DEGREE PROJECT IN THE FIELD OF TECHNOLOGY
ENGINEERING PHYSICS
AND THE MAIN FIELD OF STUDY
ELECTRICAL ENGINEERING,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2020

Shared Control for Vehicle Teleoperation with a Virtual Environment Interface

ANTON BJÖRNBERG

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Shared Control for Vehicle Teleoperation with a Virtual Environment Interface

Anton Björnberg

Master in Systems, Control and Robotics
June 2020
Supervisor: Frank Jiang
Examiner: Prof. Jonas Mårtensson
KTH Royal Institute of Technology
School of Electrical Engineering and Computer Science

Abstract

Teleoperation has been suggested as a means of overcoming the high uncertainty in automated driving by letting a remote human driver assume control of an automated vehicle when it is unable to proceed on its own. Remote driving, however, suffers from restricted situational awareness and network latency, which can increase the risk for accidents. In most cases though, the automation system will still be functional, which opens the possibility for shared control where the human and machine collaborates to accomplish the task at hand. This thesis develops one such method in which an operator demonstrates their intentions by maneuvering a virtual vehicle in a virtual environment. The real vehicle observes the virtual counterpart and acts autonomously to mimic it. For this, the vehicle uses a genetic algorithm to find a path that is similar to that of the virtual vehicle, while avoiding potential obstacles. Longitudinal control is performed with a method inspired by Adaptive Cruise Control (ACC) and lateral control is achieved with a model predictive controller (MPC). The method is evaluated with user trials in simulation and is seen to perform better in the presence of large delays when compared to a direct control approach. With negligible delay, it performs on par with or slightly worse than direct control, but test participants still reported higher confidence when using the new method as well as an overall preference for the same.

Sammanfattning

Teleoperation har föreslagits som metod för att hantera den höga osäkerheten som förekommer hos självkörande fordon. När de inte klarar av att på egen hand planera sin rutt kan kontrollen istället överföras till en fjärrlokaliserad förare. Direkt fjärrstyrning innebär dock flera nackdelar i form av exempelvis begränsad lägesuppfattning och nätverksfördröjningar, vilket kan innebära en förhöjd risk för olyckor. I det flesta fall kommer dock fordonets automationssystem fortfarande vara aktivt, vilket öppnar för att använda delad styrning där människan och fordonet samarbetar för att utföra köruppgiften. I detta examensarbete utvecklas en sådan metod, i vilken en operatör demonstrerar sina avsikter genom att manövrera ett virtuellt fordon i en virtuell miljö. Det verkliga fordonet observerar den virtuella motsvarigheten och agerar autonomt för att utföra liknande manövrar. För detta använder det verkliga fordonet en genetisk algoritm för att hitta en rutt som är lik det virtuella fordonets, samtidigt som det undviker eventuella hinder. Longitudinell styrning utförs med en metod inspirerad av adaptiv farthållning (Adaptive Cruise Control) och lateral styrning åstadkoms med en modellprediktiv regulator (MPC). Metoden utvärderas med användartester i simulering, vilket visar att den presterar bättre än en metod med direktstyrning under inverkan av stora fördröjningar. I fall där fördröjningarna är små presterar metoden likvärdigt med eller något sämre än direktstyrning, men testdeltagarna rapporterade ändå större upplevd trygghet med den nya metoden som de också föredrog.

Acknowledgements

I would like to express my sincere gratitude to Professor Jonas Mårtensson and Frank Jiang at the Division of Decision and Control Systems at KTH Royal Institute of Technology. Their guidance, extensive knowledge and intriguing ideas have been invaluable throughout this research.

Stockholm June 2020
Anton Björnberg

Contents

Nomenclature

ADS	Automated Driving System
CT	Control Tower
GPS	Global Positioning System
IMU	Inertial Measurement Unit
LIDAR	LIght Detection And Ranging
MPC	Model Predictive Control
Operator	A trained human that performs the teleoperation task
PI controller	Proportional-integral controller
RADAR	RAdio Detection And Ranging
Teleoperation	Operating at a distance
Telerobot	The robotic agent that is teleoperated
TR	Telerobot
VV	Virtual Vehicle

Chapter 1

Introduction

1.1 Automated Driving Systems

The development of autonomous vehicles or Automated Driving Systems (ADS) has progressed rapidly in recent years. The Society of Automobile Engineers (SAE) defines the level of vehicle autonomy in terms of driver involvement on a scale from 0 to 5 [1]. At level 0 the human driver performs all driving tasks, while at level 5 no human attention is needed and the vehicle can handle all driving tasks under any condition.

Highly autonomous prototype vehicles, at minimum level 3, have been demonstrated by Waymo [2], Uber [3], Scania [4], Volvo [5], Einride [6] and several others [7]. At the same time, most modern cars are now equipped with partially autonomous capabilities corresponding to level 1 or 2 autonomy for assisting in specific tasks, such as lane keeping, cruise control and automated parking [8].

The pursuit of ADS technology is motivated by several predicted societal benefits. Examples include improved traffic safety [9], reduced road congestion [10] and improved mobility for elderly or disabled people [11]. It can also lead to reduced traffic emissions, reduced transport costs and improved fuel efficiency [12], which are important in tackling the ongoing climate change.

1.2 Teleoperation

Despite the rapid advancements, full level 5 autonomy is predicted to be many years away. Automation has historically been successful in highly predictable environments for accomplishing repetitive tasks such as in manufacturing, telephone switching or aircraft control. Driving in normal traffic however offers high variability and uncertainty, and it might be an impossible endeavor to properly prepare an ADS for every possible event while not compromising safety and traffic robustness [8, 13, 14].

An illustrative case is an ADS that encounters a fallen tree blocking the lane on a bi-directional two-lane road. If safety measures prohibits the ADS from either driving off-road or into the opposing lane, the autonomy system will inevitably fail. This is an example of a legal dilemma [15].

Another type of challenging scenario is a complex cultural dilemma, where non-standard means of communication are used to communicate information to drivers [15]. This could for example be a series of cones, signs and other markers near a road construction work to signal road users how to proceed. Other examples are malfunctioning traffic lights, sabotaged traffic signs, or traffic signs communicating contradictory information as well as signs merely intended to be humorous [16].

As is demonstrated in traffic every day however, cases like the ones above pose no greater difficulties for trained human drivers. Thus, until ADSs can handle most scenarios on their own, it will remain a requirement that they be supervised by human drivers who can intervene when necessary. To still reap the benefits of having partially automated systems, several companies propose teleoperation [17,18], to let a human driver operate the (potentially unmanned) automated vehicle *remotely* when it is unable to proceed on its own.

Systems for teleoperating road vehicles are already being tested and a few commercial actors have emerged [19,20]. Typically, the interface contains multiple monitors, a steering wheel and pedals to allow for direct real-time vehicle control over a cellular network. However, teleoperation in this manner brings its own set of challenges as for example situational awareness is limited by available sensor data and communication is affected by network quality. It is currently an open question how to address these inherent drawbacks of teleoperation. In many cases it can be assumed that the automation system is still functional, which opens the possibility for shared control where human and machine collaborates to accomplish the task at hand, in the best manner possible.

1.3 Scope, Contribution and Research Questions

The scope of this thesis is to examine a novel concept for vehicle teleoperation, that utilizes shared control to address mainly the issues caused by network latency and lack of situational awareness. The concept is based on using a virtual environment in which the remote driver (operator) navigates a virtual vehicle to demonstrate what actions are to be executed by the real vehicle (telerobot). The virtual environment would be generated from sensor and map data and information about the movement of the virtual vehicle is sent to the telerobot over a cellular network. It then acts autonomously to mimic the motion of the virtual counterpart.

The work is limited to the aspect of motion control and the interaction between the virtual and the real vehicle. No in-depth considerations for operator interface or perception system is included. To demonstrate the viability of the concept, a system is developed and evaluated in a static environment with the real vehicle being represented by a simulation. The system in question is described in general terms independent of a specific hardware platform and effects that might be introduced by communicating over a cellular network are not investigated in-depth.

The main contribution is the formulation of a teleoperation concept that greatly reduces the dependency on low network latency for good operability. The concept is contextualized by a system that is described in detail with problems unique to this setup being addressed explicitly. This includes limiting vehicle divergence and robustly handling minor operator errors. The work also opens for further research to expand on the concept to promote operability and safety in vehicle teleoperation.

The thesis aims to answer the following question: How can a teleoperation system be designed to enable vehicle teleoperation by utilizing the concept of a virtual environment interface as described above? The question is answered by breaking down the concept into three subproblems, namely longitudinal control, lateral control and path planning. These are treated separately before being combined in a full system implementation that is demonstrated and evaluated in simulation. The usability of the system is assessed by user trials.

