate Actor	$1{\cdot}10^{-4}$
learning rate Critic	$3{\cdot}10^{-4}$
learning rate Lyapunov Critic (LSAC only)	$3{\cdot}10^{-4}$
α_3 (LSAC only)	0.8
discount(γ)	0.99
replay buffer size	$1{\cdot}10^{6}$
number of hidden layers	2
number of hidden units per layer	256
number of samples per minibatch	256
target entropy	-1
target smoothing coefficient (τ)	0.005
target update interval	1
gradient steps	1

5-4 Observed state

The state vector (observation) of the environment contains a selection of vital vehicle states. To make a fair comparison between RL and MPC the state does not contain any information that MPC does not have access to. The state vector includes the 19 states from Table 5-2. States 1-8 are the vehicle motion and path tracking errors. States 9-19 provide a preview of the upcoming road segment over 1 second, with 0.1 second intervals ($k_{0.1}, k_{0.2}, ...k_{1.0}$). The road preview is visualized in Figure 5-1.

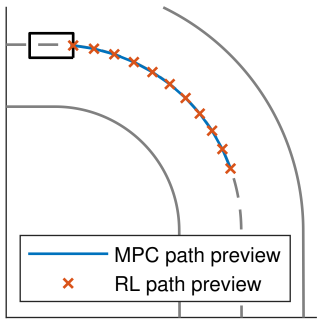


Figure 5-1: The observation of the environment and preview of the target path as seen by the MPC and RL controllers. Over the preview horizon of 1 second, the MPC has 50 preview points, the state vector for the RL algorithms only has 10 preview points.

5-5 Neural networks structure

The neural networks used in this work are multilayer perceptron (MLP) networks with 2 hidden layers. Each hidden layer has 256 nodes with a non-linear activation function. Each

Table 5-2: The state vector observed by the Reinforcement Learning algorithms for each environment step. States 1-8 describe the motion of the vehicle and path tracking errors to the target. States 9-19 are the preview of the upcoming road segment over the next second, with 0.1 second intervals ($k_{0.1}, k_{0.2}, ...k_{1.0}$).

State	Description
u	Longitudinal velocity
v	Lateral velocity
r	Yaw rate
e	Lateral error
$\Delta\phi$	Heading error
a_x	Longitudinal acceleration
a_y	Lateral acceleration
δ	Steering angle
k_c	Current curvature
$k_{0.1}$	Curvature at t+0.1 s
$k_{0.2}$	Curvature at t+0.2 s
$k_{0.3}$	Curvature at t+0.3 s
$k_{0.4}$	Curvature at t+0.4 s
$k_{0.5}$	Curvature at t+0.5 s
$k_{0.6}$	Curvature at t+0.6 s
$k_{0.7}$	Curvature at t+0.7 s
$k_{0.8}$	Curvature at t+0.8 s
$k_{0.9}$	Curvature at t+0.9 s
$k_{1.0}$	Curvature at t+1.0 s

node is fully connected to every node in the next layer. The input for the policy network π_ϕ is the observed state from Section 5-4 and the output is mean μ_ϕ and standard deviation σ_ϕ. The output of the value networks Q_{θ_i} are the Q values, and the output of the Lyapunov network L_θ for LSAC is the Lyapunov value.

5-6 Reward design

5-6-1 Introduction

The reward defines the objective of the task by providing feedback after every action. The feedback from the environment ultimately determines what the optimal behavior is to achieve the highest score. The reward also has a significant influence on training times. Finding a solution can take unreasonably long when rewards are sparse. It is good practice to shape the rewards to give continuous feedback on the policy, rather than only at the end of the episode [88].

If the rewards are not considered carefully, the resulting policy can be found to exhibit undesirable behavior, or try to 'cheat' the system to get the highest return [89]. For example; in training, it was found that the highest reward for trajectory following could be obtained by purposefully slowing the vehicle down by excessive steering.

Therefore designing the reward can be a tedious task, and it is often an iterative process of observing and retraining until the desired behavior is reached. Recent discoveries on automated reward design [90] have shown great potential but it is outside the scope of this work.

5-6-2 Survival bonus

The general design for the rewards in this work is to give a positive reward for every step taken. This means that the highest score can be achieved by staying alive and avoiding terminal states. Bad behavior is penalized by subtracting a cost from the survival bonus, so that the optimal policy still results in the desired behaviour. A reward function to minimize some error a with a survive bonus of 1:

$$r = 1 - a_{\text{error}} \tag{5-1}$$

The benefit of the survival bonus is that the optimal policy will avoid terminal states. This can also be a drawback; where instead of going for the target directly, a higher score can be obtained by getting close to the target but never reaching it and thus finishing the episode.

5-6-3 Scaling and weighing rewards

When multiple objectives are considered it is easier to compare the relative weights if the rewards are scaled to be the same order of magnitude. In this work the rewards will generally be scaled to be $\in [-1, 1]$. Relative importance of objectives are weighed with factor ω. For example:

$$r = 1 - \omega_1 \left(\frac{a_{\text{error}}}{a_{\text{limit}}}\right) - \omega_2 \left(\frac{b_{\text{error}}}{b_{\text{limit}}}\right) \tag{5-2}$$

5-6-4 Effect of exponential rewards on vehicle behavior

The tracking error from the target path in this work is denoted by $|e|$. A good reward function to minimize the tracking error could be $1 - |e|^x$. It was found that the exponent x in the reward had a significant effect on the resulting policy. Agents trained with $|e|^2$ or $|e|$ did not converge to the origin of $|e| = 0$ but overshot and oscillated around it. On the other hand, a very steep penalty around the origin such as $|e|^{0.2}$ resulted in a stable, non-oscillating solution. This is visualized in Figure 5-2.

5-6-5 Rewarding smooth steering behaviour

RL control policies can exhibit pretty erratic and noisy behavior. An erratic input on vehicle steering applications is uncomfortable and wasteful of energy; it should be as smooth as possible.

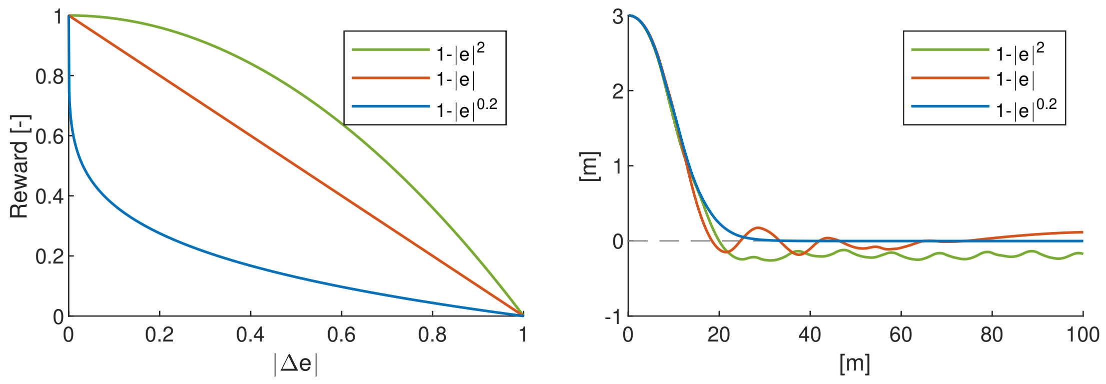


Figure 5-2: Different exponentials of the reward function of lateral error $|e|$ (left) vs the resulting vehicle behavior (right). The vehicle shows stable convergence for rewards that are steep around the origin. Rewards that are less strict around the origin result in policies with oscillating behaviour.

With only the lateral error $|e|$ in the reward, the steering actions are constantly jumping between extremes. Penalizing the difference in steering angle between steps $(\Delta\delta)$ did not calm the steering down, even with the previous control input added to the observed state. Instead, for an increasing penalty on $\Delta\delta$ only the performance of the main objective (path tracking) got worse. This agrees with the findings in [44].

The best results for the straight road sections in this work were found by penalizing the absolute steering angle $|\delta|$. This discourages the agent from performing unnecessary control inputs. See Figure 5-3.

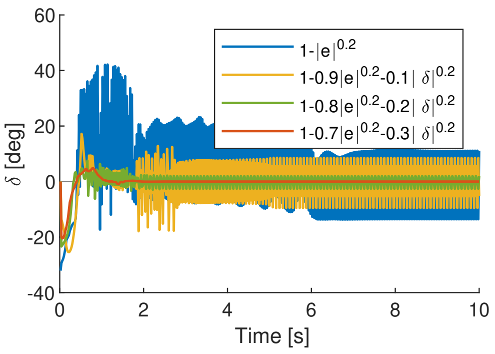


Figure 5-3: Steering behaviour for different reward functions. Policies with lower penalties on the steering angle $|\delta|$ show very volatile steering actions. Penalizing steering angle $|\delta|$ results in a smooth control policy.

For experiments in more complex environments, the steering cost was found less effective. The benefit of slowing the vehicle down by wild steering inputs was too beneficial for the task of path tracking. The best solution was found by scaling all rewards by the vehicle velocity. This indirectly rewards the smoother control policies that do not slow the vehicle down. Other methods for creating smooth control policies [44, 91] have been proposed but are outside the scope of this work and incompatible with the SAC and LSAC algorithms.

5-7 Running agent and environment at different rates

5-7-1 Drawbacks of a fast updating environment

The vehicle simulator updates with a frequency of 1000 Hz. This is necessary for the stable integration of the equations of motion. Running the simulator at lower frequencies would give numerical instability or otherwise less accurate results. However, running the control policy at such high frequencies can be very undesirable for several reasons.

First of all, for every second of driving in the simulation environment, the agent will have to take 1000 actions and the environment will return 1000 new states and rewards. This means 1000 $[s, a, s', r]$ transitions and policy updates on the GPU. This is computationally very expensive and thus time-consuming. Even on a modern GPU, every second of simulation takes about 9 seconds of real-world time. This negates one of the main benefits of training in a virtual environment; namely speed.

Another reason why such a fast control policy is undesirable is that a solution can be found that relies on controlling the system at such a high frequency. This was found in Atari video games where control policies trained with reinforcement learning had a superhuman performance by applying a different action every frame [92]. For vehicle driving and many other simulated tasks, it is not necessary to change the input every millisecond for a good control policy. Many physical systems will not even respond at that rate.

Lastly, in slower-paced control tasks such as strategy games, the effects of an action tend to take a while to become apparent. It was found that the agents are quicker to relate current actions to long term results if the policy does not get unnecessary updates that provide little new information [93]. In a way, automated driving is a slower-paced control task as well because the steering on corner entry can make a difference on corner exit, and updates at 1000 Hz do not necessarily provide new information.

5-7-2 Solution

A solution in video games can be to not return every rendered frame to the agent. Instead, each action by the agent is held constant for several frames, and only after those frames have passed a single update is given back. This is sometimes referred to as *frameskip*.

In this work, a similar approach is used. Every steering action of the agent is held constant for 20 steps (0.02 [s]) in the simulation environment before an update is returned. The control policy thus runs at 50 Hz while the simulation can run at 1000 Hz. 50 Hz is chosen because the MPC runs at 50 Hz as well, giving a fair comparison.

5-8 Agent selection

In practice it is found that agents trained on the same task in the same environment with the same rewards can find different solutions. This can be found in the real world as well, where racing drivers can obtain similar lap times with very different driving styles.

For each task, at least five agents are trained with the same settings. After evaluation, only the best agent with the desired behavior is used in the final results. The training for an agent is restarted if it gets stuck for too long at a local minimum.

Chapter 6

Experiments and Results

6-1 Introduction

In this chapter MPC, SAC and LSAC are tested in a vehicle simulator at a range of different driving tasks. For every task, the methods of training the RL agents are given, including rewards, safety costs, and training times.

6-1-1 Supplementing videos

Videos of the experiments and simulations are available on YouTube on the channel 'LSACautomatedvehicles', https://www.youtube.com/channel/UCaoRIfyEY-rA7fcTdSxM8sA/. It is strongly suggested to watch the videos together with the report, it gives a much clearer insight into the behavior and performance of the controllers than the static figures in the report.

6-1-2 List of experiments

The controllers are tested in two different environments and cover a range of different driving tasks.

The first experiments will be done on a straight piece of road with 5 lanes. Each lane has a width of 3 meters. The driving speeds in the experiments cover urban, rural, and highway driving scenarios. The experiments in this environment are:

- Navigating through traffic at 50, 75 and 100 km/h (Section 6-3)
- Recovering from destabilizing impacts (Section 6-4)

The rest of the experiments will be done on a race track to provide a more complex driving challenge. The track has a length of 565 meters and a constant width of 10 meters. The target velocity for each corner on the track is defined by Equation 2-43 and ranges between $\sim$40 and $\sim$100 km/h. The experiments on the track cover the following driving scenarios:

- Driving on a race track and maintaining vehicle stability (Section 6-5)
- Safe navigation on a race track and maintaining vehicle stability (Section 6-6)
- Finding the racing line on a race track inside track limits (Section 6-7)

An overview of the environments can be found in Figure 6-1:

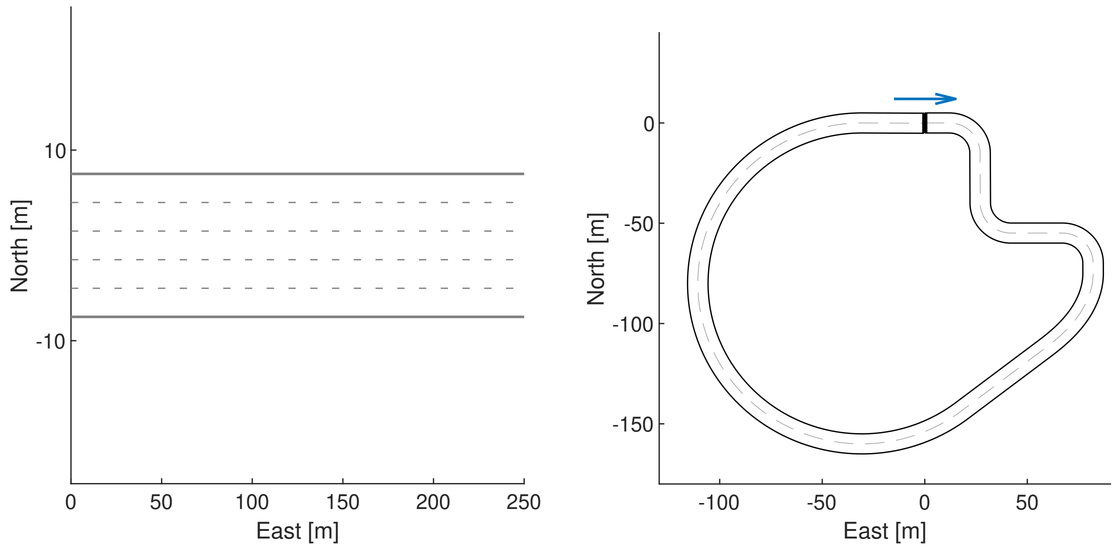


Figure 6-1: Overview of the 5-lane road (left) and the race track (right) environments that are used to evaluate the performance of the different controllers. The lanes have a width of 3 meters, the race track has a length of 565 meters and a constant width of 10 meters.

6-1-3 Target velocity V_{target} and friction utilisation μ_{des}

In every experiment the vehicle will try to follow the target speed V_{target}. The target speed will be expressed as $\sqrt{\mu_{\text{des}}} \cdot V_{\text{max}}$ where $\mu_{\text{des}} \in [0, 1]$ is the desired utilization of the maximum available friction. For $\mu_{\text{des}} = 0.75$ the target speed V_{target} is such that the vehicle will use 75% of the available tire-road friction. The derivation and background of V_{max}, V_{target} and μ_{des} can be found in Section 2-5-3.

How the target velocity around the race track changes for different values of μ_{des} is shown in Figure 6-2.

6-1-4 Figure notations

To show dynamic vehicle behavior in static images the results of the experiments are visualized using the notations from Figure 6-3. Each control strategy has its own color; MPC is blue, SAC is orange and LSAC is yellow. The trajectories of the vehicle are drawn with a thin, light coloured line. When the vehicle is unstable the trajectory is drawn with a thick, darker colored line. The vehicle is unstable when the yaw rate limit or the lateral velocity limit from Section 2-7-1 is exceeded. The vehicles are plotted at a 300% scale on the race track for better visibility.