1.4 Report Outline

The report is outlined as follows:

Chapter 2 presents relevant background including the current state, challenges and solutions in vehicle autonomy and teleoperation as well as an overview of related work.

Chapter 3 formalizes the concept of teleoperation with a virtual environment interface, breaking it down into three subproblems, which are described and solved separately in detail.

The performance results of the proposed system is presented in chapter 4 along with the results of the user trials.

In chapter 5, the results and their implications are discussed, before the thesis is concluded in chapter 6.

Chapter 2

Background

2.1 Automated Driving Systems

An Autonomous Driving System (ADS) is defined by the Society of Automotive Engineers (SAE) as the hardware and software that is capable to automatically execute dynamic driving tasks at automation level 3-5 [1]. ADSs has been researched since the 1980s and took significant leaps at the DARPA Grand and Urban Challenges between 2004 and 2007 [7]. Since then, research has accelerated and consensus has been reached regarding a general automation system architecture. Typically it can be divided into a perception system and a decision-making system [21].

The perception system uses various sensors such as LIDAR, RADAR, camera, GPS, IMU and odometers to estimate the vehicle's state (pose and linear and angular velocities) as well as an internal representation of the environment. Commonly, the vehicle also has access to offline static maps, which are used to put the state information in context for localization and enhance the internal representation of the environment.

The decision-making system's task is to use the knowledge obtained by the perception system to navigate from an initial position to a goal position. It should also ensure that additional constraints are satisfied, which could mean that the vehicle is not allowed to collide with obstacles, enter forbidden regions or exceed a certain speed. The decision-making system is often organized in a hierarchical structure with top levels being route and path planning, followed by systems for motion planning and obstacle avoidance and finally a controller to execute the intended motion by manipulating the vehicle's various actuators.

Production cars with automation capabilities corresponding to SAE levels 1 and 2 are, at the time of writing, increasingly commonplace with typical features being adaptive cruise control, lane centering, automated parking and collision detection [22]. At level 3 and above, the vehicle requires significantly less driver involvement and Audi A8 claims to be the first production car at level 3 with few competitors [23]. No production cars have currently reached level 4 or 5, but prototypes have been demonstrated by a number of companies [2–7]. Nevertheless, full level 5 autonomy remains a very complex problem to solve and it is unclear when to expected it to be achieved [8, 22].

2.2 Vehicle Teleoperation

Vehicle teleoperation can be defined as *operating a vehicle at a distance* [24], which is here interpreted broadly as the driver not being situated in the vehicle while driving or otherwise operating it. Furthermore, the trained remote driver will be referred to as the *operator* and the vehicle being teleoperated will be called a *telerobot*.

Teleoperation methods comes in different forms and can be distinguished by the required level of operator engagement. The most basic form of teleoperation is *direct control* where the operator maneuvers the telerobot by manually controlling actuator commands such as throttle and steering [24]. On the other hand, a more advanced form of teleoperation is *supervisory control*, where the telerobot has a high level of autonomy and it is sufficient for the operator to issue high-level commands such as "go left" or "go to point X" [24]. Between these extremes is a range of methods for *shared control* where the operator and telerobot collaborates to accomplish the task at hand.

In direct control, a typical setup is a remote driver's seat that resembles a static driving simulator. It is typically equipped with a steering wheel, pedals, various other control input methods and several monitors displaying a panoramic live video stream from the telerobot [25]. This allows for maneuvering the telerobot in the same manner it would be done if the operator was located in the vehicle, but it also brings challenges due mainly to restricted situational awareness and various forms of latencies [16, 25].

During normal driving, a driver obtains situational awareness through visual, aural, haptic and vestibular cues. The visual and aural inputs allows to perceive and predict position and velocity of both the ego vehicle and of other traffic participants [25]. Haptic and vestibular information is important for proprioception, i.e. to understand the behaviour of the vehicle itself and helps to recognize for example acceleration, braking, turning, skidding and collisions [26].

In a teleoperation setting like the one described above, the sensory cues are naturally restricted. Unless the system uses an advanced motion simulator, the operator will rely mainly on visual and aural information, which may lead to degraded proprioception. The quality of the video feed depends on the bandwidth of the communication link and can suffer from reduced frame rate or resolution if the quality-of-service is poor. Additionally, the field-of-view and depth perception can be restricted by using 2D-displays, which increases the cognitive workload and can be tiresome or cause motion sickness. These kinds of restrictions on situational awareness have been seen to decrease teleoperation performance [25, 26].

Network latency is another factor that can degrade teleoperation performance. It can be defined as the time delay between sending a control command and observing the output response [26] and it has been seen that controlled driving is almost impossible when the latency is higher than 300 ms [27]. The magnitude of latency depends on the network type. Typical latencies in a mobile 4G LTE network for video streams are around 50-100 ms [16, 28], but can vary significantly [27]. With 5G technology, latencies are expected to be reduced by a factor 10 as compared to 4G LTE [29].

All in all, direct control in teleoperation comes with several drawbacks. To combat these, it is often suggested to instead use shared or supervisory control. Then, the task of maneuvering the telerobot is fully or partially mitigated from the operator to the telerobot itself, which has some degree of autonomous capabilities. The main challenge with these approaches lies in effectively communicating the operator's goals to the telerobot [24]. A selection of previous studies with proposed methods for shared or supervisory control is presented in the next section.

2.3 Related Work

Space Industry

Telerobotic systems have long been employed in the space industry for extraterrestrial exploration. The distances and consequently the time delays are typically very large, which has spurred significant technological development as summarized in [30]. One of the first teleoperation approaches was to apply direct control via a "move-and-wait" strategy, where commands were discretized and success of execution had to be confirmed before a a new command was issued. Later, direct control was enhanced by the advent of predictive displays and time-delayed force feedback, which offered operators an increased sense of where a system was heading. Teleoperation performance was seen to improve as the prediction models became more complex.

Eventually, the human operator was taken out of the control loop by the introduction of supervisory control. The telerobot was instead sent entire programs that explicitly stated how to accomplish the goal as well as how to handle errors. The main drawback was that writing the instructions was very time consuming and for simple tasks it was often less costly to instead use the older direct control methods.

In modern endeavors such as the exploration of Mars with robotic rovers, time delays are typically around 40 minutes and data transmission can only occur on a few short occasions every day. This makes adding large amounts of software on-the-fly infeasible, and the telerobots have instead been given higher levels of autonomy to independently handle tasks like navigation with obstacle avoidance [31–33].

General Shared Control Concepts with Goal Inference

Contrasting to applications in space, which are typically very static, telerobots often operate in dynamic environments where fluid real-time human-robot interaction is important for successful mission execution. When using supervisory or shared control, the main challenge is therefore to efficiently communicate the operator's goal to the telerobot. One approach is goal inference, where the telerobot actively tries to guess the intentions of the operator. This problem has been formally investigated in [34] and [35] and below are a few notable examples.

In [36], an operator controls a telerobot directly, while the telerobot infers the operator's intended objective from the control commands and can apply corrections if new commands deviate from the currently inferred goal. The inference is performed using reachability set analysis with linear temporal logic and the method is verified in experiments in the context of parking a vehicle in one of several free parking spots.

Another shared control approach is presented in [37] where the telerobot uses information gathering to actively learn the operator's intentions. This is done by having the telerobot perform probing actions and observe the operator's reactions via control inputs. This active learning behaviour is balanced with a goal-oriented behaviour, which prioritizes completing an inferred goal over exploring other potential goals. The method was evaluated with a Fetch robot and it was found that performance was increased in comparison to using direct control. However, testers reported that the shared control method posed a higher workload. A similar strategy was investigated for cars in [38].

Several methods for shared control are compared in [39], which in particular found that the superior method was a real-time sample based motion planner based on the RRT algorithm [40]. Drawbacks of the method was that it introduced loss of control, planning delays and suboptimal motions, but the users expressed no complaints over those issues.

Shared Control for Road Vehicles

Many studies have investigated using shared control for road vehicles specifically. A few examples are summarized below.

An infrastructure concept that supports large scale teleoperation of traffic systems is explored in [15]. A central element is the *control tower* from which human operators can supervise and teleoperate connected vehicles. It is argued that control towers are necessary as exception-handling layers to enable safe operation of ADSs in uncertain safety-critical situations. This notion of a control tower will be repeatedly referenced throughout this thesis.

A supervisory control approach with interactive path planning in urban environments is proposed in [41]. When the autonomous telerobot is uncertain of how to proceed, it can generate a small number of feasible routes using an RRT algorithm with route clustering and let the operator decide which one to pursue. The method could also be adapted for shared control, by using steering wheel force feedback to assist the operator in stabilizing the vehicle to one of the proposed paths. The method is seen to reduce operator workload in comparison to direct control and also overcomes the issues with delay. However, it is limited to situations where only a few route options exist.