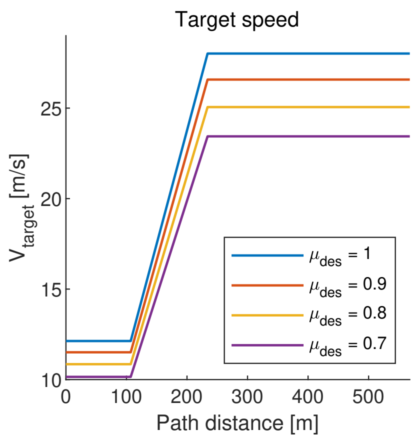


Figure 6-2: Target velocity V_{target} for different values of the desired friction utilization μ_{des} around the race track.

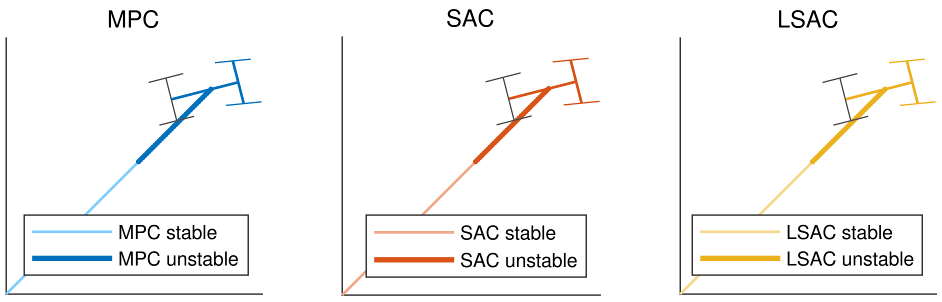


Figure 6-3: This figure is a reference for the colors and notation used to display the results of the experiments. The vehicles and trajectories are blue for MPC, orange for SAC, and yellow for LSAC. The steering wheels are colored, the driven wheels are black. The vehicles in the track environment are plotted at a 300% scale for better visibility. The thin line sections represent the traveled trajectory. The overlaying thick line sections indicate that the vehicle is in an unstable state as defined in Section 2-7-1. That means that either the yaw rate limit or the lateral velocity limit is exceeded.

6-2 Training agents for the straight environment

6-2-1 Introduction

This section explains how SAC and LSAC agents can be trained to follow a trajectory on a 5-lane road section. The goal is to minimize the distance to the target path. The performance of the controllers is tested in two different experiments:

- Navigating through traffic at 50, 75 and 100 km/h (Section 6-3)
- Recovering from destabilizing impacts (Section 6-4)

The first experiment is a test of the general driving abilities under normal circumstances. The second experiment tests the robustness of the controllers under an outside disturbance.

6-2-2 Training

Episodes

The SAC and LSAC agents are trained for 2e6 time steps each. This is around 11 hours of driving time in the simulation environment. The training episodes are time-limited to 8 seconds per episode, and then the environment is reset. There are roughly 5000 episodes in 2e6 time steps.

The vehicle is started in a different location and at a different speed at the beginning of each training episode. The goal is to move the car from the initial position to the center of the road as quickly as possible. The initial velocity is picked from $\in [50, 100]$ [km/h], the lateral error e from $\in [-4, 4]$ [m], and the heading error $\Delta\phi$ from $\in [-20, 20]$ [deg].

An episode is ended if $|e| > e_{\text{lim}}$ (10 m), $|\Delta\phi| > \Delta\phi_{\text{lim}}$ (90 deg) or the time limit (8 sec) is reached.

Reward and constraints

The SAC and LSAC agents are rewarded for minimizing tracking error $|e|$ and penalized for all steering actions $|\delta|$. This penalty on the steering is necessary to get a smoother steering policy, see Section 5-6-5. The reward is visualized in Figure 6-4. The reward drops sharply as e and δ move from the origin. The components of the reward are scaled with their respective limits so all components are in the same order of magnitude.

$$\text{reward } r(s,a) = 1 - 0.7\left(\frac{|e|}{e_{\text{lim}}}\right)^{0.2} - 0.3\left(\frac{|\delta|}{\delta_{\text{lim}}}\right)^{0.2} \tag{6-1}$$

Here the maximum lateral error e_{lim} is 10 [m] and the maximum steering angle δ_{lim} is 40 [deg].

Additionally, the LSAC agents are constrained with a safety cost to keep the vehicle within the stable limits of vehicle handling:

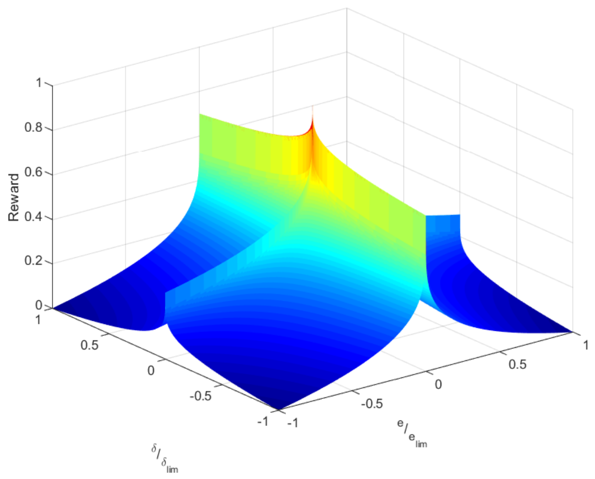


Figure 6-4: Visualization of the reward $r \in [0, 1]$ from Equation 6-1. The agents are rewarded for minimizing the tracking error e and penalized for steering actions δ. The reward drops sharply as e and δ move from the origin.

$$\text{safety cost } b(s, a) = \left(\frac{\max(0, |v - v_{\max}|)}{|v|} + \frac{\max(0, |r - r_{\max}|)}{|r|} \right) \tag{6-2}$$

Where $v_{\max}$ and $r_{\max}$ are the limits for stable driving for the lateral velocity v and yaw rate r respectively. The average reward and safety cost per episode during the training process can be seen in Figure 6-5.

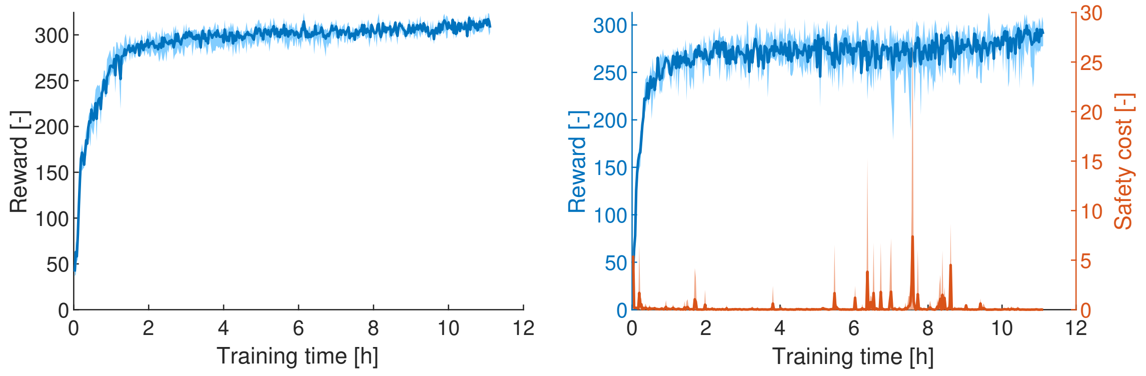


Figure 6-5: The average reward of the SAC (left) and LSAC (right) agents per episode during the training progress. The shaded area shows the variance between the different agents. The agents are rewarded for minimizing the tracking error e and penalized for steering actions δ (Equation 6-1). The safety costs for the LSAC agent are for violations of the stability constraint (Equation 6-2).

6-3 Navigating trough traffic

6-3-1 Introduction

The goal of this experiment is to change lanes on a 5 lane road every 50 meters to simulate navigating through traffic. The vehicle is started on the middle lane and every 50 meters the target moves to an adjacent lane. The controllers will be tested at low, medium, and high speeds. The first test is at 50 km/h to represent typical urban driving speeds. The second test is at 75 km/h to represent driving on rural roads and the third test is at 100 km/h to show the behavior at highway speeds.

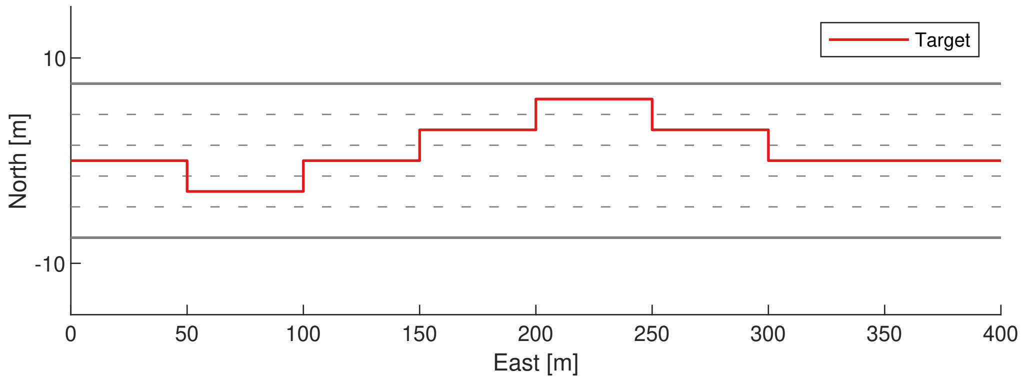


Figure 6-6: Target path for navigating through traffic on a 5 lane road. The target lane changes every 50 meters. The test is repeated for 3 different velocities; 50, 75 and 100 km/h.

6-3-2 Navigating through traffic at 50 km/h

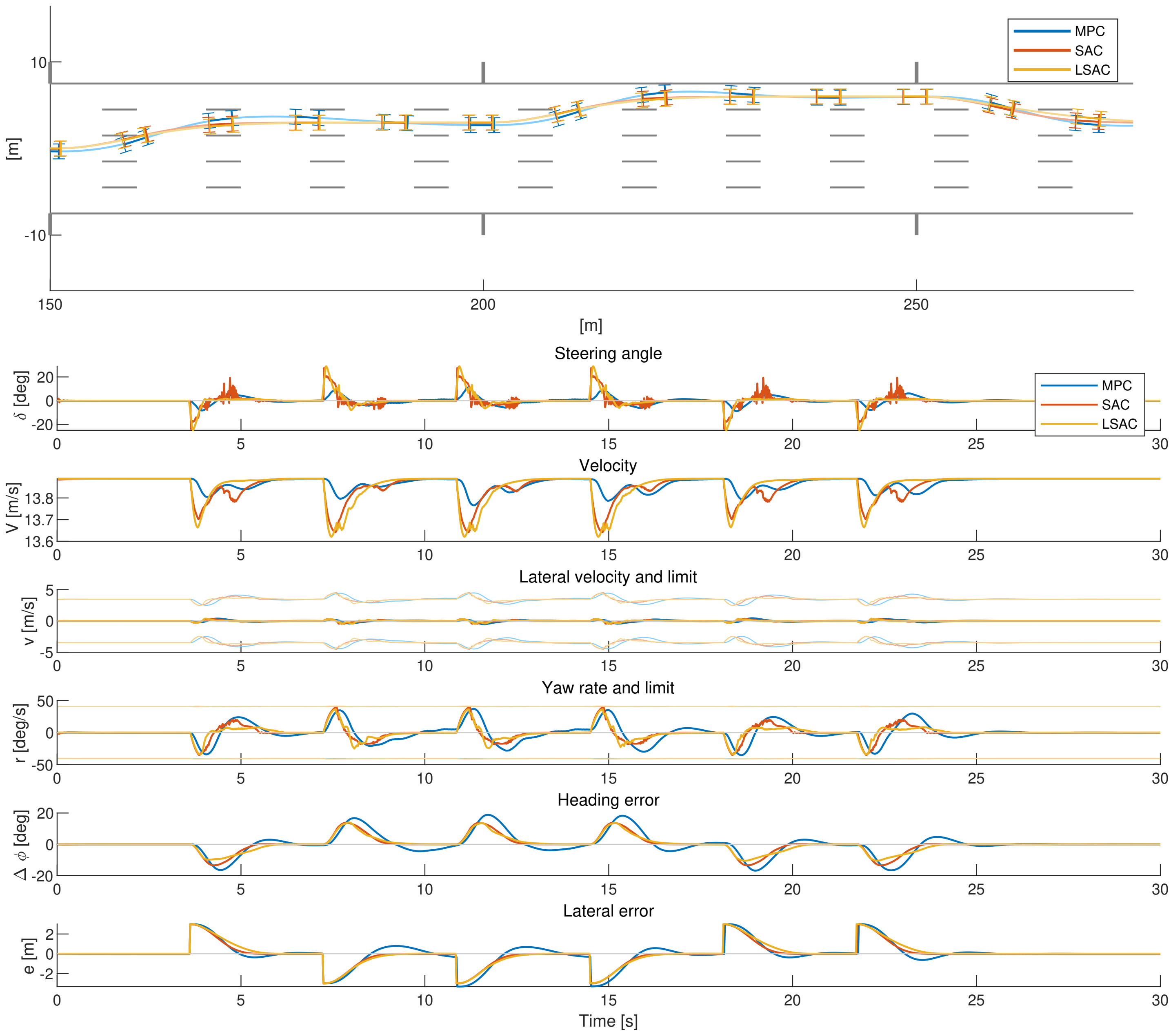


Figure 6-7: Top: Vehicle trajectories for vehicles changing lanes every 50 meters at 50 km/h. The thick line sections on the vehicle trajectory indicate that the vehicle is outside the stable region as defined in section 2-7-1, see Figure 6-3. Bottom: Steering actions and relevant vehicle states and limits during the experiment.

6-3-3 Navigating through traffic at 75 km/h

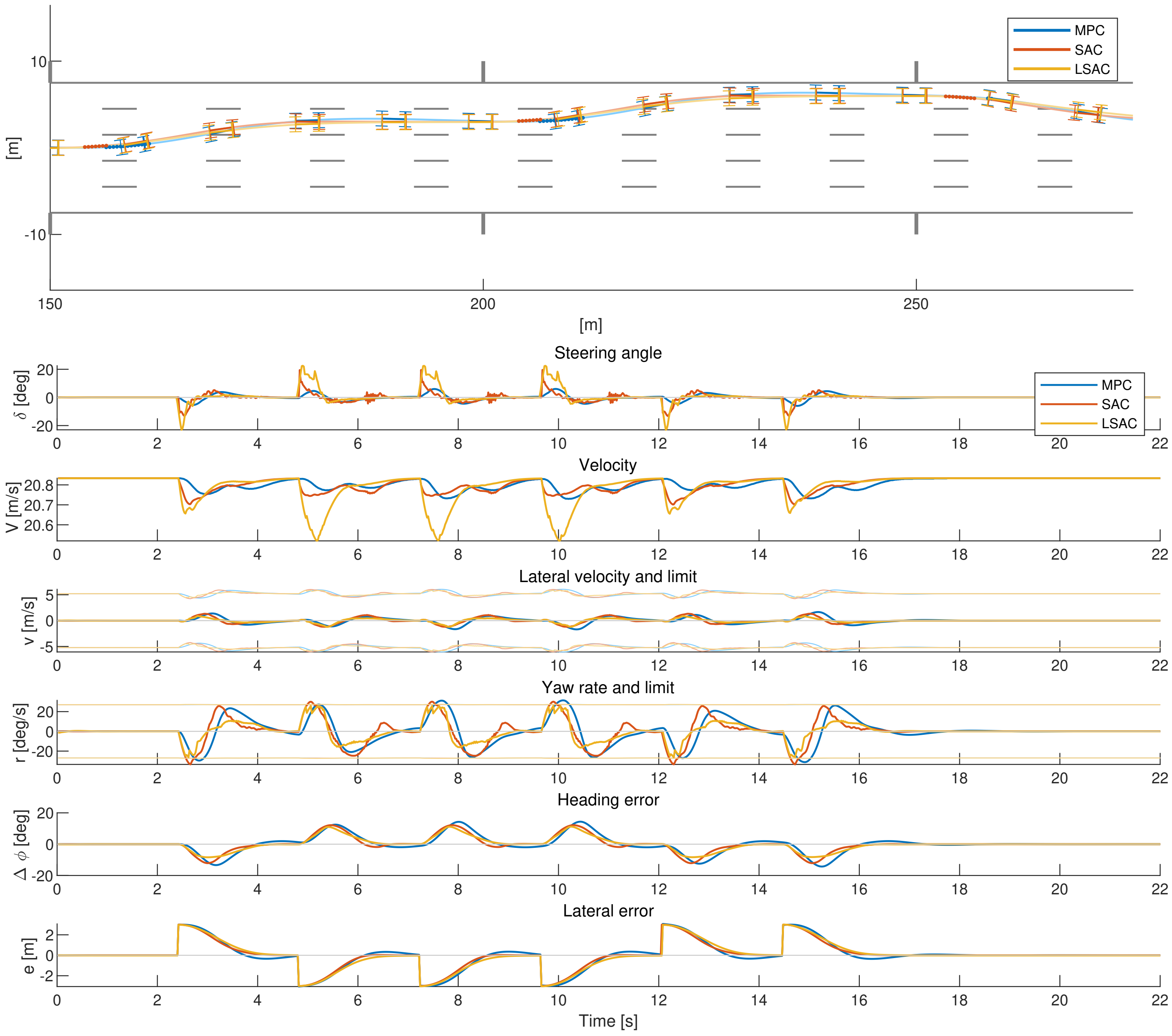


Figure 6-8: Top: Vehicle trajectories for vehicles changing lanes every 50 meters at 75 km/h. The thick line sections on the vehicle trajectory indicate that the vehicle is outside the stable region as defined in section 2-7-1, see Figure 6-3. Bottom: Steering actions and relevant vehicle states and limits during the experiment.