The method in [42] assists the operator by using an efficient operator interface with 3D maps and time delay compensation. 3D maps are created in advance and are presented to the operator as a virtual 3D environment along with a virtual representation of the telerobot itself. Time-delay is measured and compensated for by displaying the telerobot in a predicted state. Additionally, the virtual environment is enhanced to display any obstacles that the telerobot detects. The operator operates in a direct control manner, but the telerobot is 'semiautonomous' and control is sent as waypoints rather than explicit speed commands and steering angles. Evaluation indicates that this interface allowed an operator to complete a given course faster and with fewer collisions compared to direct control.

In [43] an active safety system is proposed. This takes into account time-delay by predicting trajectories of dynamic objects in the surroundings and reacting through speed control if a hazard is detected. This can occur before the operator becomes aware of the problem, The operator is informed about the intervention and after the time-delay has the option of overriding the telerobot's maneuver and is thereby the main decision maker.

In cases when connection between operator and telerobot is completely lost, a solution is proposed by [44], where a safe emergency route is continuously computed and presented to the operator in the form of a "Free corridor".

Chapter 3

Methodology

This chapter describes the proposed teleoperation method. First, an overview of the architecture and main ideas is presented, followed by a brief description of the operator interface with the virtual environment. The subsequent sections details the subproblems of longitudinal control, lateral control and path planning.

3.1 Concept Overview

The teleoperation method proposed in this thesis uses a shared control approach that is somewhat inspired by the virtual interface approach presented in [42]. However, in their work the virtual vehicle attempted to match the state of the real vehicle, whereas here the goal is for the telerobot to match the virtual counterpart. For brevity, the virtual vehicle and the physical telerobot will be abbreviated VV and TR respectively.

Figure 3.1 illustrates the concept and Figure 3.2 shows the proposed system architecture. The structure at the operator end is referred to as the Control Tower (CT) in reference to [15] and houses the operator and the virtual environment with VV, which is controlled by means of direct control. By running the virtual environment locally at CT, the operator will not experience any time-delay during teleoperation. The state of VV is continuously sent to TR via a communication link such as a cellular network.

TR contains submodules for perception, path planning, and motion control. The perception module is not examined in this work, but is assumed to exist and provide information about the state of TR and the environment with negligible noise. This information is communicated to CT for visualization in the virtual environment. Also, the path planner uses the perception information together with knowledge of the state of VV to perform local path planning.

In this work, focus is on the aspect of motion control and the interaction between TR and VV. It will therefore mainly concern the motion controller and path planner components shown in Figure 3.2. Local path planning is accomplished with a genetic algorithm that runs continuously on TR and generates path segments that connect the current the position of TR with a desired goal position close to VV. A global path for TR to follow is built gradually by stitching together each new segment with a previously planned global path. The method is described in detail later in this chapter.

The motion controller observes the global path and the states of TR and VV to determine throttle and steering to achieve a desired motion. In this work, the problems of longitudinal (velocity) and lateral (steering) control are separated and treated individually as outlined in the next sections. In the proposed motion controller implementation, throttle and steering are decided in sequence in the same process and applied to TR simultaneously at the end of every iteration.

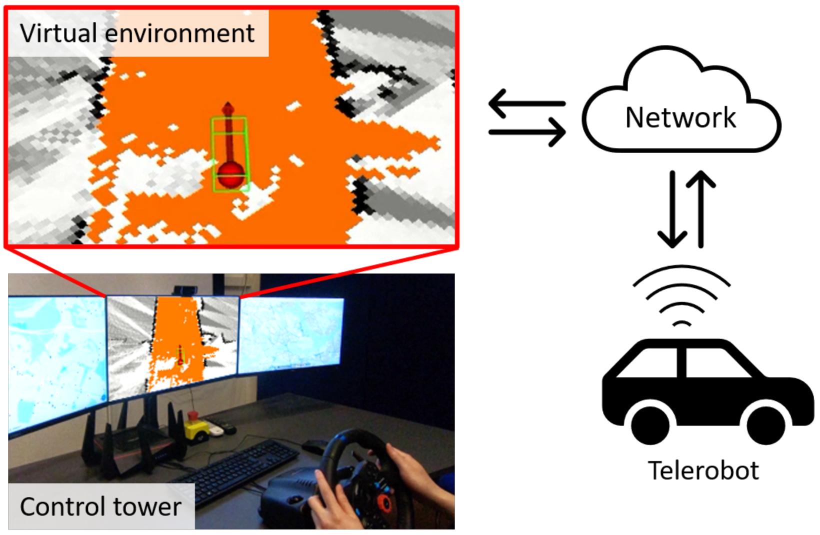

Figure 3.1: Illustration of the concept, where an operator in a control tower controls a virtual vehicle (green wireframe cuboid) in a virtual environment that runs locally and the movements are sent to the telerobot via a communication network. The bottom left figure is a montage with an image taken from [36].

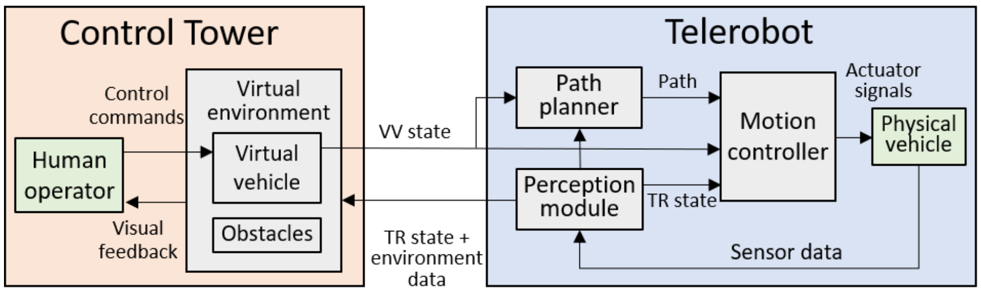

Figure 3.2: Proposed system architecture

3.2 Virtual Environment

The purpose of introducing the virtual environment is to split the typical time-delayed direct control loop (Figure 3.3) into two local control loops (Figure 3.2). At CT, the operator controls VV locally in a manner similar to a video game or a driving simulator with no regards given to the actual motion of TR. TR on the other hand operates only with regards to the planned path which grows dynamically as VV progresses.

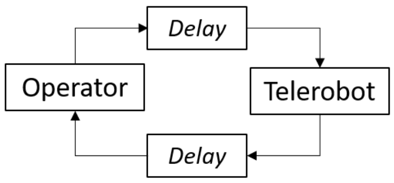

Figure 3.3: Schematic view of time-delays during typical direct control teleoperation

In order to drive safely, the virtual environment should be updated continuously to resemble the physical environment as observed by TR's perception system. Throughout this work however, the environment is assumed to be static and is represented by a gridmap that is known a priori. Furthermore, the implemented system uses the visualization package RVIZ [45] for the virtual environment. The operator is thus presented a view as shown in Figure 3.4 where the perspective is fixed behind the wireframe cuboid that represents VV.

With this setup, teleoperation is performed by having TR execute motion similar to that

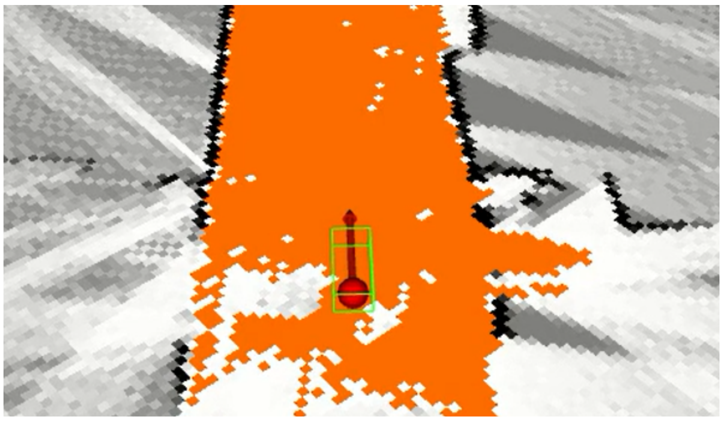

Figure 3.4: Virtual environment using RVIZ

of VV. To make the endeavor feasible, it is important that the vehicles have similar motion constraints such as dynamics and maximum steering angles. Here, VV is therefore a simulation based on the bicycle model as derived in [46], with longitudinal dynamics approximated by a first order system. The dynamics are given by:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{v} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \frac{1}{\tau}(u-v) \\ \frac{v\tan\delta}{L} \end{bmatrix}$$

Where x, y are the Cartesian coordinates, v is the longitudinal velocity and θ is the yaw angle. The entities u and δ are control variables for velocity and steering angle respectively and τ and L are model parameters corresponding to a time constant and the vehicle's wheelbase. Furthermore, u and δ are limited to $u \in [u_{\min}, u_{\max}]$ and $\delta \in [\delta_{\min}, \delta_{\max}]$, to match the constraints of the physical system.