6-3-4 Navigating through traffic at 100 km/h

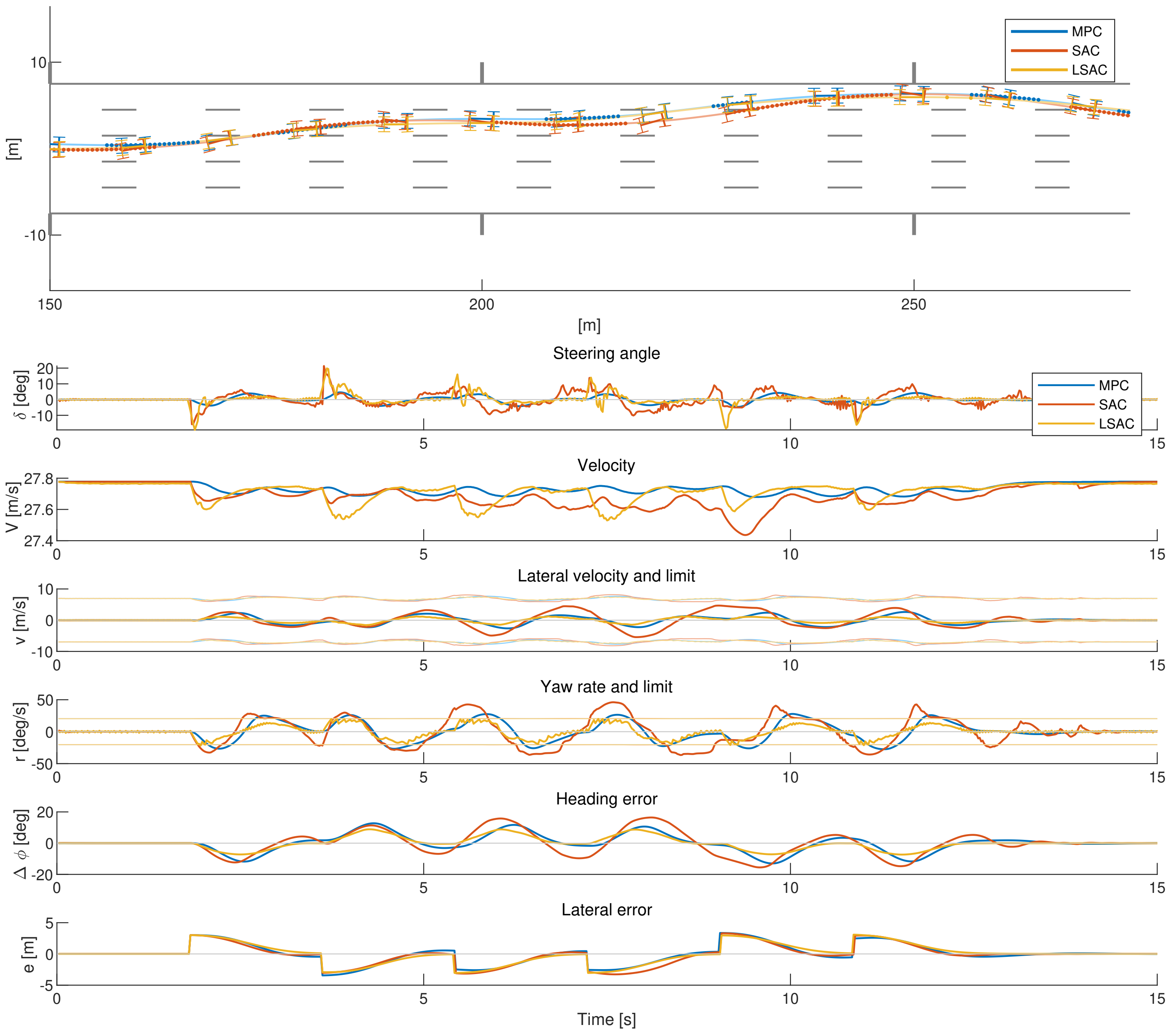


Figure 6-9: Top: Vehicle trajectories for vehicles changing lanes every 50 meters at 100 km/h. The thick line sections on the vehicle trajectory indicate that the vehicle is outside the stable region as defined in section 2-7-1, see Figure 6-3. Bottom: Steering actions and relevant vehicle states and limits during the experiment.

6-3-5 Discussion

The goal of this experiment was to move the vehicle to the center of the target lane, with the target lane changing every 50 meters. MPC and LSAC are constrained to keep the vehicle stable, SAC is only concerned with minimizing the tracking error.

The RL agents were trained for 11 hours in the simulator but the average reward per episode in Figure 6-5 shows that the agents are more or less 'fully trained' in less than 2 hours. After that, the average reward does not increase significantly from the remaining training. The spread in reward between the different agents is very small, suggesting that the training is stable and consistent to replicate.

The experiments show no significant difference between MPC, SAC, and LSAC at 50 and 75 km/h. In all tests, MPC has the smoothest steering inputs, whereas SAC and LSAC use much larger steering angles.

MPC overshoots the target slightly, mainly at the lower speeds. It is likely that this can be minimized with more careful tuning of the MPC settings.

At 100 km/h the difference between MPC, SAC, and LSAC is more pronounced. The vehicle is barely in the target lane before the target is moved again. Here MPC starts to show small stability violations of the yaw rate constraint; the vehicle is rotating too fast. SAC is even worse, showing large violations of the yaw rate constraint in almost every transition. Nevertheless, SAC does not lose control over the vehicle and the tracking performance is good.

In contrast, the LSAC agent never violates the stability constraints. Figure 6-9 shows how LSAC moves the vehicle right to the edges of stability but does not violate them.

Unstable driving can be dangerous and result in a loss of control. The next experiment will test how MPC, SAC, and LSAC recover from an impact that is designed to bring the vehicle in an unstable state.

6-4 Recovering from destabilizing impacts

6-4-1 Introduction

In this experiment, the vehicle will be hit by another vehicle on the rear axle. An impact perpendicular to the vehicle on the rear axle is particularly destabilizing; it tends to over-rotate the car and force a loss of control [94]. This experiment is designed to test the robustness of the controllers under an unpredicted outside disturbance. Dealing with disturbances is an important aspect of safe automated driving without relying on the driver to take over.

The vehicle will be driving on the target path with a velocity of 55 km/h. This is a typical speed in urban environments where such collisions may take place. The colliding vehicle has a mass of 1000 kg and the test is repeated for 4 different impact velocities between 12.6 and 18 km/h. The goal is to regain control over the vehicle and return to the original target path.

6-4-2 Simulating a collision

Simulating a collision between vehicles accurately can be very difficult, especially when all the deformations are considered. The collisions in this experiment are simplified because high fidelity is not required to test the vehicle response. The collision is modelled as a constant impact force F_{impact} over the time of collision t_{impact}. The average impact force F_{impact} is given as:

$$F_{\text{impact}} = \frac{\Delta V \cdot m}{t_{\text{impact}}} \tag{6-3}$$

Here ΔV and m are the change in velocity and mass of the colliding vehicle respectively. The colliding vehicle is a 1000 kg car that loses all its velocity (ΔV) in the collision over 0.5 seconds. The experiments are repeated for 4 different impact velocities of 12.6, 14.4, 16.2, and 18 km/h. This corresponds to average impact forces F_{impact} of 7kN, 8kN, 9kN, and 10kN.

The impact force is added to the sum of lateral forces F_y and torque T acting on the vehicle body from Equations 2-30 and 2-31, see Figure 6-10.

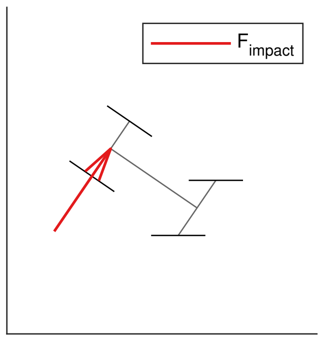


Figure 6-10: The application of the impact force F_{impact} on the rear axle of the vehicle to simulate a destabilizing collision.

The new sum of lateral forces F_y and body torque T:

$$F_y = F_{y,fl} + F_{y,fr} + F_{y,rl} + F_{y,rr} + F_{\text{impact}} \tag{6-4}$$

$$T = (F_{y,fl} + F_{y,fr})l_f - (F_{y,rl} + F_{y,rr} + F_{\text{impact}})l_r + (F_{x,fl} - F_{x,fr} + F_{x,rl} - F_{x,rr})\frac{1}{2}\text{B} \tag{6-5}$$

6-4-3 Recovering from destabilizing impact: impact velocity = 12.6 km/h

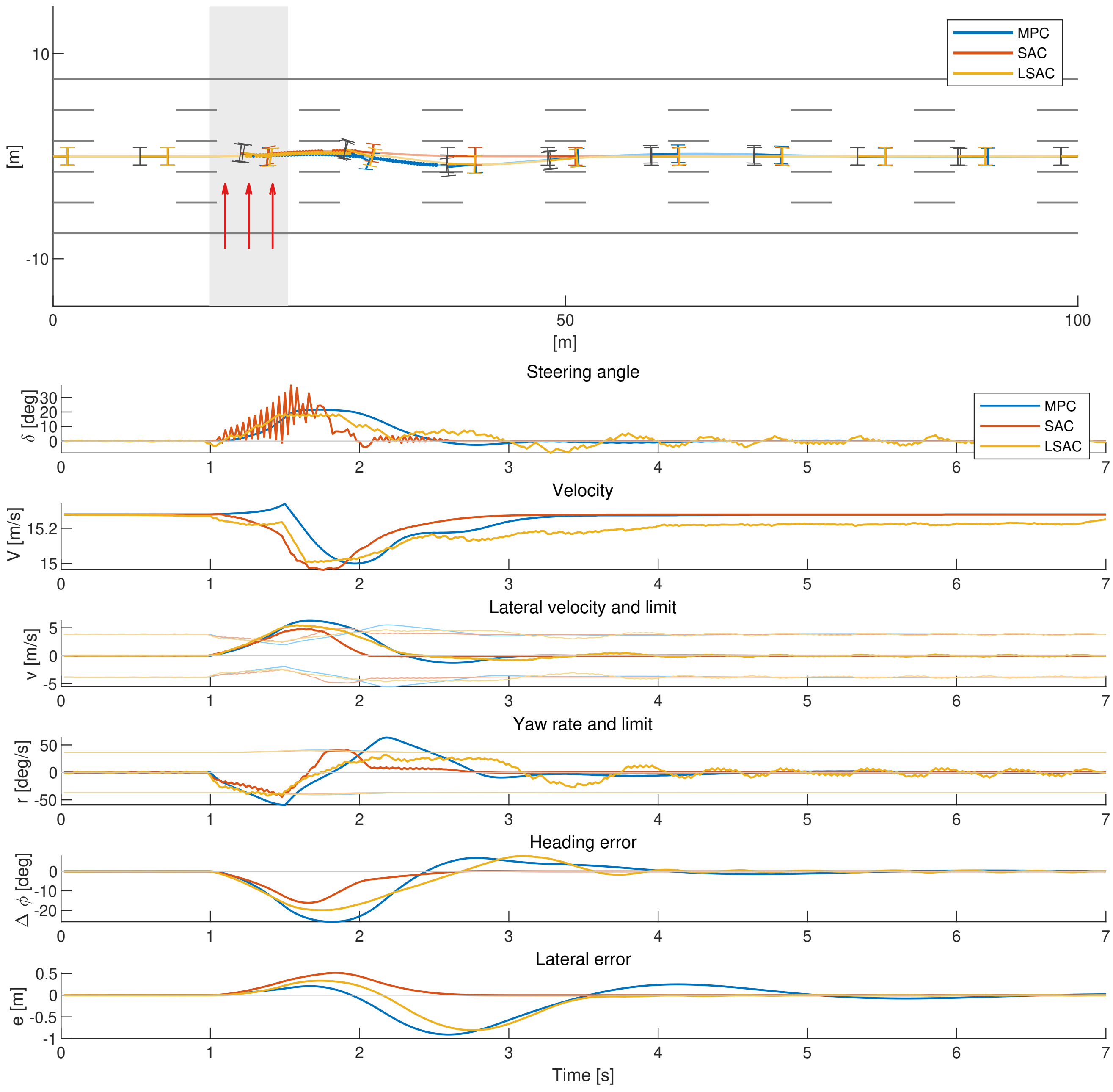


Figure 6-11: Top: Vehicle trajectories after an impact perpendicular on the rear axle while the vehicle is traveling at 55km/h. The impact is from t = 1 [s] to t = 1.5 [s] and is marked by the shaded area. The impacts represent a 1000 kg vehicle colliding on the rear of the vehicle and losing all its speed in the process from an initial speed of 12.6 km/h. The average impact force F_{impact} is 7kN. The thick line sections indicate that the vehicle is in an unstable state as defined in section 2-7-1, see Figure 6-3. Bottom: Steering actions and relevant vehicle states and limits during the experiment.

6-4-4 Recovering from destabilizing impact: impact velocity = 14.4 km/h

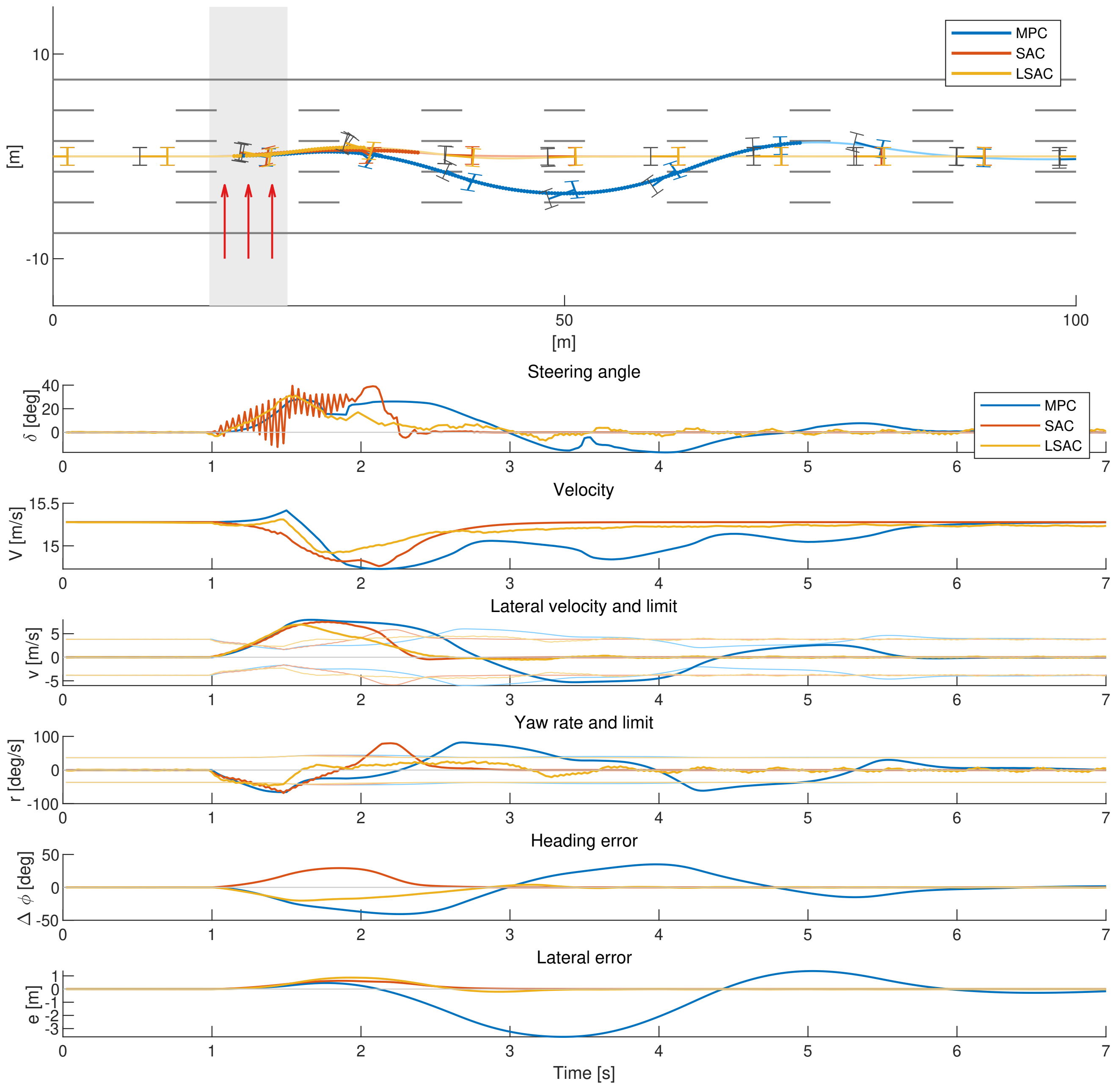


Figure 6-12: Top: Vehicle trajectories after an impact perpendicular on the rear axle while the vehicle is traveling at 55km/h. The impact is from t = 1 [s] to t = 1.5 [s] and is marked by the shaded area. The impacts represent a 1000 kg vehicle colliding on the rear of the vehicle and losing all its speed in the process from an initial speed of 14.4 km/h. The average impact force F_{impact} is 8kN. The thick line sections indicate that the vehicle is in an unstable state as defined in section 2-7-1, see Figure 6-3. Bottom: Steering actions and relevant vehicle states and limits during the experiment.