To control VV, the operator inputs the steering angle δ and a desired velocity v_r that in turn acts as input to a controller that generates the throttle signal u that brings the velocity of VV to v_r. The operator control interface could take the form of a setup with a physical steering wheel and pedals as in typical direct control implementations. For the scope of this work however, the operator is able to input control by using a simple computer mouse interface.

3.3 Longitudinal Control

The longitudinal control problem concerns the control of TR's throttle to achieve a certain desired velocity. An important aspect is that TR should maintain a certain separation to VV and should neither be outpaced by nor overtake VV. It is found that a method inspired by Adaptive Cruise Control (ACC) is suitable for these conditions. Only forward motion is considered, but the method can be extended to include reversing maneuvers as well.

Adaptive Cruise Control (ACC)

ACC is a common feature in modern production cars and enables a car to automatically maintain a steady spacing to the vehicle proceeding it. A general overview is provided in [47, Chapter 9]. Figure 3.5 illustrates the ACC problem in relation to a planned path, which the vehicles may deviate slightly from. The objective is to ensure that TR is close to the green target point, which is located a distance s_r behind VV, as measured along the path. The reference spacing s_r can be chosen to depend linearly on the follower vehicle's velocity such that

$$s_r = s_0 + t_h v_T$$

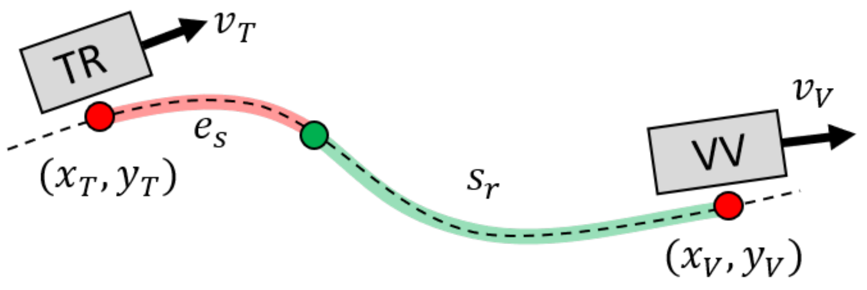

Figure 3.5: The ACC problem in the context of teleoperation with a virtual environment interface. The dashed curve represents a planned path, which the vehicles may deviate slightly from as indicated. The red points are the points on the path closest to each vehicle and the green middle point marks the target position. The reference spacing s_r is the length of the green segment and the control error e_s is the deviation in position from the target point as measured along the path.

where s_0 is a constant minimum separation, t_h is a constant time-headway and v_T is the velocity of TR. Here, $s_0 = 0$ to ensure that the vehicles converge to the same point when stopping. The control error e_s to be minimized is the deviation in position from the target point as measured along the path. Furthermore, s denotes the full spacing between the vehicles such that:

$$s = e_s + s_r$$

The control variable is a bounded throttle command $u_T \in [-\bar{u}_T, +\bar{u}_T]$ and the complete control loop is shown in Figure 3.6. A similar controller structure is used for longitudinal control at VV. The reference in that case is however the desired velocity v_r, and the controlled variable is the velocity v_V.

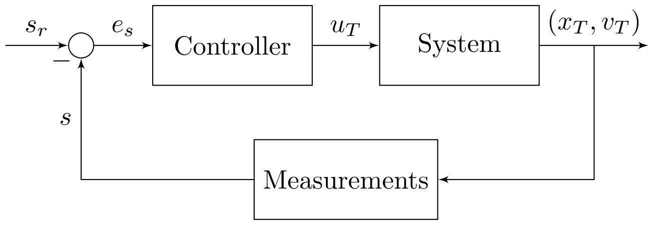

Figure 3.6: Longitudinal controller structure of TR

PI Controller

The controller block in Figure 3.6 is proposed to be a PI (Proportional-Integral) controller, which can be written on standard form as

$$u(t) = K\left(e(t) + \frac{1}{T_i}\int_0^t e(s)ds\right)$$

Here, $e(t) = r(t) - y(t)$ is the error between a general reference $r(t)$ and an output $y(t)$. The parameters are the gain K and the integration time T_i, which can be interpreted as the time in which the sum of all past errors are to be compensated for.

A discrete-time approximation that takes into account the sampling period is given in [48] and can be written

$$u(t) = P(t) + I(t)$$

with the terms defined as

$$P(kh) = K(br(kh) - y(kh))$$
$$I(kh) = I(kh - h) + \frac{Kh}{T_i}(r(kh - h) - y(kh - h))$$

where h is the sampling period and b is the fraction of r to use for the proportional part.

Due to the throttle signal being bounded, a saturation is also included in the controller. If the PI output is denoted $\tilde{u}$ the output control signal can be written

$$u = \text{sat}(\tilde{u}|\, \bar{u})$$

with the saturation function being defined for $y > 0$ as:

$$\text{sat}(x|\, y) = \begin{cases} -y & x < -y \\ x & -y \leq x < y \\ y & x \geq y \end{cases}$$

Integral windup is avoided by not incrementing the integral term if $|u(kh - h)| \geq \bar{u}$.

Managing Divergence

Special measures must be taken to ensure that the vehicles do not diverge, which could occur if VV has a higher velocity than TR can achieve. If the maximum reachable velocity $v_{\max}$ is known, divergence can be avoided by limiting the maximum velocity of VV. This can be achieved by saturating v_r before applying the PI controller for VV such that the PI input becomes

$$\tilde{v}_r = \text{sat}(v_r|\, \delta_v v_{\max})$$

where the parameter $\delta_v \in (0, 1]$ is used to allow TR to have a higher velocity than VV to "catch up" if the separation has become too large.

In general however, $v_{\max}$ is not known and may be varying due to external conditions such as road inclination or varying battery voltage. Therefore, $v_{\max}$ must be estimated continuously and the method proposed here goes by the following intuition: *It is only necessary to know $v_{\max}$ when v_T approaches $v_{\max}$, which occurs when u_T approaches $\bar{u}_T$.*

This means that $v_{\max}$ can be determined by directly observing the velocities that are achieved with maximum throttle $u_T = \bar{u}_T$. In the case of less extreme u_T, $v_{\max}$ can be predicted by approximating the longitudinal vehicle dynamics as a simple first order system

$$\dot{v} = cu - dv$$

where c and d are model parameters. With constant throttle u, the steady-state velocity is

$$v_{ss} = \frac{c}{d}u = K_{ss}u$$

and the steady state gain K_{ss} can be estimated from v_T and u_T as

$$K_{ss} \approx \frac{v_T}{u_T}.$$

This allows for predicting $v_{\max}$, which essentially is the steady state velocity at $u_T = \bar{u}_T$. The estimated maximum velocity $v_{\max}$ is therefore given by

$$v_{\max} = \begin{cases} \frac{v_T}{u_T}\bar{u}_T & \text{If } u_T > \delta_u \bar{u}_T \\ v_0 & \text{Otherwise} \end{cases}$$

where the parameter $\delta_u \in (0, 1]$ is used to specify within which range u_T must be for estimation to occur and v_0 is a (large) value to apply when velocity restriction is not needed. Note especially that as $u_T \to \bar{u}_T$ the estimation converges to the directly observed velocities.

A low-pass filter is furthermore applied in order to avoid discontinuities in the estimation.

The estimation is performed on TR and the result is communicated to the CT for VV to account for.

Summary of Parameters

The parameters governing the longitudinal control are summarized in Table 3.1.

Symbol	Description
t_h	Time-headway
$\bar{u}_T, \bar{u}_V$	Maximum throttle commands for TR and VV respectively
h_T, h_V	Sampling periods for TR and VV
K_T, K_V	PI gains for each controller
b_T, b_V	Fraction of the PI reference r to use for computing the P term
$T_{i,T}, T_{i,V}$	Integration time for each controller
δ_v	Fraction of $v_{\max}$ that determines the maximum allowed velocity of VV
δ_u	Fraction of $\bar{u}_T$ required to for $v_{\max}$ estimation to occur

Table 3.1: Longitudinal control parameters

3.4 Lateral Control

Lateral control concerns the control of the steering angle δ that along with the longitudinal control enables TR to track a given path. It is important that the path is tracked with high precision and accuracy, and a technique that often has this quality and is commonly applied for lateral control is Model Predictive Control (MPC).