6-4-5 Recovering from destabilizing impact: impact velocity = 16.2 km/h

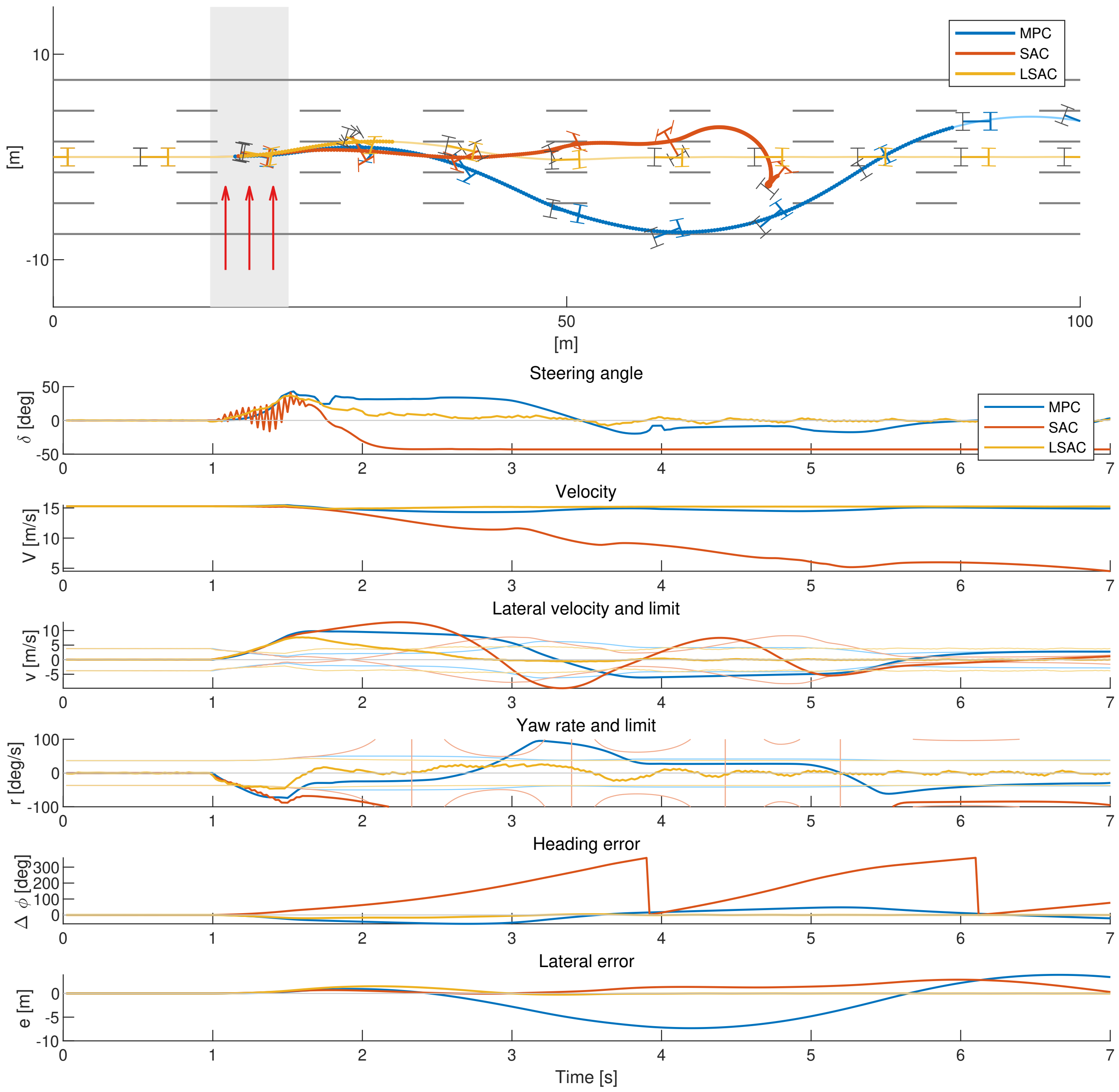


Figure 6-13: Top: Vehicle trajectories after an impact perpendicular on the rear axle while the vehicle is traveling at 55km/h. The impact is from t = 1 [s] to t = 1.5 [s] and is marked by the shaded area. The impacts represent a 1000 kg vehicle colliding on the rear of the vehicle and losing all its speed in the process from an initial speed of 16.2 km/h. The average impact force F_{impact} is 9kN. The thick line sections indicate that the vehicle is in an unstable state as defined in section 2-7-1, see Figure 6-3. Bottom: Steering actions and relevant vehicle states and limits during the experiment.

6-4-6 Recovering from destabilizing impact: impact velocity = 18.0 km/h

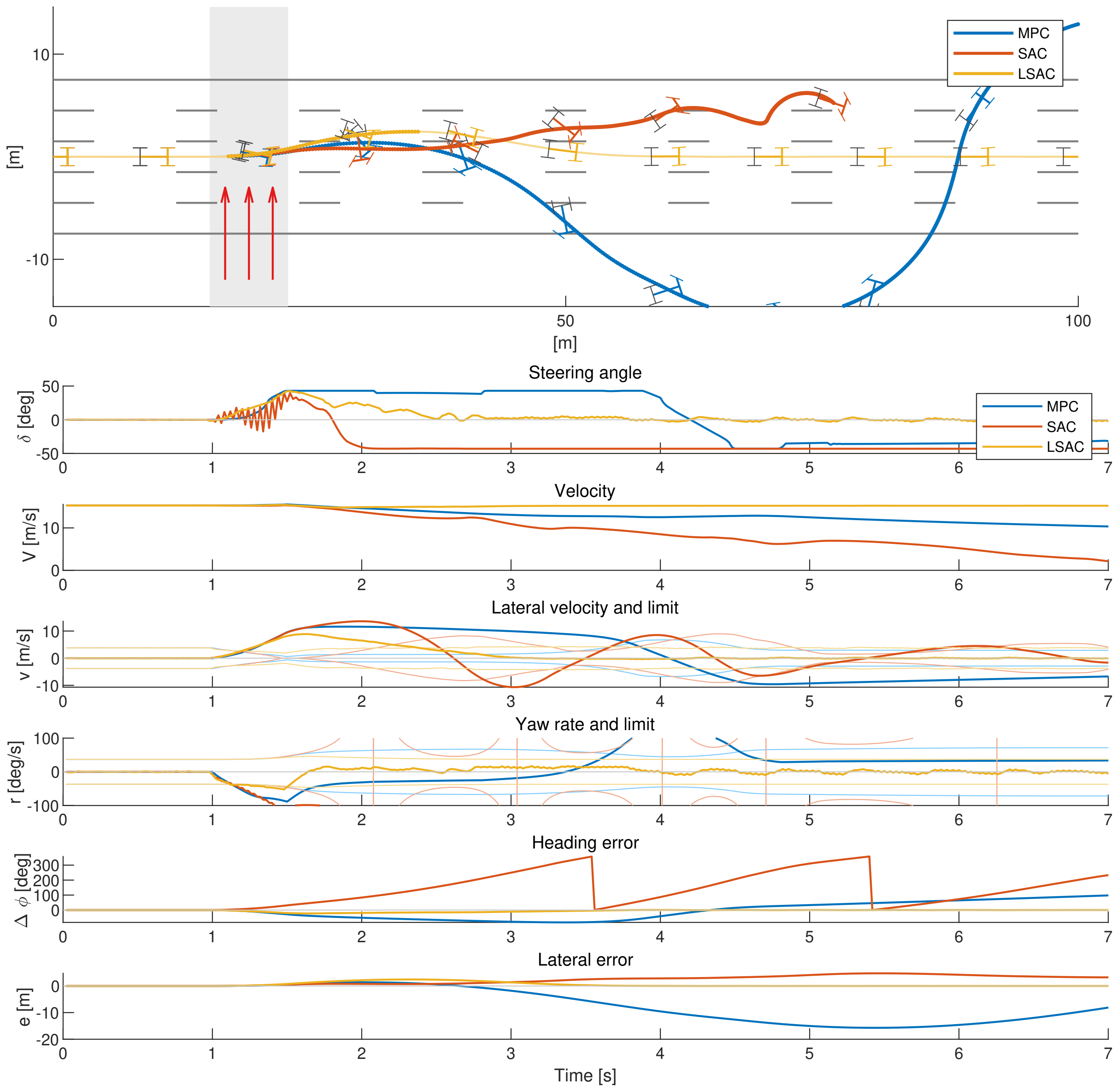


Figure 6-14: Top: Vehicle trajectories after an impact perpendicular on the rear axle while the vehicle is traveling at 55km/h. The impact is from t = 1 [s] to t = 1.5 [s] and is marked by the shaded area. The impacts represent a 1000 kg vehicle colliding on the rear of the vehicle and losing all its speed in the process from an initial speed of 18.0 km/h. The average impact force F_{impact} is 10kN. The thick line sections indicate that the vehicle is in an unstable state as defined in section 2-7-1, see Figure 6-3. Bottom: Steering actions and relevant vehicle states and limits during the experiment.

6-4-7 Discussion

In this experiment, the controllers were tested at their ability to regain control over the vehicle after a destabilizing collision. It shows the importance of vehicle stability and why it is necessary to control the vehicle near the limits of friction. Like in the previous experiment MPC and LSAC are constrained to keep the vehicle stable, SAC is not.

Even though SAC is not constrained on vehicle stability it deals fairly well with the lower impact velocities; it can recover from the 12.6 and 14.4 km/h impacts. However, for larger impact velocities, it fails and the vehicle starts spinning.

Arguably MPC does perform better than SAC. The constraints help to keep the vehicle from spinning so it eventually returns to the target path. The downside is that the vehicle slides all over the road and eventually even outside.

LSAC outperforms both MPC and SAC. LSAC starts counter steering sooner to slow down the rotation, even before the vehicle is unstable. As a result the vehicle never exceeds the stable yaw rate limits, while MPC and SAC violate the yaw rate limits in almost every experiment. Once the vehicle is back inside the stable zone it does not leave it again. Because the vehicle is stable sooner LSAC can keep closest to the original path as well.

6-5 Path tracking at the friction limit with vehicle stability

6-5-1 Introduction

The goal in this task is to drive around a race track at the limit of friction. The target path to follow is the centerline of the track, see Figure 6-15. The track has a length of 565 meters and a constant width of 10 meters.

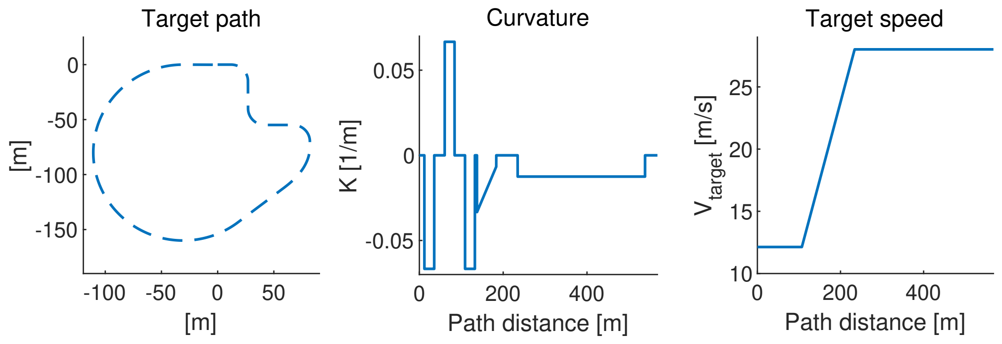


Figure 6-15: Target path (left), curvature (middle) and speed profile (right) of the track used in the experiments.

MPC and LSAC are constrained to keep the vehicle stable in addition to following the target path, SAC is not.

6-5-2 Training

Episodes

The SAC and LSAC agents are trained for 3e6 time steps in the environment, which translates to a total driving time of around 16 hours. A full lap around the track at racing speed takes about 35 seconds. So in the 16 hours of training the car can do roughly 1700 laps around the track.

The vehicle is started with a random target velocity the beginning of each training episode. The target velocity is picked from $\mu_{\text{des}} \in [0.70, 0.85]$, the initial lateral error e is picked from $\in [-0.1, 0.1]$ [m] and the initial heading error $\Delta\phi$ from $\in [-10, 10]$ [deg].

The track is randomly mirrored to be clockwise or counterclockwise at the start of each episode, see Figure 6-16. This is to make sure the policy does not converge to a solution that would only work in one direction.

An episode is ended if $|e| > e_{\text{lim}}$ (10 m), $|\Delta\phi| > \Delta\phi_{\text{lim}}$ (90 deg) or the end of the lap is reached ($s > 565$ [m]).

Rewards and constraints

The agents are rewarded for minimizing path tracking error $|e|$ and minimizing unnecessary steering actions δ that result in slowing the vehicle down:

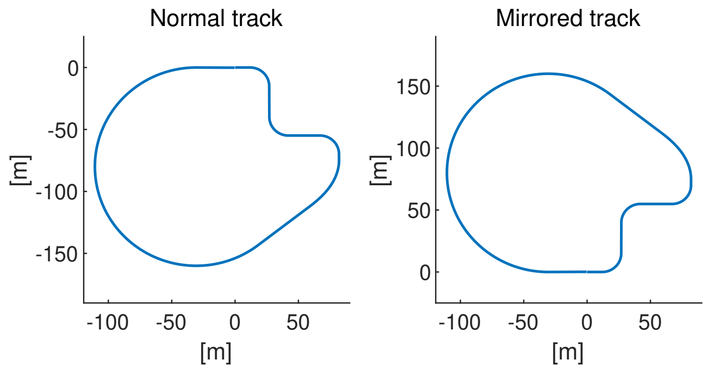


Figure 6-16: At the beginning of each training episode the race track is either generated normally and clockwise (left) or mirrored and counterclockwise (right) to provide some randomness in the environment. This makes sure the policy does not converge to a solution that only works in one direction.

$$\text{reward } r(s,a) = 1 - 0.7\left(\frac{|e|}{e_{\text{lim}}}\right)^{0.2} - 0.3\left(\frac{|\delta|}{\delta_{\text{lim}}}\right)^{0.2} \tag{6-6}$$

Where the maximum lateral error e_{lim} is 10 [m] and the maximum steering angle δ_{lim} is 40 [deg]. Additionally the LSAC agents receive a safety cost for violating the stability constraints:

$$\text{safety cost } b(s,a) = \left(\frac{\max(0, |v - v_{\text{max}}|)}{|v|} + \frac{\max(0, |r - r_{\text{max}}|)}{|r|}\right) \tag{6-7}$$

v_{max} and r_{max} are the stable driving limits for the lateral velocity v and yaw rate r (Section 2-7-1). The average reward and safety cost per training episode for the SAC and LSAC agents can be seen in Figure 6-17.