Model Predictive Control (MPC)

MPC is a control method that relies on a dynamic model of the process to be controlled and determines the control signal in every time step by optimizing a predicted outcome. The optimization is performed on-line with regards to a given cost function and typically results in a sequence of optimal control signals. However, only the first element of the sequence is applied to the system. The MPC formulation for lateral control presented below is largely inspired by [49].

For clear notation, let $t = 0$ mark the time when the optimization is initiated. To account for the computation time required for the optimization, this method takes as input the *current* error $e(0)$ and steering angle $\delta(0)$. The latter is assumed to be constant over the sampling period (zero-order-hold). For $t > 0$, $e(t)$ is a predicted control error and $\delta(t)$ a future steering angle. The output is the steering angle $\delta(h)$ to be applied in the *next* time step, with h being the sampling period.

The optimization problem to be solved in every time can be written:

$$\begin{aligned}
\bar{\delta}^* = \arg\min_{\bar{\delta}} \quad & E^T Q E + D^T R D && (3.1a)\\
\text{subject to} \quad & e(kh+h) = f\big(e(kh), \delta(\min\{k, n_c\} \cdot h)\big), \quad k \in \{0, 1, ..., n_p\} && (3.1b)\\
& e(0) = e_0, && (3.1c)\\
& \delta(0) = \delta_0, && (3.1d)\\
& \bar{\delta} \subseteq \mathcal{D} && (3.1e)
\end{aligned}$$

Where

- $\bar{\delta} = \big(\delta(h), ..., \delta(n_c h)\big)$ is a sequence of n_c steering angles to be computed. n_c is the *control horizon*. $\bar{\delta}^*$ denotes the optimal sequence with $\delta^*(h)$ being the controller output.
- $e(t) = \begin{bmatrix} e_1(t) & \dot{e}_1(t) & e_2(t) & \dot{e}_2(t) \end{bmatrix}^T$ is an error vector, which is defined in detail in the next sections.
- E and D are column vectors defined, with *prediction horizon* $n_p \geq n_c$, as

$$E = \begin{bmatrix} e(h) \\ e(2h) \\ \vdots \\ e(n_p h) \end{bmatrix}, \quad D = \begin{bmatrix} \delta(h) - \delta(0) \\ \delta(2h) - \delta(h) \\ \vdots \\ \delta(n_c h) - \delta(n_c h - h) \end{bmatrix}$$

- Q and R are diagonal weight matrices of dimension $4n_p \times 4n_p$ and $n_c \times n_c$ respectively defined as

$$Q = \begin{bmatrix} \bar{q} & 0 & \dots & 0 \\ 0 & \bar{q} & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & \bar{q} \end{bmatrix}, \text{ where } \bar{q} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & q & 0 \\ 0 & 0 & 0 & q \end{bmatrix} \text{ and } R = \begin{bmatrix} rn_p & 0 & \dots & 0 \\ 0 & rn_p & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & rn_p \end{bmatrix}$$

  such that q and r can be used to balance between different errors and control input respectively.
- $f(e, \delta)$ represents a discrete-time system model that predicts the errors in the next sampling instant.
- $\mathcal{D} = [\delta_{\min}, \delta_{\max}]$ is the range of admissible steering angles.
- e_0 and δ_0 are current measured errors and applied steering angle respectively.

This quadratic optimization problem is solved on-line using the Python package CVXPY [50]. For real-time performance n_p and n_c are limited to small values.

Control Errors

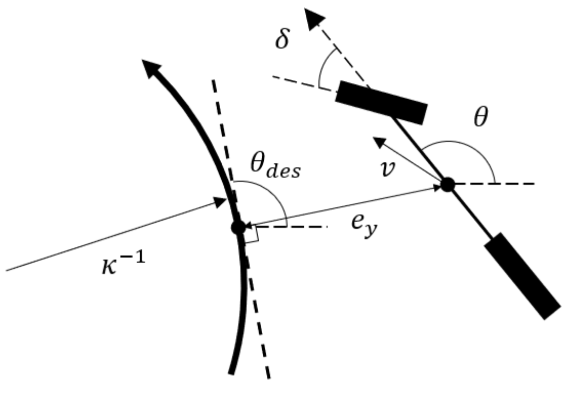

Figure 3.7: The lateral control situation, where the vehicle is approximated with a bicycle model.

The lateral control errors are defined with respect to a given path that the vehicle is to be stabilized to. The situation is illustrated in Figure 3.7 where the vehicle is approximated with a bicycle model. The arched arrow represents the path to track and is assumed to have constant curvature κ. The control objective is to minimize the yaw offset $|e_\theta| = |\theta - \theta_{\text{des}}|$ and the lateral offset $|e_y|$, which can be computed as outlined below.

Let the location of TR be $r_T = (x_T, y_T)$ and the closest point on the path $r_p = (x_p, y_p)$ (using a global coordinate system). The desired yaw angle at r_p is θ_{des}, which is tangent to the path. Then, e_y is defined to be the lateral component (in relation to the path) of the separation vector $r = r_T - r_p$. The lateral component is parallel to the unit vector $\hat{t} = (-\sin\theta_{\text{des}}, \cos\theta_{\text{des}})$ and the lateral offset can thereby be computed with a scalar product like

$$e_y = \hat{t} \cdot r = (y_T - y_p)\cos\theta_{\text{des}} - (x_T - x_p)\sin\theta_{\text{des}}.$$

Lateral Dynamic Model

The MPC in this work uses the linear lateral dynamic model derived in [46, chapter 2]. This is based on a 2-degrees-of-freedom bicycle model, which has been shown to behave similarly to a more complex 6-degrees-of-freedom model under normal conditions [51]. It is expressed in terms of a state vector $e = \begin{bmatrix} e_1 & \dot{e}_1 & e_2 & \dot{e}_2 \end{bmatrix}^T$ where the elements are computed from the control errors and their derivatives. With v_x denoting the longitudinal velocity of TR they are defined as:

$$\begin{aligned} e_1 &\equiv e_y \\ \dot{e}_1 &\equiv \dot{e}_y + v_x \dot{e}_\theta \\ e_2 &\equiv e_\theta = \theta - \theta_{\text{des}} \\ \dot{e}_2 &\equiv \dot{e}_\theta \end{aligned}$$

The model can be written:

$$\begin{aligned} \dot{e}(t) = \frac{d}{dt}\begin{bmatrix} e_1(t) \\ \dot{e}_1(t) \\ e_2(t) \\ \dot{e}_2(t) \end{bmatrix} &= \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\frac{4C_\alpha}{mv_x} & \frac{4C_\alpha}{m} & \frac{2C_\alpha(l_r - l_f)}{mv_x} \\ 0 & 0 & 0 & 1 \\ 0 & \frac{2C_\alpha(l_r - l_f)}{I_z v_x} & \frac{2C_\alpha(l_f - l_r)}{I_z} & -\frac{2C_\alpha(l_f^2 + l_r^2)}{I_z v_x} \end{bmatrix} \begin{bmatrix} e_1(t) \\ \dot{e}_1(t) \\ e_2(t) \\ \dot{e}_2(t) \end{bmatrix} \\ &+ \begin{bmatrix} 0 \\ \frac{2C_\alpha}{m} \\ 0 \\ \frac{2l_f C_\alpha}{I_z} \end{bmatrix} \delta(t) + \begin{bmatrix} 0 \\ \frac{2C_\alpha(l_r - l_f)}{mv_x} - v_x \\ 0 \\ -\frac{2C_\alpha(l_f^2 + l_r^2)}{I_z v_x} \end{bmatrix} \dot{\theta}_{\text{des}} = Ae(t) + B\delta(t) + C\dot{\theta}_{\text{des}} \end{aligned} \tag{3.2}$$

where the parameters are defined like the following:

- C_α: Cornering stiffness for the each tire,
- $l_f, l_r,$: Distance between center of gravity (CG) and front and rear tires respectively,
- m : the mass of the vehicle,
- I_z : Moment of inertia about the z-axis,
- v_x : Longitudinal velocity, assumed constant
- $\dot{\theta}_{\text{des}} \equiv v_x \kappa$: Desired yaw rate taking into account the (assumed constant) curvature of the path.

Discretized Dynamic Model

In order to use (3.2) within the MPC, it must be expressed in discrete-time. This can be done without introducing approximations with the method outlined in [48, Chapter 1]. For clarity, the state vector e is in this section denoted ε, to not be confused with the mathematical constant e.

Let h be the sampling period and assume that, at the kth sampling instant, the state $\varepsilon(kh)$ is available and that $\delta(t)$ and $\dot{\theta}_{\text{des}}$ are constant over $kh \leq t \leq kh + h$ (zero-order-hold). Over this interval, the model (3.2) can be rewritten:

$$\dot{\varepsilon}(t) = A\varepsilon(t) + B\delta(t) + C\dot{\theta}_{\text{des}} \tag{3.3a}$$
$$= A\varepsilon(t) + b(kh) \tag{3.3b}$$

where $b(kh) = B\delta(kh) + C\dot{\theta}_{\text{des}}$ is constant such that (3.3) can be regarded a first order ordinary differential matrix equation, with the solution given by

$$\varepsilon(t) = e^{A(t-kh)}\varepsilon(kh) + e^{At}b(kh)\int_{kh}^{t} e^{-As'}ds'$$

where e^X denotes a matrix exponential defined like

$$e^X = \sum_{n=0}^{\infty}\frac{1}{n!}X^n \text{ with } X^0 = I.$$

Substituting variables in the integral with $s = t - s'$ gives

$$\varepsilon(t) = e^{A(t-kh)}\varepsilon(kh) + b(kh)\int_{0}^{t-kh} e^{As}ds$$

and setting $t = kh + h$ yields the discretized system:

$$\begin{aligned}\varepsilon(kh+h) &= e^{Ah}\varepsilon(kh) + b(kh)\int_0^h e^{As}ds \\ &= \Phi(h)\varepsilon(kh) + \Gamma(h)b(kh)\end{aligned}$$

As noted in [48] it follows that $\Phi(t)$ and $\Gamma(t)$ satisfy

$$\frac{d}{dt}\begin{bmatrix}\Phi(t) & \Gamma(t)\\ 0 & I\end{bmatrix} = \begin{bmatrix}\Phi(t) & \Gamma(t)\\ 0 & I\end{bmatrix}\begin{bmatrix}A & b\\ 0 & 0\end{bmatrix}$$

and can be computed as the sub-matrices of a matrix exponential

$$\begin{bmatrix}\Phi(t) & \Gamma(t)\\ 0 & I\end{bmatrix} = \exp\left(\begin{bmatrix}A & b\\ 0 & 0\end{bmatrix}t\right),$$

which is often available for direct computation in scientific computing software like Matlab and Scipy.

Summary of Parameters

The parameters governing the lateral control are summarized in Table 3.2.

Symbol	Description
h	Sampling period
n_p, n_c	Prediction and control horizons
q	Optimization weight to balance between errors in positions and orientation
r	Optimization weight to balance between control errors and control input size
C_α	Cornering stiffness
l_f, l_r	Position of front and rear tires relative to the center of gravity
m	Vehicle mass
I_z	Vehicle moment of inertia

Table 3.2: Lateral control parameters.

3.5 Path Planner

The overall objective for TR is to mimic the motion of VV. However, it is not guaranteed that blindly following the path of VV results in motion that is safe and efficient. One reason is that obstacles, which were not present for VV and therefore not considered by the operator, could appear for TR in a dynamic environment. Another risk is suboptimal maneuvering by the operator due to for example erroneous scene representation or by mistake. This motivates the inclusion of a dynamic path planner that continuously generates a path for TR to track as VV progresses.

The planned path must be feasible to track and is therefore required to be collision-free and not exceed a maximum admissible curvature as imposed by motion constraints. Additional soft constraints are that it should be short and similar to the path taken by VV. The method proposed here relies on a genetic algorithm that generates path segments that account for the constraints and join smoothly to gradually build the full path of TR. Paths are represented by Bézier curves, which are described in the following sections followed by a detailed description of the genetic algorithm.

Bézier Curves Overview

Bézier curves are convenient as path representations as they are easily parameterized to pass through given waypoints with imposed constraints on curvature and tangents as described in detail in [52]. A summary of relevant characteristics is given below.

A Bézier curve of degree n is defined in terms of $n+1$ control points $(c_0, ..., c_n)$, $c_i = (x_i, y_i)$ and is given by

$$P_b(t) = (x_b(t), y_b(t)) = \sum_{i=0}^{n} B_i^n(t) c_i, \quad t \in [0, 1],$$

where $B_i^n(t)$ is a Bernstein polynomial defined as:

$$B_i^n(t) = \binom{n}{i} (1-t)^{n-i} t^i, \quad i \in \{0, 1, ..., n\}.$$

The curve's first and second order derivatives can be computed from the control points as

$$\dot{P}_b(t) = (\dot{x}_b(t), \dot{y}_b(t)) = n \sum_{i=0}^{n-1} B_i^{n-1}(t)(c_{i+1} - c_i)$$

$$\ddot{P}_b(t) = (\ddot{x}_b(t), \ddot{y}_b(t)) = n(n-1) \sum_{i=0}^{n-1} B_i^{n-1}(t)(c_{i+2} - 2c_{i+1} + c_i)$$

and it follows that the derivatives at the end points are

$$\dot{P}_b(0) = n(c_1 - c_0) \tag{3.5a}$$

$$\dot{P}_b(1) = n(c_n - c_{n+1}) \tag{3.5b}$$

$$\ddot{P}_b(0) = n(n-1)(c_2 - 2c_1 + c_0) \tag{3.5c}$$

$$\ddot{P}_b(1) = n(n-1)(c_n - 2c_{n-1} + c_{n-2}). \tag{3.5d}$$

The curvature at t is given by

$$\kappa(t) = \frac{|\dot{x}(t)\ddot{y}(t) - \dot{y}(t)\ddot{x}(t)|}{(\dot{x}^2(t) + \dot{y}^2(t))^{3/2}}.$$

Several Bézier curves can therefore be joined with continuous curvature at a shared endpoint if their first and second order derivatives are equal at the point.

Bézier Curves for Path Planning

The path planner uses two smoothly joined Bézier curves to represent a path segment. The first curve, $P(t)$, is required to join an existing path W at an attachment point and orientation $z_0 = (x_0, y_0, \theta_0)$ where the first and second order derivatives are w' and w''. Next, $P(t)$ should join with the second curve $Q(t)$ at an intermediate pose $z_1 = (x_1, y_1, \theta_1)$.

$Q(t)$ should finally terminate at $z_2 = (x_2, y_2, \theta_2)$.

Continuous curvature can be ensured if $P(t)$ and $Q(t)$ each is constructed from 5 control points, denoted p_i and q_i, $i \in \{0, ..., 4\}$, that are placed to satisfy the following constraints that follow from (3.5):

$$\begin{aligned}
P_b(0) &= (x_0, y_0) && \implies p_0 = (x_0, y_0) && (3.6a)\\
\dot{P}_b(0) &= w' && \implies 4(p_1 - p_0) = w' && (3.6b)\\
\ddot{P}_b(0) &= w'' && \implies 12(p_2 - 2p_1 + p_0) = w'' && (3.6c)\\
P_b(1) &= (x_1, y_1) && \implies p_4 = (x_1, y_1) && (3.6d)\\
Q_b(0) &= (x_1, y_1) && \implies q_0 = (x_1, y_1) && (3.6e)\\
Q_b(1) &= (x_2, y_2) && \implies q_4 = (x_2, y_2) && (3.6f)\\
\dot{P}_b(1) &= \dot{Q}_b(0) && \implies 4(p_4 - p_3) = 4(q_1 - q_0) && (3.6g)\\
\ddot{P}_b(1) &= \ddot{Q}_b(0) && \implies 12(p_4 - 2p_3 + p_2) = 12(q_2 - 2q_1 + q_0) && (3.6h)\\
\dot{P}_b(1) &\propto (\cos\theta_1, \sin\theta_1) && \implies p_3 = p_4 - \alpha(\cos\theta_1, \sin\theta_1) && (3.6i)\\
\dot{Q}_b(0) &\propto (\cos\theta_1, \sin\theta_1) && \implies q_1 = q_0 + \gamma(\cos\theta_1, \sin\theta_1) && (3.6j)\\
\dot{Q}_b(1) &\propto (\cos\theta_2, \sin\theta_2) && \implies q_3 = q_4 - \beta(\cos\theta_2, \sin\theta_2) && (3.6k)
\end{aligned}$$

It follows that the points must have the following positions:

$$\begin{aligned}
p_0 &= (x_0, y_0) && && (3.7a)\\
p_1 &= \frac{w'}{4} + p_0 && \big(w' \propto (\cos\theta_0, \sin\theta_0)\big) && (3.7b)\\
p_2 &= \frac{w''}{12} + 2p_1 + p_0 && && (3.7c)\\
p_3 &= p_4 - \alpha(\cos\theta_1, \sin\theta_1) && && (3.7d)\\
p_4 &= (x_1, y_1) && && (3.7e)\\
q_0 &= (x_1, y_1) && && (3.7f)\\
q_1 &= q_0 + \alpha(\cos\theta_1, \sin\theta_1) && (\alpha = \gamma) && (3.7g)\\
q_2 &= p_2 + 4\alpha(\cos\theta_1, \sin\theta_1) && && (3.7h)\\
q_3 &= q_4 - \beta(\cos\theta_2, \sin\theta_2) && && (3.7i)\\
q_4 &= (x_2, y_2) && && (3.7j)
\end{aligned}$$

The parameters α and β can be used to adjust inner curvature by altering the positions of p_3, q_1 and q_3 as illustrated in Figure 3.8.