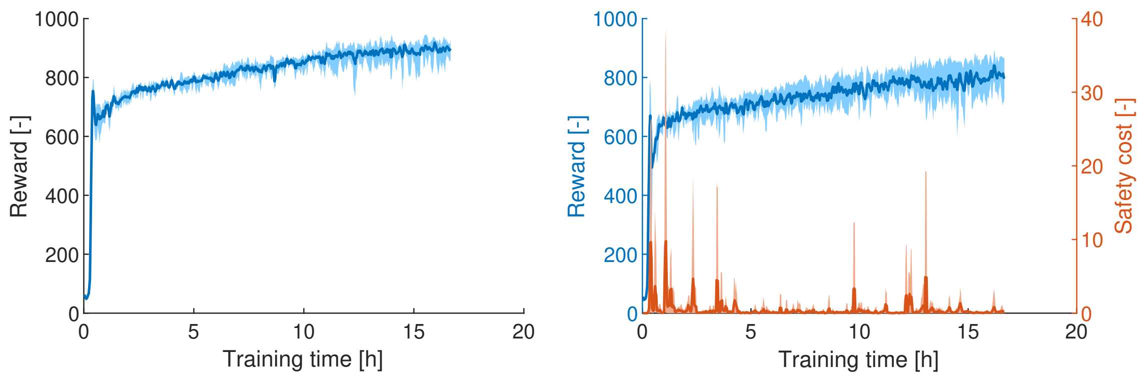


Figure 6-17: The solid blue lines are the average reward of the SAC (left) and LSAC (right) agents during the training progress. The shaded area shows the variance between the different agents. The agents are rewarded for minimizing the tracking error e and penalized for steering actions δ (Equation 6-6). The safety costs for the LSAC agent are for violations of the stability constraint (Equation 6-7).

6-5-3 Path tracking at 80% friction utilisation: $V_{\textbf{target}} = \sqrt{0.80} \cdot V_{\textbf{max}}$

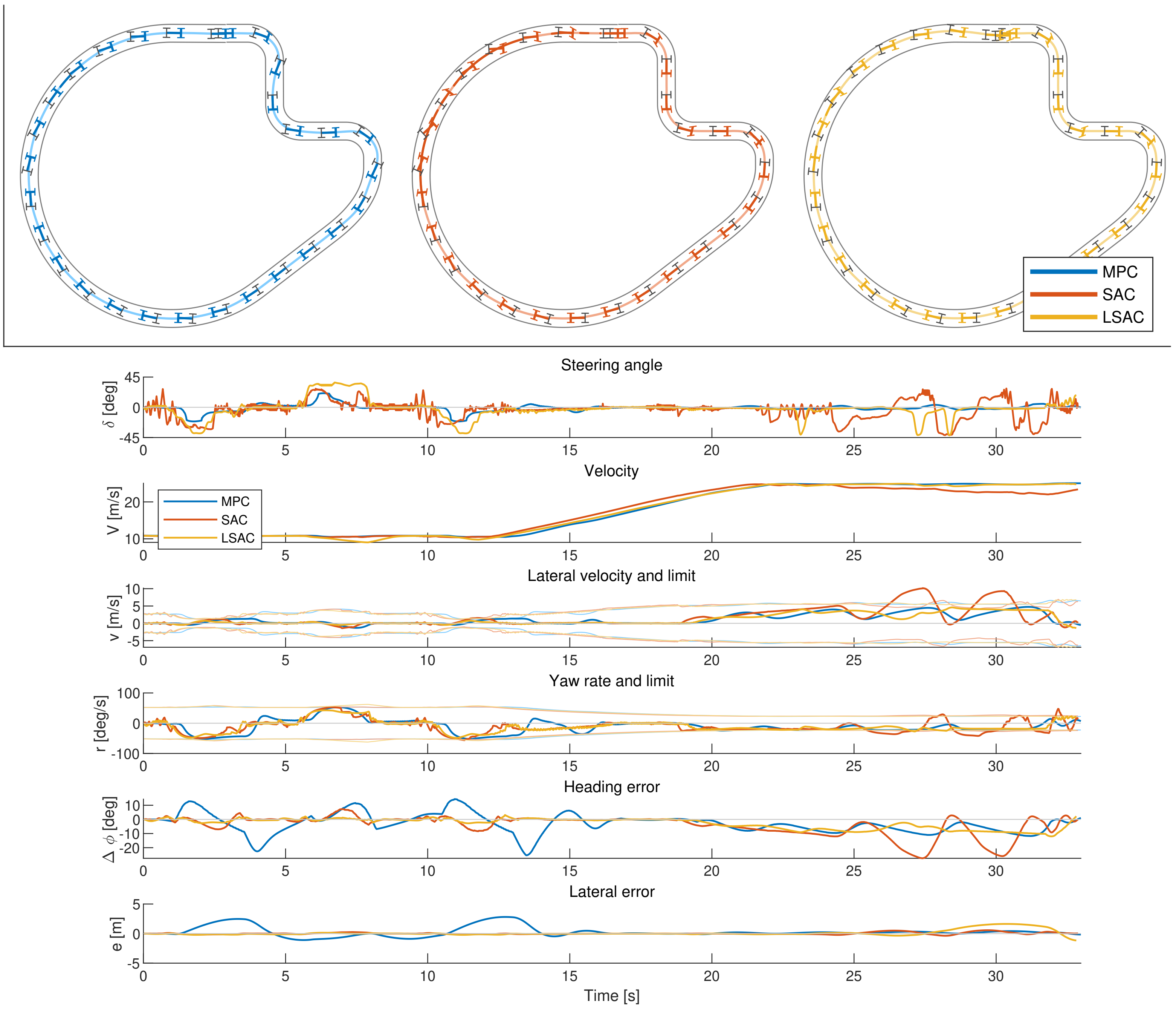
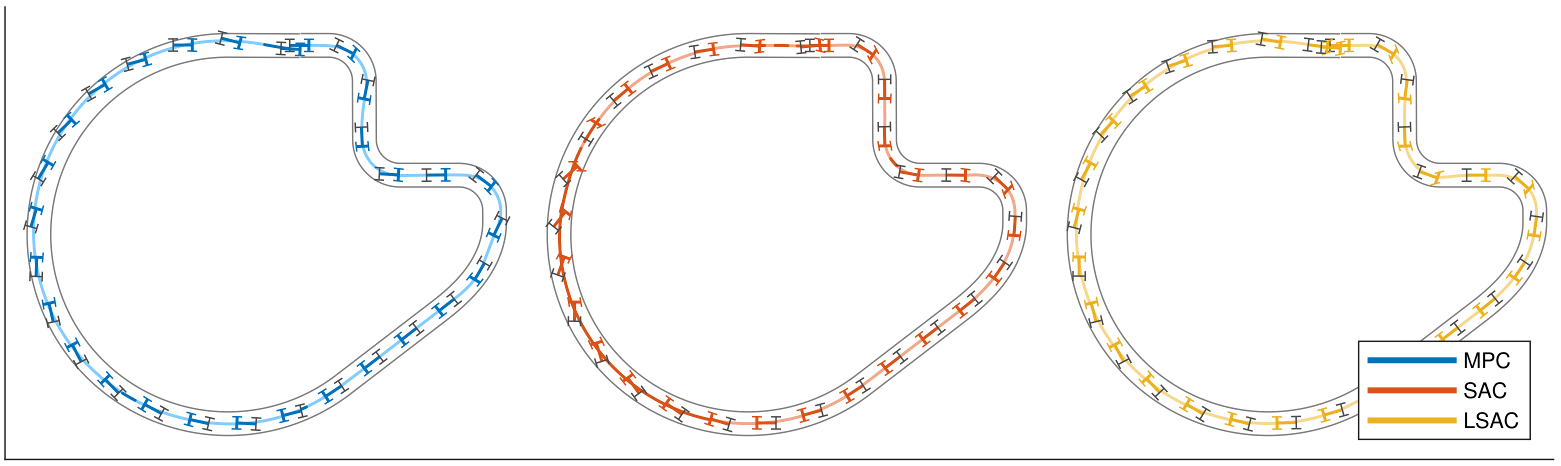


Figure 6-18: Top: Vehicles and trajectories for path tracking with vehicle stability around the race track. The target velocity $V_{\text{target}} = \sqrt{0.80} \cdot V_{\text{max}}$ so that the vehicle uses 80% of the available tire-road friction. V_{target} ranges from 39.0 km/h at the start to 90.2 km/h at the finish. The thick line sections indicate that the vehicle is in an unstable state. Bottom: Steering actions and relevant vehicle states and limits during the experiment.

6-5-4 Path tracking at 85% friction utilisation: $V_{\text{target}} = \sqrt{0.85} \cdot V_{\text{max}}$



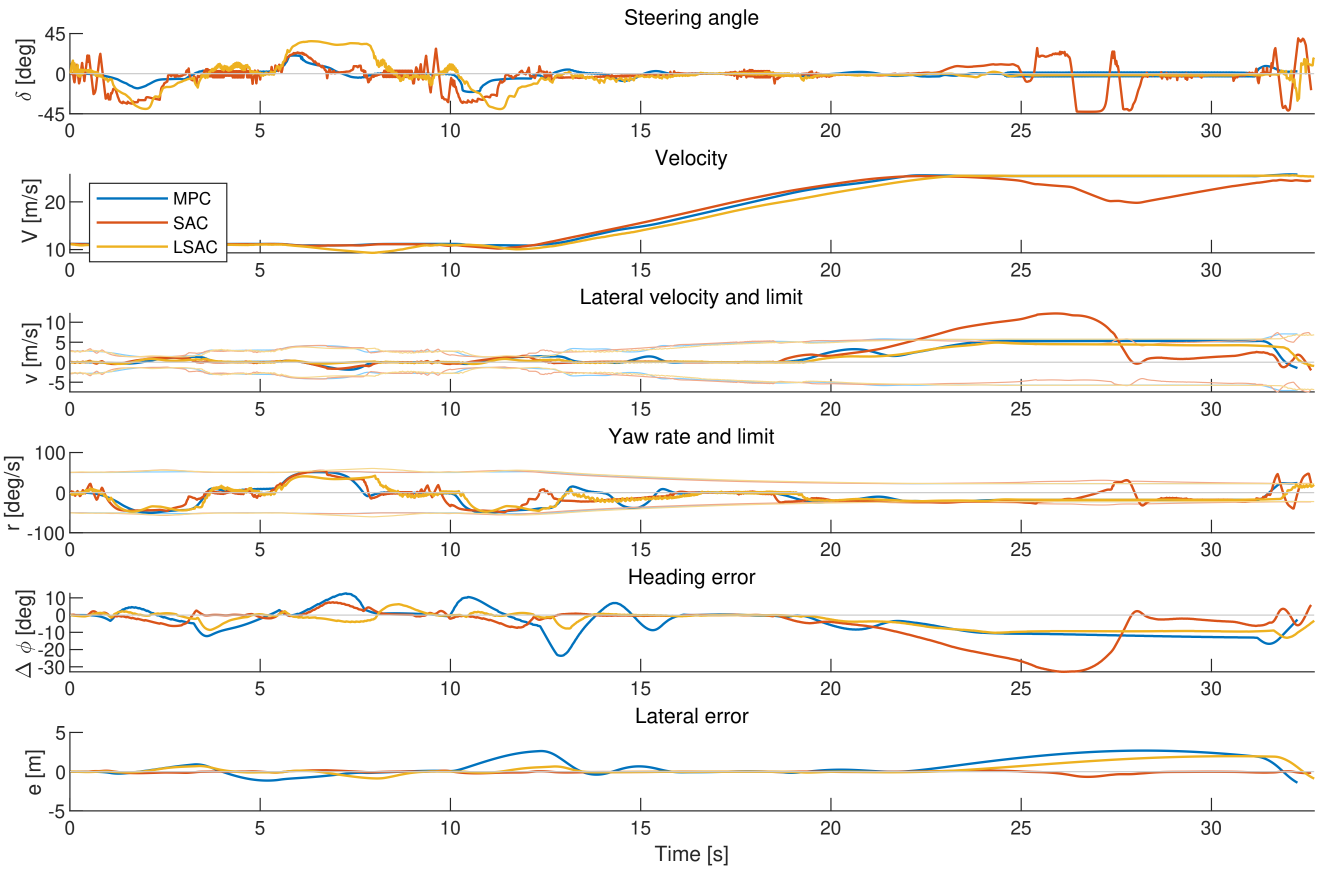


Figure 6-19: Top: Vehicles and trajectories for path tracking with vehicle stability around the race track. The target velocity $V_{\text{target}} = \sqrt{0.85} \cdot V_{\text{max}}$ so that the vehicle uses 85% of the available tire-road friction. V_{target} ranges from 40.2 km/h at the start to 93.0 km/h at the finish. The thick line sections indicate that the vehicle is in an unstable state. Bottom: Steering actions and relevant vehicle states and limits during the experiment.

6-5-5 Path tracking at 90% friction utilisation: $V_{\textbf{target}} = \sqrt{0.90} \cdot V_{\textbf{max}}$

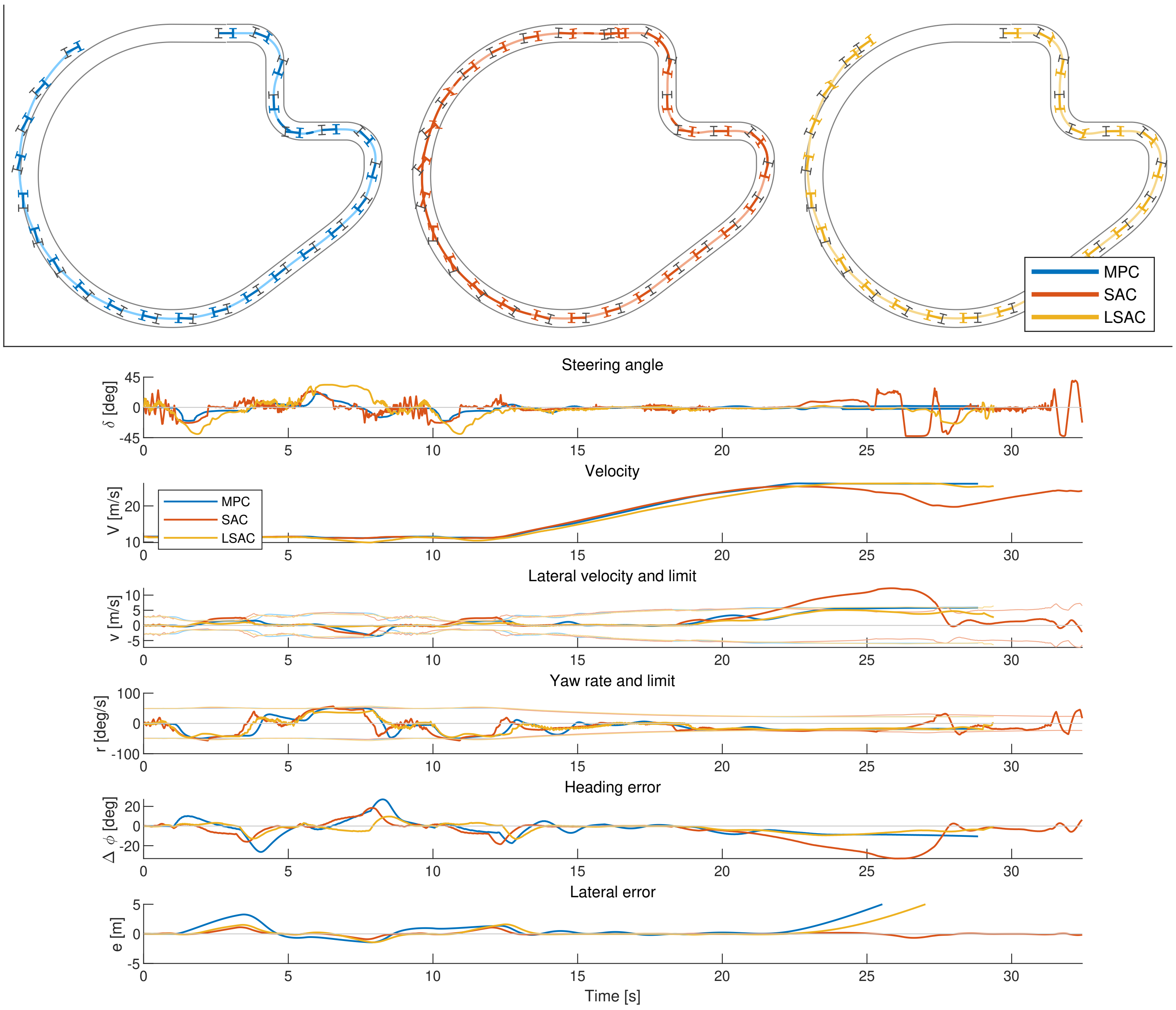


Figure 6-20: Top: Vehicles and trajectories for path tracking with vehicle stability around the race track. The target velocity $V_{\text{target}} = \sqrt{0.90} \cdot V_{\text{max}}$ so that the vehicle uses 90% of the available tire-road friction. V_{target} ranges from 41.4 km/h at the start to 95.7 km/h at the finish. The thick line sections indicate that the vehicle is in an unstable state. Bottom: Steering actions and relevant vehicle states and limits during the experiment.

6-5-6 Discussion

In this task, the main objective was to minimize the path tracking error $|e|$ around a race track when the vehicle is driving near the limit of friction. After 16 hours of training and roughly 1700 laps around the circuit the average reward for both SAC and LSAC is still climbing (Figure 6-17). This suggests that better performance may be obtained if the agents are trained for longer. Due to time limitations and sufficient performance, this is not investigated further.

MPC and LSAC are constrained to keep the vehicle stable, SAC only optimizes the main objective of path tracking.

In terms of pure path tracking, SAC outperforms both MPC and LSAC. This is especially apparent for higher speeds ($\mu_{\text{des}} = 85\%$ and 90%) where MPC and LSAC do not have the same maneuverability as SAC.

SAC can operate the vehicle in an unstable state without losing control. It is drifting the vehicle around the corners, which is typically a skill reserved for trained racing drivers. The agent steers in the opposite direction of the corner when the vehicle starts to over-rotate to keep it from spinning. This can best be seen in the steering actions between t = 25 [s] and t = 30 [s]. Keep in mind that this behavior was not taught in any way, the agent was only rewarded for minimizing the tracking error.

A downside of the large body slip angles is that it slows the vehicle down significantly. Therefore it is difficult to judge the performance compared with the faster LSAC and MPC agents.

LSAC and MPC behavior is very similar. Both rarely violate the stability constraints, but to keep the vehicle stable at higher speeds they must take the corner so wide that the vehicle leaves the road. Constraining the vehicle to stability at all times limits the maneuverability which can be crucial in emergency situations.

It is obviously undesirable that the vehicle will ultimately navigate off the road to avoid stability violations. On the other hand, the experiments in Section 6-4 showed the importance of vehicle stability. The next experiment tries to find a good balance between safe navigation and vehicle stability.

6-6 Safe navigation at the friction limit with vehicle stability

6-6-1 Introduction

The results from Section 6-5 show that when only the path tracking error $|e|$ is considered, the SAC agent has excellent path tracking but the vehicle is very unstable. When the stability is constrained with LSAC and MPC the vehicle will drive off the road to avoid instability.

The goal of this experiment is to find a control policy that balances path tracking and vehicle stability while navigating safely within the limits of the environment. The desired solution would show unstable driving behavior only when it is necessary to avoid leaving the road but keep the vehicle stable otherwise.

6-6-2 Problem definition

The aim is to find a control policy that correctly balances three -sometimes conflicting- objectives. Vehicle stability should be prioritized over path tracking, but the vehicle may temporarily be brought in an unstable state if it is necessary to satisfy the environmental constraint of not leaving the road.

In MPC the constraints can be scaled with different weights to prioritize safe navigation over vehicle stability. In RL a different solution is required. Simply scaling the rewards of the different objectives gives no guarantee that the most important constraints will not be violated temporarily to optimize the long term return.

6-6-3 Proposed solution and approach

The proposed solution is as follows. The reward function will define the desired balance between path tracking and vehicle stability by penalizing instability. The optimal policy without constraints on the environment (SAC) should behave similarly to the results of 6-5 where MPC and LSAC take the corner wider and wider to keep the vehicle stable.

Then the LSAC agent is constrained to navigate within the safe boundaries of the road, forcing the agent to find a non-optimal (lower reward) but safe solution.

There are several benefits of this arrangement over using regular unconstrained reinforcement learning methods. Perhaps the largest benefit is that the desired driving behavior does not have to be defined explicitly. Only two objectives need to be tuned in the reward; path tracking and vehicle stability. If possible, the constraint will be satisfied with the UUB stability guarantee of the LSAC algorithm.

6-6-4 Training

The training setup is similar to the previous experiment of Section 6-5. The agents are trained for 3e6 time steps in the environment, which translates to a driving time of around 16 hours.

The vehicle is started with a random target velocity from $\mu_{\text{des}} \in [0.70, 0.85]$. The initial lateral error e is picked from $\in [-0.1, 1]$ [m] and the initial heading error $\Delta\phi$ from $\in [-10, 10]$

[deg]. An episode is ended if $|e| > e_{\text{lim}}$) (30 m), $|\Delta\phi| > \Delta\phi_{\text{lim}}$ (90 deg) or the end of the lap is reached ($s > 565$ [m]).

With both the tracking error e and the stability constraints in the reward it was found that penalizing the steering actions δ as in the previous experiments to smooth the steering input was not effective. Instead, the entire reward is multiplied with a measure of the vehicle velocity $(u^2 + v^2)$. This does not punish steering actions directly but rather rewards the higher vehicle speeds that result from smooth steering inputs.

The safety cost is only active when the vehicle goes outside the track limits. The average reward and safety cost per episode during the training process are plotted in Figure 6-21.

$$\text{reward } r(s,a) = \left(u^2 + v^2\right)\left(1 - 0.9\left(\frac{|e|}{e_{\text{lim}}}\right)^{0.2} - 0.1\left(\frac{\max(0, |v - v_{\max}|)}{|v|} + \frac{\max(0, |r - r_{\max}|)}{|r|}\right)\right) \tag{6-8}$$

Where e_{lim} is 30 [m] and $v_{\max}$ and $r_{\max}$ are the limits for stable driving. The safety cost is used to penalize the agents when the vehicle is outside the road boundaries:

$$\text{safety cost } b(s,a) = \max\left(0, |e| - 5\right)^2 \tag{6-9}$$

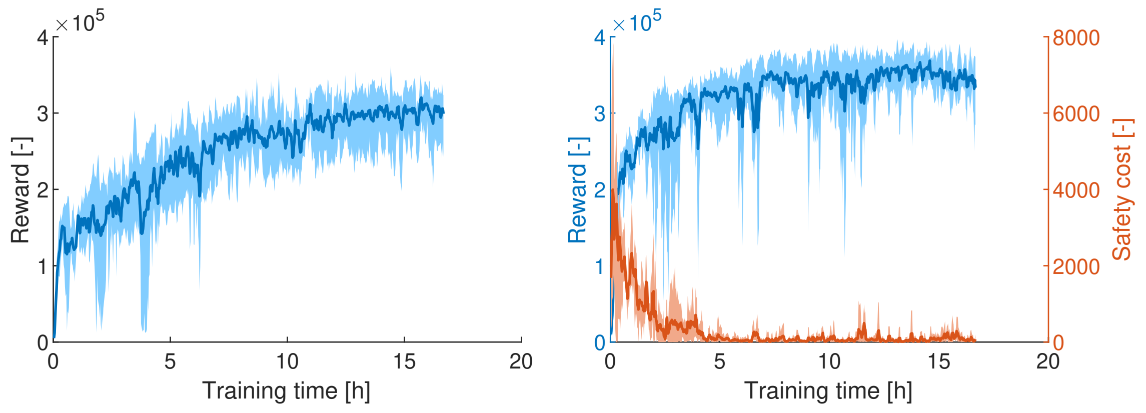


Figure 6-21: The solid lines are the average reward and safety cost of 5 agents during the training progress. The shaded areas show the variance between the different agents. The agents are rewarded for minimizing the tracking error e and vehicle instability. The safety costs are for driving the vehicle outside the road boundaries.

6-6-5 Safe navigation at 80% friction utilisation: $V_{\textbf{target}} = \sqrt{0.80} \cdot V_{\textbf{max}}$

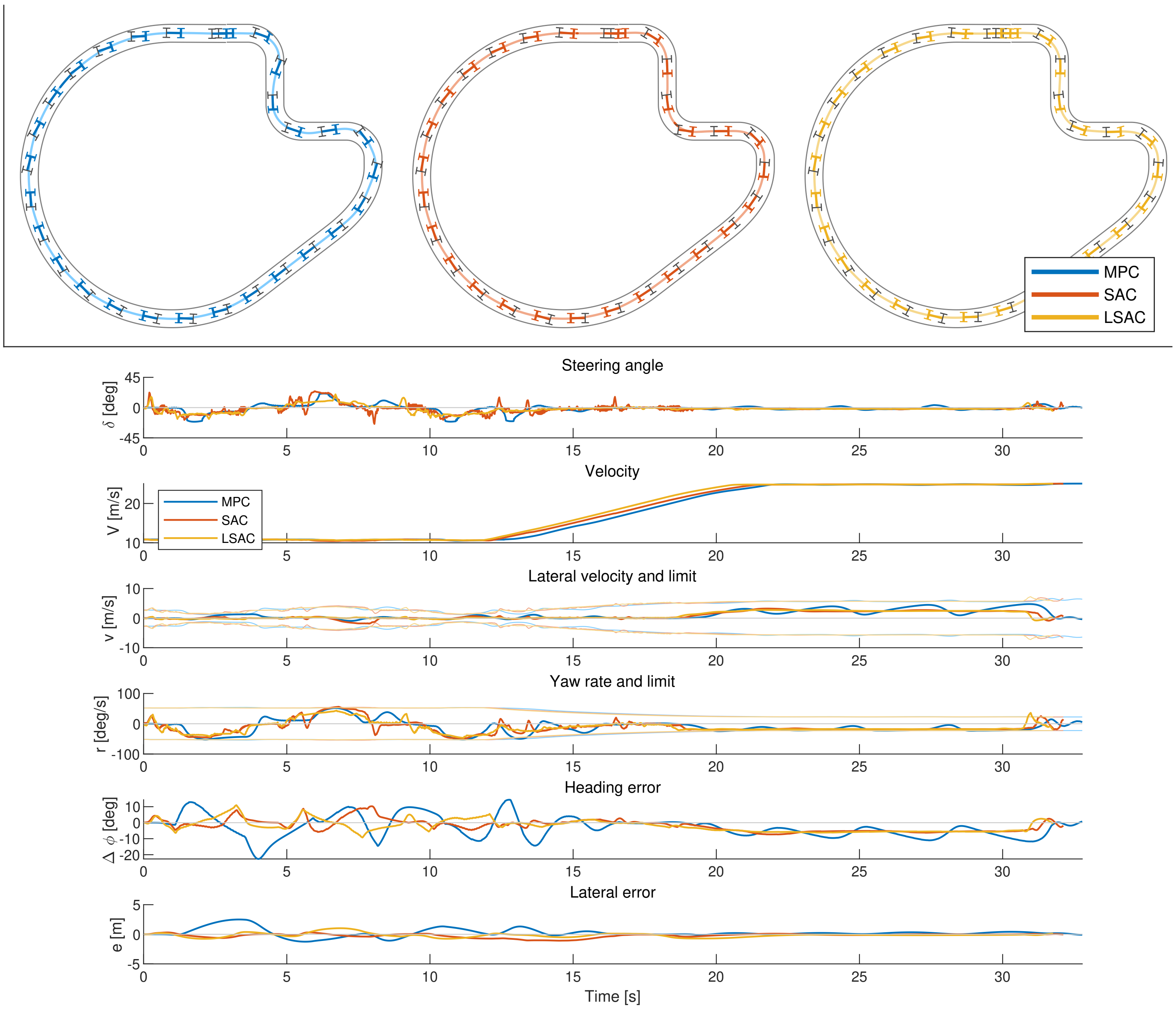


Figure 6-22: Top: Vehicles and trajectories for safe navigation with vehicle stability around the race track. The target velocity $V_{\text{target}} = \sqrt{0.80} \cdot V_{\text{max}}$ so that the vehicle uses 80% of the available tire-road friction. V_{target} ranges from 39.0 km/h at the start to 90.2 km/h at the finish. The thick line sections indicate that the vehicle is in an unstable state. Bottom: Steering actions and relevant vehicle states and limits during the experiment.

6-6-6 Safe navigation at 85% friction utilisation: $V_{\textbf{target}} = \sqrt{0.85} \cdot V_{\textbf{max}}$

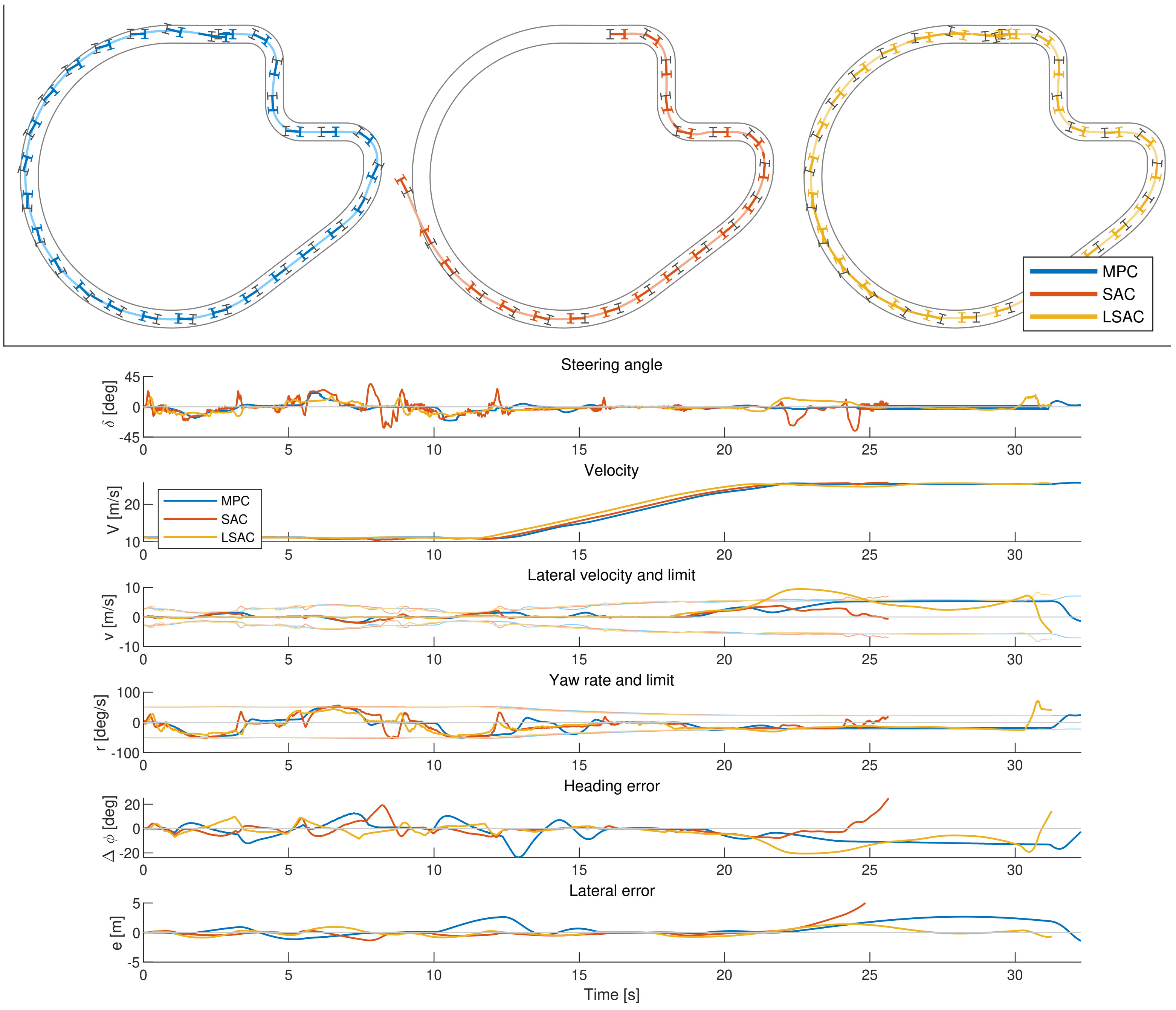


Figure 6-23: Top: Vehicles and trajectories for safe navigation with vehicle stability around the race track. The target velocity $V_{\text{target}} = \sqrt{0.85} \cdot V_{\text{max}}$ so that the vehicle uses 85% of the available tire-road friction. V_{target} ranges from 40.2 km/h at the start to 93.0 km/h at the finish. The thick line sections indicate that the vehicle is in an unstable state. Bottom: Steering actions and relevant vehicle states and limits during the experiment.