A curve segment consisting of two Bézier curves in this way is completely defined by the tuples $\mathcal{W} = (z_0, w', w'')$ and $\phi = (z_1, z_2, \alpha, \beta)$ and can be translated into a sequence of points by sampling the associated segments $P(t)$ and $Q(t)$ at appropriate t. However, having equidistant samples requires knowing the segment lengths, which can be approximated from a presampling of each segment. The kth point of segment $P(t)$ when presampled into N'_P points is for example given by:

$$r_k = (x_k, y_k) = P\left(\frac{k}{N'_P - 1}\right), \quad h \in \{0, 1, ..., N_P - 1\}$$

The segment length can be approximated as

$$L_P \approx \sum_{i=0}^{N'_p - 2} \|r_{i+1} - r_i\|$$

and points can be spaced approximately l apart by choosing for the final sampling

$$N_P = \frac{L_P}{l} + 1.$$

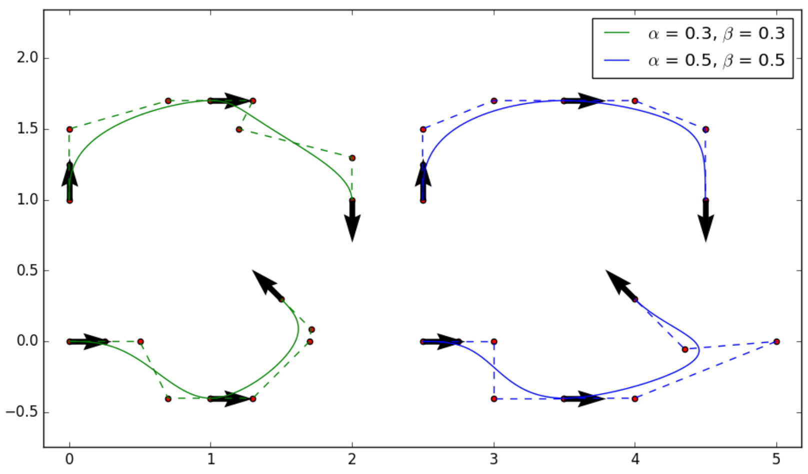

Figure 3.8: Path examples with the control points shown. The curves next to each other are defined by identical waypoints z_0, z_1, z_2 as indicated by the arrows, but with different values for α, β.

Genetic Algorithm

A genetic algorithm can be described as a function optimizer that draws inspiration from Darwinian evolution. Several studies have proposed genetic algorithms as methods for path planning in different settings [53–55] and a thorough theoretical background can be found in [56]. In summary it consists of the following steps:

1. **Initialization**
   An initial set of solution candidates, a *population*, is generated.
2. **Evaluation**
   Every solution candidate in the population is evaluated in terms of its performance and assigned a *fitness* value. High performance correlates with high fitness.
3. **Selection**
   The candidates are "selected" i.e. sampled randomly into an intermediate population. This is meant to resemble the process of natural selection and candidates with high fitness are more likely to be selected.
4. **Recombination**
   The candidates in the intermediate population are recombined to generate offspring solutions that share features with several parent solutions. This is omitted in this work.
5. **Mutation**
   Each solution candidate in the intermediate population has a probability of undergoing mutation, which is a random change in the individual's genetic material i.e. the parameters defining the path.

Steps 2 to 5 are repeated until a termination condition is met. A termination condition could be expressed as a fixed number of iterations or in terms of the performance convergence rate. In the next sections each step is described in greater detail.

Initialization

A path candidate is parametrized by the tuples $\mathcal{W} = (z_0, w', w'')$ and $\phi = (z_1, z_2, \alpha, \beta)$ as shown in the previous section. The start point conditions conveyed by $\mathcal{W}$ remain constant over the solution candidates, but ϕ may vary and is used to identify each candidate. The

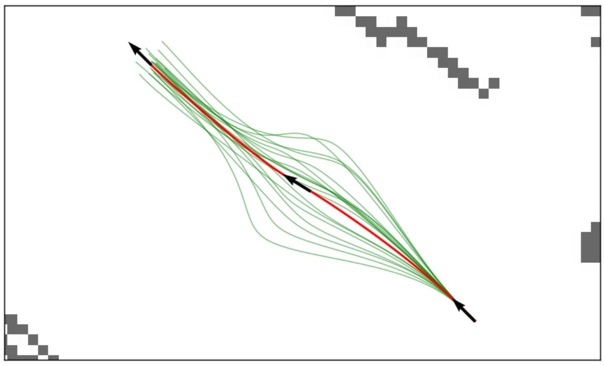

Figure 3.9: An initial population of path candidates. The red curve corresponds to ϕ_1.

population is therefore defined as the set

$$\Pi = \{\phi_1, ..., \phi_N\}.$$

Let $\phi_1 = \hat{\phi}$ where $\hat{\phi}$ is given together with $\mathcal{W}$ as input to the algorithm as an initial guess and let the rest be generated by randomly sampling from a multivariate (8-dimensional) Gaussian distribution with mean $\hat{\phi}$ and covariance Σ_{init} such that:

$$\phi_i \sim \mathcal{N}(\hat{\phi}, \Sigma_{\text{init}}), \quad i \in \{2, ..., N\}$$

where

$$\Sigma_{\text{init}} = \begin{bmatrix} \Sigma_{z_1} & 0 & 0 & 0 \\ 0 & \Sigma_{z_2} & 0 & 0 \\ 0 & 0 & \sigma_\alpha & 0 \\ 0 & 0 & 0 & \sigma_\beta \end{bmatrix}, \quad \text{with } \Sigma_{z_j} = \begin{bmatrix} \sigma_{x_j} & 0 & 0 \\ 0 & \sigma_{y_j} & 0 \\ 0 & 0 & \sigma_{\theta_j} \end{bmatrix}. \tag{3.8}$$

where σ_X denotes the variance for each sampled variable. An example of an initial population is shown in Figure 3.9.

Evaluation

The performance of every path candidate in Π is evaluated with regards to the requirements described in the introduction. It is found that the fitness of candidate i can be computed as:

$$F_i = \frac{f_0}{1 + c_L s_L + c_d s_d + c_\chi s_\chi + c_\kappa s_\kappa}$$

where c_L, c_d, c_χ, c_κ are weight parameters, f_0 a maximum fitness value and s_L, s_d, s_χ and s_κ are evaluated for each candidate as:

- s_L is a cost on path length:
$$s_L = (L - L^*)^2$$
where L is the length of the candidate and $L^* = \|r_{\text{goal}} - r_{\text{start}}\|$ is the length of the shortest possible (straight) path connecting the start and goal positions.

- s_d is a cost on terminal offset:
$$s_d = \|r_{\text{end}} - r_{\text{goal}}\|$$
where r_{end} is the termination point of the candidate path.

- s_χ is a cost on collisions:
$$s_\chi = \sum_j \|r_{j+1} - r_j\| \cdot \mathbf{1}(\rho(r_j) \leq \rho_{\text{safe}})$$
which corresponds to the length of the path segment that is closer to an obstacle than a minimum distance ρ_{safe}. The sum is taken over the sampled path points and $\mathbf{1}(\cdot)$

denotes an indicator function that is 0 if the argument is false, and 1 otherwise. $\rho(r)$ is a mapping between a position and the distance to the closest obstacle .

- s_κ is a cost on infeasible curvatures:

$$s_\kappa = \begin{cases} \max_j |\kappa_j| & \text{If} \max_j |\kappa_j| \geq \kappa_{\max} \\ 0 & \text{Otherwise} \end{cases}$$

  which corresponds to the path's maximum curvature if it exceeds maximum admissible curvature $\kappa_{\max}$.

The weights can be chosen to prioritize between different properties. Especially they should be chosen so that $c_\chi, c_\kappa \gg c_L, c_d$ since the associated quantities indicate that a path is infeasible and are zero otherwise. Furthermore, if either s_χ or s_κ are nonzero, a flag is raised to indicate that the path is infeasible to avoid giving it as output even if it happens to have highest fitness.