6-6-7 Safe navigation at 90% friction utilisation: $V_{\textbf{target}} = \sqrt{0.90} \cdot V_{\textbf{max}}$

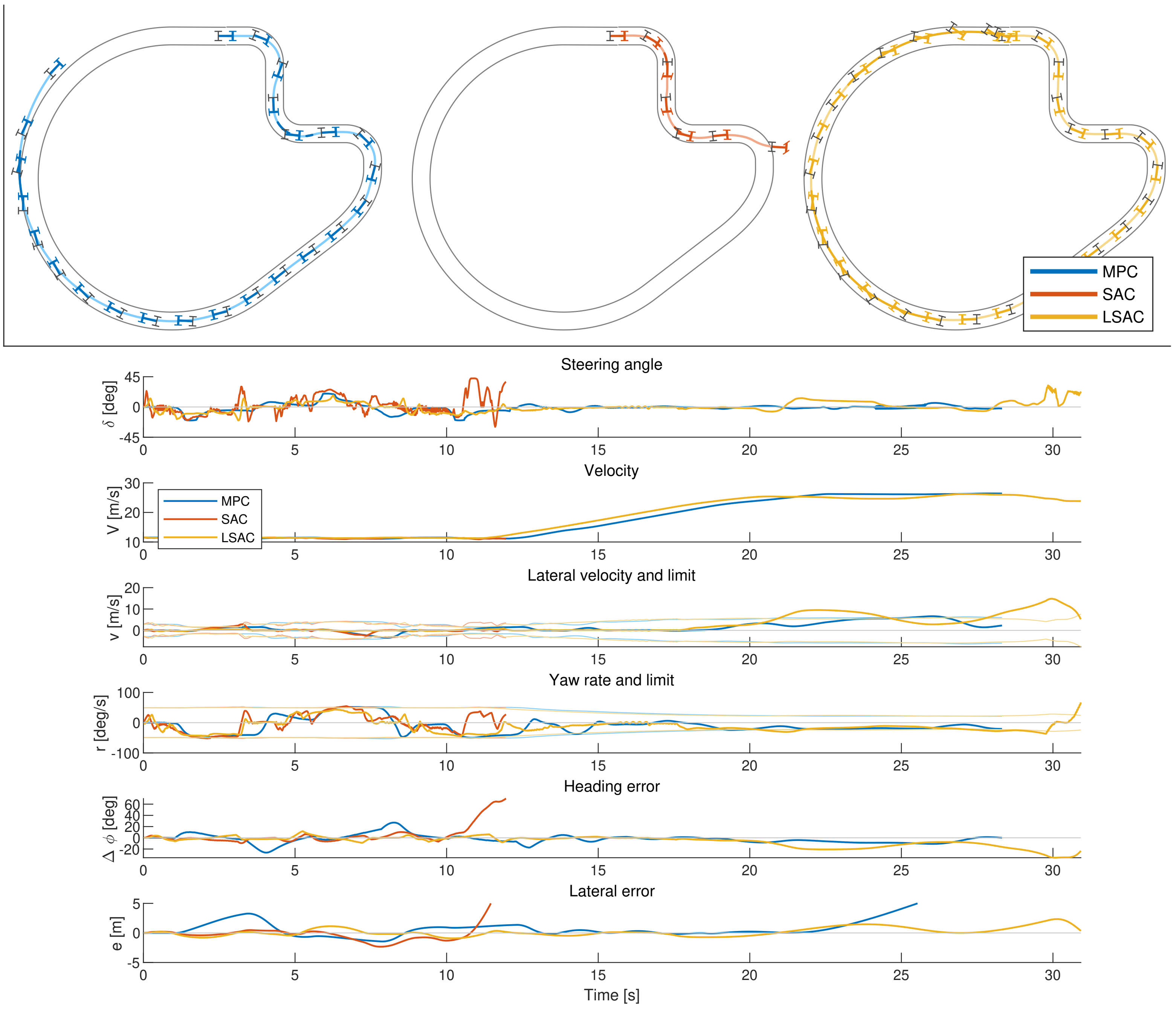


Figure 6-24: Top: Vehicles and trajectories for safe navigation with vehicle stability around the race track. The target velocity $V_{\text{target}} = \sqrt{0.90} \cdot V_{\text{max}}$ so that the vehicle uses 90% of the available tire-road friction. V_{target} ranges from 41.4 km/h at the start to 95.7 km/h at the finish. The thick line sections indicate that the vehicle is in an unstable state. Bottom: Steering actions and relevant vehicle states and limits during the experiment.

6-6-8 Discussion

The goal of this experiment was to safely navigate around the circuit while keeping the vehicle stable if possible.

The performance of MPC with the environmental constraint is only marginally better than without from Section 6-5. It reacts too late to the limits of the road and is only unstable if the vehicle is already off the road. Better performance may be possible with a different prediction model or different tuning of the controller settings.

While the SAC behavior looks bad, it shows exactly the intended behavior; the vehicle is stable most of the time.

The LSAC agent is generally stable as well, but it starts to violate the stability constraints when the vehicle is at risk of leaving the road. It is important to realize that the SAC and LSAC agents are trained with the same rewards. Without the constraint, the results of SAC and LSAC would be the same. The difference between the agents is only because the LSAC agent is constrained to stay inside the road boundaries.

6-7 Racing inside track limits

6-7-1 Introduction

The results from the previous experiment show that LSAC can be used to train an agent that only explores in the safe limits of the environment. The goal of this experiment is to show the versatility of LSAC in constrained control tasks by finding the fastest way around a race track while staying within the track limits.

6-7-2 Training

The agents are trained for 2e6 time steps in the simulation environment. This is around 11 hours of driving on the track. At the beginning of an episode the target velocity is picked from $\mu_{\text{des}} \in [0.70, 0.85]$, the initial lateral error e from $\in [-0.1, 1]$ [m] and the initial heading error $\Delta\phi$ from $\in [-10, 10]$ [deg]. An episode is ended if $|e| > e_{\text{lim}})$ (10 m), $|\Delta\phi| > \Delta\phi_{\text{lim}}$ (90 deg) or the end of the lap is reached ($s > 565$ [m]).

The fastest way around a race track is purely defined by the lap time, and the best performance is the lowest lap time. Therefore the most accurate reward would be related -inversely- to the lap time. But only giving a reward after an entire lap has passed is very difficult for learning. The agent will get no feedback until it learns to drive around the track first. Learning is much quicker and more consistent when a reward 'guides' the agent to the desired solution.

Therefor the agent is rewarded for maximizing $\dot{s}$. $\dot{s}$ is the distance covered in the direction of the track every time step:

$$\dot{s} = \left(\frac{u \cos \Delta\phi - v \sin \Delta\phi}{1 - K(s)e} \right) \tag{6-10}$$

with u and v the longitudinal and lateral vehicle velocities, $\Delta\phi$ the heading angle, e the distance from the centerline of the track and K the curvature of the track. Vehicle instability is discouraged with a small penalty. The safety costs constrain the agents to find the best solution within the track limits. The average reward and safety cost per training episode can be seen in Figure 6-25.

$$\text{reward } r(s,a) = \dot{s} \left(1 - 0.1 \left(\frac{\max(0, |v - v_{\max}|)}{|v|} \right) - 0.1 \left(\frac{\max(0, |r - r_{\max}|)}{|r|} \right) \right) \tag{6-11}$$

Where $v_{\max}$ and $r_{\max}$ are the limits for stable driving.

$$\text{safety cost } b(s,a) = 0.1 \cdot \max(0, |e| - 5) \tag{6-12}$$

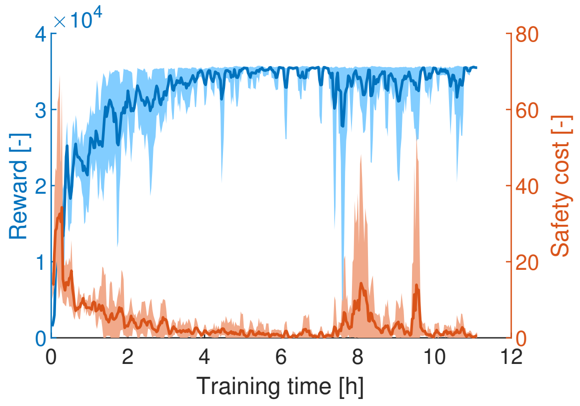


Figure 6-25: The solid lines are the average reward and safety cost per episode of 5 agents during the training progress. The shaded areas show the variance between the different agents. The agents are rewarded for maximizing the traveled distance on the track ($\dot{s}$). The safety costs are for driving the vehicle outside the track limits.

6-7-3 Racing inside track limits for 85% friction utilisation: $V_{\textbf{target}} = \sqrt{0.85} \cdot V_{\textbf{max}}$

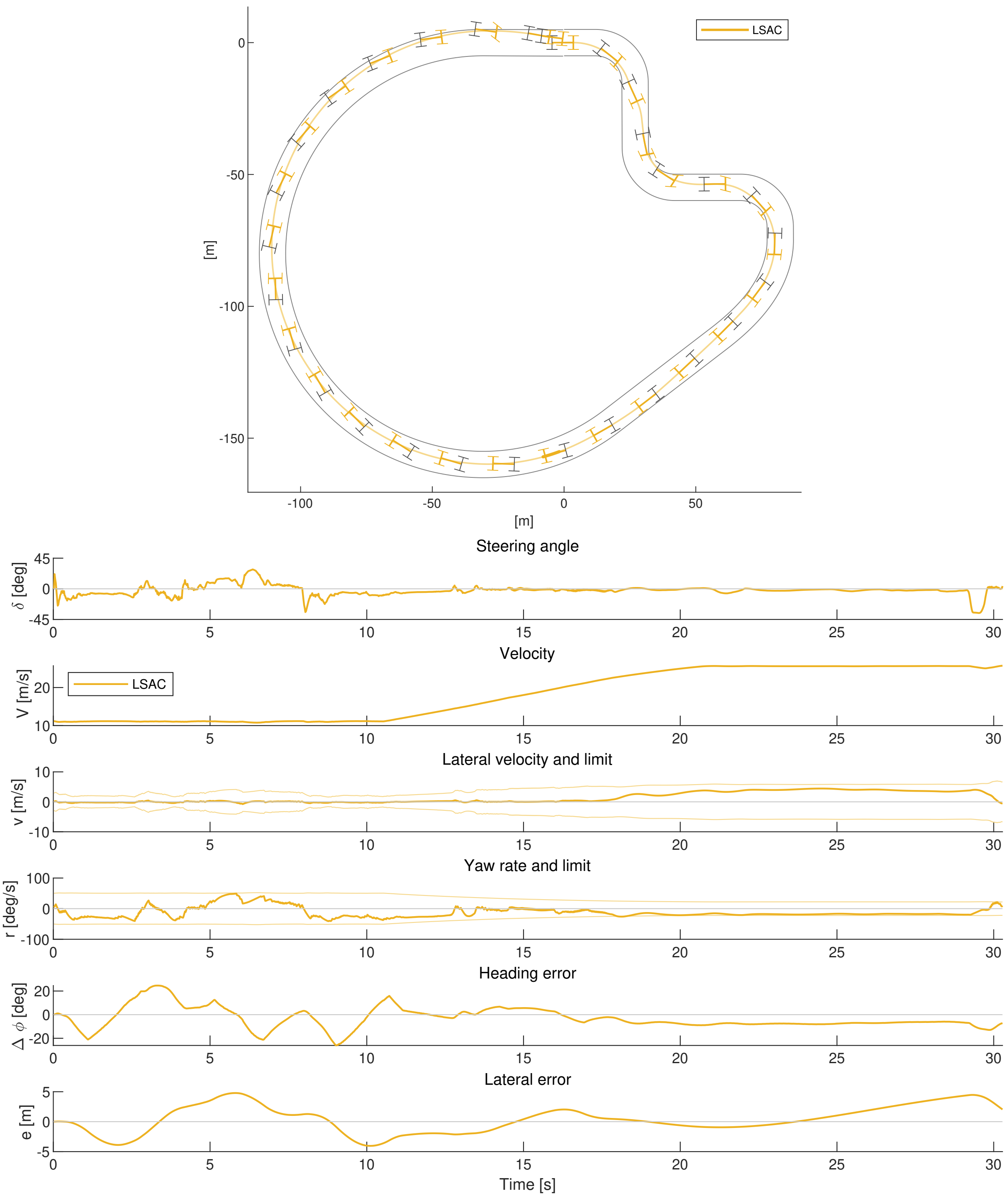


Figure 6-26: Top: Vehicle and trajectory around the race track for a target velocity $V_{\text{target}} = \sqrt{0.85} \cdot V_{\text{max}}$ so that the vehicle uses 85% of the available tire-road friction. V_{target} ranges from 40.2 km/h at the start to 93.0 km/h at the finish. Bottom: Steering actions and relevant vehicle states and limits during the experiment.

6-7-4 Discussion

In this experiment, the reward is set to maximize the traveled distance on the path each time step ($\dot{s}$). Maximizing $\dot{s}$ will maximize both the speed of the vehicle as well as optimize the position and orientation on the track. It is clear that LSAC tries to maximize the reward by cutting the corners and minimizing the steering inputs to hold its speed.

To quantify how much time can be gained per lap, LSAC is compared against the MPC and SAC agents with the same target speed ($\sqrt{0.85} \cdot V_{\max}$) from Section 6-5. LSAC manages to drive around the track in 30.28 seconds, which is significantly faster than MPC (32.26 seconds) and SAC (32.62 seconds). The comparison with lap times and average velocities can be found in Table 6-1.

Table 6-1: Lap times and average vehicle speeds V_{avg} for MPC, SAC and LSAC for a target velocity $V_{\text{target}} = \sqrt{0.85} \cdot V_{\text{max}}$.

Controller	V_{avg} [m/s](km/h)	Laptime [s]
LSAC	18.27 (65.8)	30.28
MPC	17.85 (64.3)	32.26 +1.98 (106.5%)
SAC	17.31 (62.3)	32.62 +2.34 (107.7%)

Chapter 7

Conclusions and Recommendations

7-1 Conclusions

Automated vehicles have the potential to increase road safety, reduce congestion, and provide mobility to those who are unable to operate a vehicle manually. Developing safe and robust control strategies for automated vehicles is perhaps the biggest challenge towards fully automated driving under all conditions.

Reinforcement Learning (RL) algorithms have been able to master complicated tasks such as video games, image recognition, and robot locomotion. Tasks that can be very difficult to solve with classical control methods can now be solved without the need to fully understand all aspects of the problem.

One of the downsides of RL for safety-critical tasks is that there is no guarantee on the behavior of the policies. A policy may make a bad move now because it expects a better reward in the future. For automated vehicles, a single bad decision can have a fatal outcome.

A study of the relevant literature reveals that there are currently no publications of automated driving policies with a focus on vehicle stability and safety when driving near the friction limit. Therefore the objective of this work is the following:

Developing a safe and robust control policy for automated vehicles using safe reinforcement learning.

The main contributions of this work are automated driving policies developed with the Lyapunov Safe Actor-Critic (LSAC) algorithm. LSAC is a combination of the Soft Actor-Critic (SAC) algorithm and Lyapunov stability analysis to guarantee a uniformly ultimately bounded (UUB) policy. LSAC can be used to solve control problems with safety constraints.

The performance of the LSAC policies is compared against SAC and Model Predictive Control (MPC). SAC is a state-of-the-art RL algorithm for unconstrained control tasks that outperformed most other algorithms in a series of benchmark tests. MPC represents the best of the classic control methods to deal with constrained control problems.

The control policies are trained and evaluated in a vehicle simulator for the following driving tasks:

- Navigating through traffic at 50, 75 and 100 km/h
- Recovering from destabilizing impacts
- Driving on a race track and maintaining vehicle stability
- Safe navigation on a race track and maintaining vehicle stability
- Finding the racing line on a race track inside track limits

The following conclusions can be drawn from the experiments:

There is no significant difference in performance for simple driving tasks that are well within the limits of friction. The minor differences between MPC, SAC, and LSAC come down to optimizing the controller settings.

When the vehicle is suddenly disturbed and forced outside the stability region, LSAC clearly outperforms MPC and SAC. The experiments where the vehicle undergoes a destabilizing impact on the rear the LSAC agent with stability constraint can regain control over the vehicle in cases where MPC and SAC lose control.

Only constraining vehicle stability can lead to unsafe policies. Experiments on a race track showed that a constrained policy will go too far off the road to avoid stability violations.

Constraining the environment instead of vehicle stability gives a policy that can safely navigate through the environment at the limit of friction. The best all-round policy was obtained with LSAC with constraints on the environment and a penalty for vehicle instability. The policy will generally keep the vehicle stable, but when it gets too close to the edges of the road it will use the extra maneuverability of unstable vehicle states to keep the vehicle on the road.

LSAC can be used to optimize the speed around a race track. Experiments on a race track show that LSAC with constraints on the environment can be used to find the fastest way around a racetrack while staying within the track limits. It is significantly faster than regular SAC and MPC agents with the same target velocity.

7-2 Recommendations for future work

Many limitations in this work are deliberate to provide a fair comparison between LSAC, SAC, and a derivation of the MPC strategy in the work of Funke et al [11, 12]. Now that results suggest that RL and specifically LSAC is a viable alternative to MPC for automated vehicles, further investigation to unlock the full potential is warranted. The following issues should be considered in future work:

7-2-1 Action space

In this work, only the steering actions of the vehicle are considered. The longitudinal speed control is still done by an outside controller. In future work, using reinforcement learning for both lateral (steering) and longitudinal (throttle) control -and their interaction- should be investigated.

7-2-2 Environment

The test environments in this work are fairly simple and focused on providing a clear comparison between different control strategies. However, more varied environments should be considered, as well as their effect on training times and the robustness of policies.

Previewing the environment was done by predicting the road curvature over the next second with 0.1 second time intervals. The effect of different time horizons as well as the effect of different preview resolutions is currently unknown. Noisy measurements are also not yet considered.

7-2-3 Robustness and scalability

Future work should consider the robustness of the policies under heavily changing conditions such as different road frictions. Are the policies viable on a different vehicle? Do the policies scale from the simulator to a real vehicle? How well do the policies deal with the measurement noise of the real world? Answering these questions can be a key component of safe automated driving.

Bibliography

[1] E. Commission, "Advanced driver assistance systems," *European Commission, Directorate General for Transport*, February 2018.

[2] S. International, "Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles." https://www.sae.org/standards/content/j3016_201806/, 2018.

[3] Tesla, "Tesla autopilot." https://tesla.com/autopilot/, 2020.

[4] J. M. Anderson, N. Kalra, K. D. Stanley, P. Sorensen, C. Samaras, and T. A. Oluwatola, "Autonomous vehicle technology: A guide for policymakers.," *Santa Monica, CA: RAND Corporation*, 2016.

[5] D. Fagnant and K. Kockelman, "Preparing a nation for autonomous vehicles: Opportunities, barriers and policy recommendations," *Transportation Research Part A: Policy and Practice*, vol. 77, 07 2015.

[6] I. Barabás, A. Todoruţ, N. Cordoş, and A. Molea, "Current challenges in autonomous driving," *IOP Conference Series: Materials Science and Engineering*, vol. 252, p. 012096, oct 2017.

[7] N. Kallioinen, M. Pershina, J. Zeiser, F. Nosrat Nezami, G. Pipa, A. Stephan, and P. König, "Moral judgements on the actions of self-driving cars and human drivers in dilemma situations from different perspectives," *Frontiers in Psychology*, vol. 10, p. 2415, 2019.

[8] C. E. Beal and J. C. Gerdes, "Model predictive control for vehicle stabilization at the limits of handling," *IEEE Transactions on Control Systems Technology*, vol. 21, pp. 1258–1269, July 2013.

[9] X. He, Y. Liu, C. Lv, X. Ji, and Y. Liu, "Emergency steering control of autonomous vehicle for collision avoidance and stabilisation," *Vehicle System Dynamics*, vol. 57, no. 8, pp. 1163–1187, 2019.

[10] K. Kritayakirana and J. C. Gerdes, “Autonomous vehicle control at the limits of handling,” *International Journal of Vehicle Autonomous Systems*, vol. 10, p. 271, 2012.

[11] J. Funke, M. Brown, S. M. Erlien, and J. C. Gerdes, “Prioritizing collision avoidance and vehicle stabilization for autonomous vehicles,” in *2015 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1134–1139, June 2015.

[12] J. Funke, M. Brown, S. M. Erlien, and J. C. Gerdes, “Collision avoidance and stabilization for autonomous vehicles in emergency scenarios,” *IEEE Transactions on Control Systems Technology*, vol. 25, pp. 1204–1216, July 2017.

[13] A. Arab, K. Yu, J. Yi, and Y. Liu, “Motion control of autonomous aggressive vehicle maneuvers,” in *2016 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, pp. 1663–1668, 2016.

[14] Y. Zheng, “Thesis: Real-time nonlinear mpc for extreme lateral stabilization of passenger vehicles,” 2018.

[15] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, “Playing atari with deep reinforcement learning,” 2013.

[16] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. P. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, “Mastering the game of go with deep neural networks and tree search,” *Nature*, vol. 529, pp. 484–489, 2016.

[17] Y. Zheng, “Reinforcement learning and video games,” 2019.

[18] A. R. Mahmood, D. Korenkevych, G. Vasan, W. Ma, and J. Bergstra, “Benchmarking reinforcement learning algorithms on real-world robots,” 2018.

[19] OpenAI, M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba, “Learning dexterous in-hand manipulation,” 2018.

[20] S. Gu, E. Holly, T. Lillicrap, and S. Levine, “Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates,” 2016.

[21] N. Kohl and P. Stone, “Policy gradient reinforcement learning for fast quadrupedal locomotion,” vol. 3, pp. 2619 – 2624 Vol.3, 01 2004.

[22] G. Endo, J. Morimoto, T. Matsubara, J. Nakanishi, and G. Cheng, “Learning cpg-based biped locomotion with a policy gradient method: Application to a humanoid robot,” *The International Journal of Robotics Research*, vol. 27, no. 2, pp. 213–228, 2008.

[23] J. Peters and S. Schaal, “Reinforcement learning of motor skills with policy gradients,” *Neural Networks*, vol. 21, no. 4, pp. 682 – 697, 2008. Robotics and Neuroscience.

[24] E. Theodorou, J. Buchli, and S. Schaal, “Reinforcement learning of motor skills in high dimensions: A path integral approach,” pp. 2397 – 2403, 06 2010.

[25] M. Kalakrishnan, L. Righetti, P. Pastor, and S. Schaal, "Learning force control policies for compliant manipulation," pp. 4639–4644, 09 2011.

[26] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2015.

[27] Y. Peng, J. Zhang, and Z. Ye, "Deep reinforcement learning for image hashing," 2018.

[28] T. Chen, Z. Wang, G. Li, and L. Lin, "Recurrent attentional reinforcement learning for multi-label image recognition," 2017.

[29] Z. Wang, S. Sarcar, J. Liu, Y. Zheng, and X. Ren, "Outline objects using deep reinforcement learning," 2018.

[30] S. Mathe, A. Pirinen, and C. Sminchisescu, "Reinforcement learning for visual object detection," pp. 2894–2902, 06 2016.

[31] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J.-M. Allen, V.-D. Lam, A. Bewley, and A. Shah, "Learning to drive in a day," 2018.

[32] J. Chen, B. Yuan, and M. Tomizuka, "Model-free deep reinforcement learning for urban autonomous driving," 2019.

[33] D. Isele, R. Rahimi, A. Cosgun, K. Subramanian, and K. Fujimura, "Navigating occluded intersections with autonomous vehicles using deep reinforcement learning," 2017.

[34] M. Vitelli and A. Nayebi, "Carma : A deep reinforcement learning approach to autonomous driving," 2016.

[35] P. Wang and C.-Y. Chan, "Formulation of deep reinforcement learning architecture toward autonomous driving for on-ramp merge," *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, Oct 2017.

[36] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani, "Deep reinforcement learning framework for autonomous driving," 2017.

[37] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "Safe, multi-agent, reinforcement learning for autonomous driving," 2016.

[38] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani, "End-to-end deep reinforcement learning for lane keeping assist," 2016.

[39] X. Xiong, J. Wang, F. Zhang, and K. Li, "Combining deep reinforcement learning and safety based control for autonomous driving," 2016.

[40] D. C. K. Ngai and N. H. C. Yung, "A multiple-goal reinforcement learning method for complex vehicle overtaking maneuvers," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 2, pp. 509–522, 2011.

[41] A. Folkers, M. Rick, and C. Buskens, "Controlling an autonomous vehicle with deep reinforcement learning," *2019 IEEE Intelligent Vehicles Symposium (IV)*, Jun 2019.

[42] D. Maravall, M. A. Patricio, and J. Asiaín, “Automatic car parking: A reinforcement learning approach,” vol. 2686, pp. 214–221, 06 2003.

[43] P. Zhang, L. Xiong, Z. Yu, P. Fang, S. Yan, J. Yao, and Y. Zhou, “Reinforcement learning-based end-to-end parking for automatic parking system,” *Sensors (Basel, Switzerland)*, vol. 19, 2019.

[44] A. Raffin and R. Sokolkov, “Learning to drive smoothly in minutes.” `https://github.com/araffin/learning-to-drive-in-5-minutes/`, 2019.

[45] A. Yu, R. Palefsky-Smith, and R. Bedi, “Deep reinforcement learning for simulated autonomous vehicle control,” 2016.

[46] A. Remonda, S. Krebs, E. Veas, G. Luzhnica, and R. Kern, “Formula rl: Deep reinforcement learning for autonomous racing using telemetry data,” 06 2019.

[47] M. Jaritz, R. de Charette, M. Toromanoff, E. Perot, and F. Nashashibi, “End-to-end race driving with deep reinforcement learning,” 2018.

[48] P. Cai, X. Mei, L. Tai, Y. Sun, and M. Liu, “High-speed autonomous drifting with deep reinforcement learning,” *IEEE Robotics and Automation Letters*, vol. 5, p. 1247–1254, Apr 2020.

[49] M. Han, Y. Tian, L. Zhang, J. Wang, and W. Pan, “Model-free reinforcement learning with control theoretic guarantee,” 2019.

[50] Y. Tian, “Thesis: Model free reinforcement learning with stability guarantee,” 2019.

[51] M. Han, Y. Tian, L. Zhang, J. Wang, and W. Pan, “h_∞ model-free reinforcement learning with robust stability guarantee,” 2019.

[52] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor,” 2018.

[53] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine, “Soft actor-critic algorithms and applications,” 2018.

[54] “Openai gym.” `http://https://gym.openai.com/`.

[55] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, “Carla: An open urban driving simulator,” 2017.

[56] C. G. C. D. R. C. A. S. Bernhard Wymann, Eric Espié, “TORCS, The Open Racing Car Simulator.” http://www.torcs.org, 2014.

[57] C. Quiter and M. Ernst, “deepdrive/deepdrive: 2.0.” `https://doi.org/10.5281/zenodo.1248998`, Mar. 2018.

[58] Nvidia, “Drive constellation now available.” `https://blogs.nvidia.com/blog/2019/03/18/drive-constellation-now-available/`, 2019.

[59] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[60] H. Pacejka, *Tire and Vehicle Dynamics*. Oxford: Elsevier, 3 ed., 2012.

[61] G. Rill, "Wheel dynamics," in *The XII international symposium on dynamic problems of mechanics*, Sao Paulo, Brazil, 26 February–2 March, 2007.

[62] J. E. Bernard and C. L. Clover, "Tire modeling for low-speed and high-speed calculations," in *International Congress Exposition*, SAE International, feb 1995.

[63] M. C. LÓPEZ A., VÉLEZ P., "Approximations to the magic formula," *Int J Automot Technol*, vol. 11, no. 2, pp. 155–166, 2010.

[64] S. Jung, T.-Y. Kim, and W.-S. Yoo, "Advanced slip ratio for ensuring numerical stability of low-speed driving simulation. part i: Longitudinal slip ratio," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 233, no. 8, pp. 2000–2006, 2019.

[65] T.-Y. Kim, S. Jung, and W.-S. Yoo, "Advanced slip ratio for ensuring numerical stability of low-speed driving simulation: Part ii—lateral slip ratio," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 233, no. 11, pp. 2903–2911, 2019.

[66] M. Blundell and D. Harty, "Chapter 5 - tyre characteristics and modelling," in *The Multibody Systems Approach to Vehicle Dynamics (Second Edition)* (M. Blundell and D. Harty, eds.), pp. 335 – 450, Oxford: Butterworth-Heinemann, second edition ed., 2015.

[67] H. Dugoff, P. S. Fancher, and L. Segel, "An analysis of tire traction properties and their influence on vehicle dynamic performance," *SAE Transactions*, vol. 79, pp. 1219–1243, 1970.

[68] tass international, "Mf-tyre/mf-swift." `https://tass.plm.automation.siemens.com/delft-tyre-mf-tyremf-swift`, 2020.

[69] N. Ding and S. Taheri, "A modified dugoff tire model for combined-slip forces," *Tire Science and Technology*, vol. 38, no. 3, pp. 228–244, 2010.

[70] A. Bhoraskar and P. Sakthivel, "A review and a comparison of dugoff and modified dugoff formula with magic formula," in *2017 International Conference on Nascent Technologies in Engineering (ICNTE)*, pp. 1–4, 2017.

[71] J. S. Il Bae, Jaeyoung Moon, "Toward a comfortable driving experience for a self-driving shuttle bus," *Electronics*, vol. 8, no. 9, p. 943, 2019.

[72] S. M. Erlien, S. Fujita, and J. C. Gerdes, “Safe driving envelopes for shared control of ground vehicles,” *IFAC Proceedings Volumes*, vol. 46, no. 21, pp. 831 – 836, 2013. 7th IFAC Symposium on Advances in Automotive Control.

[73] E. Fiala, *Lateral Forces at the Rolling Pneumatic Tire.* 1954.

[74] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, “Kinematic and dynamic vehicle models for autonomous driving control design,” *2015 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1094–1099, 2015.

[75] P. Polack, F. Altché, B. d’Andréa Novel, and A. de La Fortelle, “The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?,” *2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 812–818, 2017.

[76] M. Hessel, J. Modayil, H. van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, “Rainbow: Combining improvements in deep reinforcement learning,” 2017.

[77] Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. de Freitas, “Sample efficient actor-critic with experience replay,” *CoRR*, vol. abs/1611.01224, 2016.

[78] S. Fujimoto, H. van Hoof, and D. Meger, “Addressing function approximation error in actor-critic methods,” *CoRR*, vol. abs/1802.09477, 2018.

[79] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, “Policy gradient methods for reinforcement learning with function approximation,” in *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS’99, (Cambridge, MA, USA), p. 1057–1063, MIT Press, 1999.

[80] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *CoRR*, vol. abs/1707.06347, 2017.

[81] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, “Asynchronous methods for deep reinforcement learning,” 2016.

[82] C. Watkins, “Learning from delayed rewards,” 01 1989.

[83] M. G. Bellemare, W. Dabney, and R. Munos, “A distributional perspective on reinforcement learning,” *CoRR*, vol. abs/1707.06887, 2017.

[84] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey, “Maximum entropy inverse reinforcement learning,” in *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*, AAAI’08, p. 1433–1438, AAAI Press, 2008.

[85] M. Toussaint, “Robot trajectory optimization using approximate inference,” in *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML ’09, (New York, NY, USA), p. 1049–1056, Association for Computing Machinery, 2009.

[86] K. Rawlik, M. Toussaint, and S. Vijayakumar, “On stochastic optimal control and reinforcement learning by approximate inference (extended abstract),” in *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI ’13, p. 3052–3056, AAAI Press, 2013.

[87] H. K. Khahil and J. W. Grizzle, “Nonlinear systems,” *Prentice hall Upper Saddle River, NJ*, vol. 3, 2002.

[88] E. Wiewiora, *Reward Shaping*, pp. 863–865. Boston, MA: Springer US, 2010.

[89] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, “Concrete problems in ai safety,” 2016.

[90] A. Faust, A. Francis, and D. Mehta, “Evolving rewards to automate reinforcement learning,” 2019.

[91] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, “Learning agile and dynamic motor skills for legged robots,” *Science Robotics*, vol. 4, no. 26, 2019.

[92] A. Braylan, M. Hollenbeck, E. Meyerson, and R. Miikkulainen, “Frame skip is a powerful parameter for learning to play atari,” in *AAAI Workshop: Learning for General Competency in Video Games*, 2015.

[93] A. S. Lakshminarayanan, S. Sharma, and B. Ravindran, “Dynamic frame skip deep Q network,” *CoRR*, vol. abs/1605.05365, 2016.

[94] J. Zhou, “Vehicle dynamics in response to the maneuver of precision immobilization technique,” 08 2008.