Selection

In the selection step, an intermediate population $\Pi' = \{\phi'_1, ..., \phi'_N\}$ with N candidates is generated by randomly sampling from Π with replacement. The probability p_i that candidate i is selected is determined from its fitness value according to

$$p_i = \frac{G_i}{\sum_j G_j}, \text{ where } G_i = \left(\frac{F_i}{\frac{1}{N} \sum_j F_j} \right)^4 .$$

This exaggerates differences in fitness values to further promote solutions with higher fitness.

Mutation

There is a probability p_{mut} for each candidate in Π' to mutate, which corresponds to random resampling of ϕ'_i using Gaussian distributions similar to the resampling in the initialization. If a candidate mutates, it will be replaced by a random sample according to:

$$\phi'_i \leftarrow \phi''_i \sim \mathcal{N}(\phi'_i, \Sigma_{\text{mut}}), \quad i \in \{1, ..., N\}$$

where Σ_{mut} is a diagonal covariance matrix with elements defined like Σ_{init} in (3.8). The variances can furthermore scale with path length and fitness so that the mutation is reasonably large for the given path and to promote exploration when fitness is low. Each element is therefore determined by the following function:

$$\sigma_X = \frac{(\lambda_X L^*)^2}{1 + (k_X F_i)^2}$$

such that the standard deviation of the Gaussian distribution is $\lambda_X L^*$ for variable X when $F_i = 0$. The parameter λ_X is a fraction of the shortest possible path distance L^*. Furthermore, k_X can be used to tune the sensitivity to F_i.

Termination Condition

The above steps are repeated until a termination condition is satisfied. Here, termination occurs either when a maximum number of iterations m is reached or when the population has converged to a feasible solution. Convergence is detected by tracking the best fitness value $F^*(k) \equiv \max_i F_i(k)$, where k is the iteration number. Termination occurs when $|F^*(k) - F^*(k-1)| < \delta_F$ for n consecutive iterations.

The output is the best feasible path. If none is found, the planner is said to have failed and a new attempt can be made using updated initial conditions.

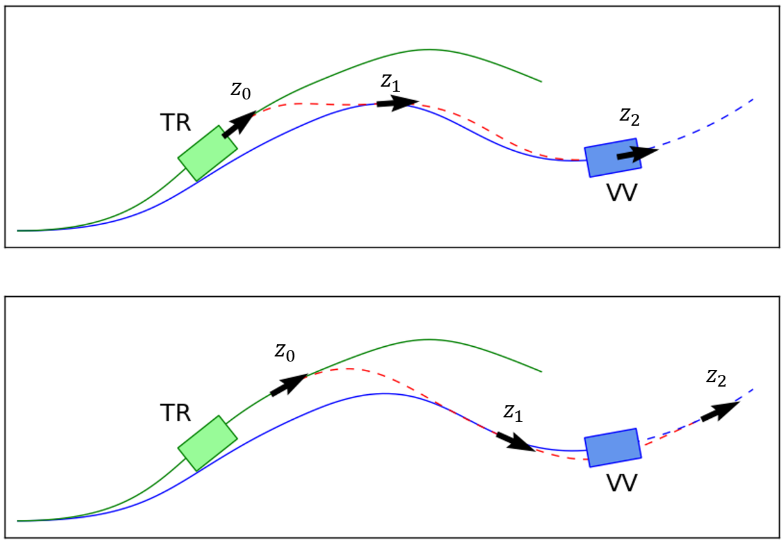

Figure 3.10: Two possibilities for choosing (z_0, z_1, z_2). The green curve is a previously planned path for TR. The blue solid curve is the path taken by VV. The dashed blue curve is a predicted path for VV. The red dashed curve is the path resulting from the choice of waypoints.

Initial Conditions $(\mathcal{W}, \hat{\phi})$

The algorithm takes as input the start point conditions $\mathcal{W} = (z_0, w', w'')$ and an initial guess $\hat{\phi} = (z_1, z_2, \alpha, \beta)$. The poses z_0, z_1 and z_2 are chosen as illustrated in Figure 3.10 according to the following:.

- z_0 lies on the previously planned path (green) a distance d_0 ahead of TR.
- z_1 lies on the path taken by VV (solid blue) a distance d_1 behind z_2
- z_2 lies on a predicted path (dashed blue) a distance d_2 ahead of VV.

First, d_0 is chosen to taken account for the time it takes for the path planner to find a path, so that $d_0 = \tau_0 v_T$. Where τ_0 is larger that the worst case planner time and v_T is the velocity of TR.

Second, d_2 is chosen to allow the planner to take road conditions further ahead into consideration when planning the path. It can be chosen in terms of VV's velocity as $d_2 = \tau_2 V_V$. Path prediction is performed assuming that the endpoint curvature κ remains constant. A point a distance d ahead is then given by:

$$\begin{aligned} x_2 &= x_V + \frac{\sin(\theta_V + d\kappa) - \sin\theta_V}{\kappa} \\ y_2 &= y_V - \frac{\cos(\theta_V + d\kappa) - \cos\theta_V}{\kappa} \\ \theta_2 &= \theta_v + d\kappa \end{aligned}$$

Finally, d_1 can be chosen so that z_1 is between z_0 and z_2. Let z_0' be the point on VV's path closest to z_0, and let d_{02} be the distance along the path between z_0 and z_2. Then, $d_1 = k_d d_{02}$, where $k_d \in [0, 1]$.

The initial Bézier offset parameters α and β are chosen as fractions of L^* such that: $\alpha = \alpha_0 L^*$ and $\beta = \beta_0 L^*$.

Furthermore, when a new path becomes available it will join with the previous at z_0 and the previously planned path beyond that point is disregarded.

Summary of Parameters

The parameters governing the path planning process are summarized in Table 3.3.

Symbol	Description
$\mathcal{W} = (z_0, w', w'')$	Fixed starting point conditions for path planning
$\hat{\phi} = (\hat{z}_1, \hat{z}_2, \alpha_0, \beta_0)$	Initial values for path planning variables
$\tau_0,\ \tau_2$	Time constants for determining $z_0,\ z_2$
k_d	Distance parameter for determining z_1
N	Number of population candidates
$\kappa_{\max}$	Highest admissible curvature
Σ_{init}*	Covariance matrix for initial population sampling
$c_L, c_d, c_\chi, c_\kappa$	Weights for evaluating fitness
f_0	Maximum fitness
ρ_{safe}	Minimum admissible obstacle distance
p	Probability for a candidate to mutate
λ_X, k_X	Tuning parameters for mutation variance
m	Maximum number of iterations
n	Maximum number of iterations with converged population
δ_F	Threshold for convergence detection

* Elements defined in equation (3.8)

Table 3.3: Path planning parameters

Chapter 4

Results

The system described in the previous chapter has been implemented and tested in a simulation of the ADS research platform Small-Vehicles-For-Autonomy (SVEA), which is shown in Figure 4.1. First, results indicative of each subsystem's performance are presented. This is followed by an evaluation of overall teleoperation performance conducted with user trials.

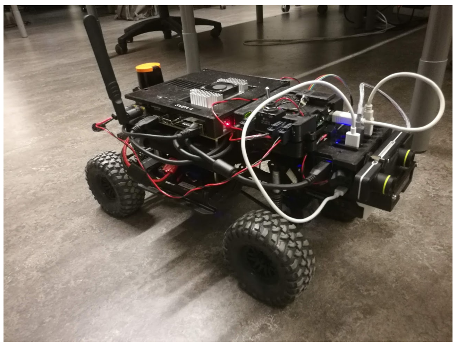

Figure 4.1: The SVEA vehicle that is simulated

4.1 Longitudinal Control

The longitudinal control performance is demonstrated by having an operator control VV to perform a step-like acceleration. TR follows along the path of VV (no planning) and neither vehicle applies any steering. Figure 4.2 shows the outcome when TR and VV have the same longitudinal dynamics and the same maximum velocities (1.0 m/s, at maximum throttle $\bar{u} = 1.0$). Figure 4.3 shows the outcome when TR is simulated to be heavier and have a maximum velocity limited to 80% of that of VV, while the operator requests maximum velocity ($v_r = 1.0$ m/s). This illustrates the performance of the $v_{\max}$ estimation to avoid that the vehicles diverge. The same parameters are applied in the two cases and are given in Table 4.1.

System	t_h	h	δ_v	δ_u	K	b	T_i
TR	1.0 s	0.1 s	-	0.5	1.0	1.0	0.2
VV	-	0.033 s	0.8	-	0.1	0.5	0.05

Table 4.1: Longitudinal control parameters

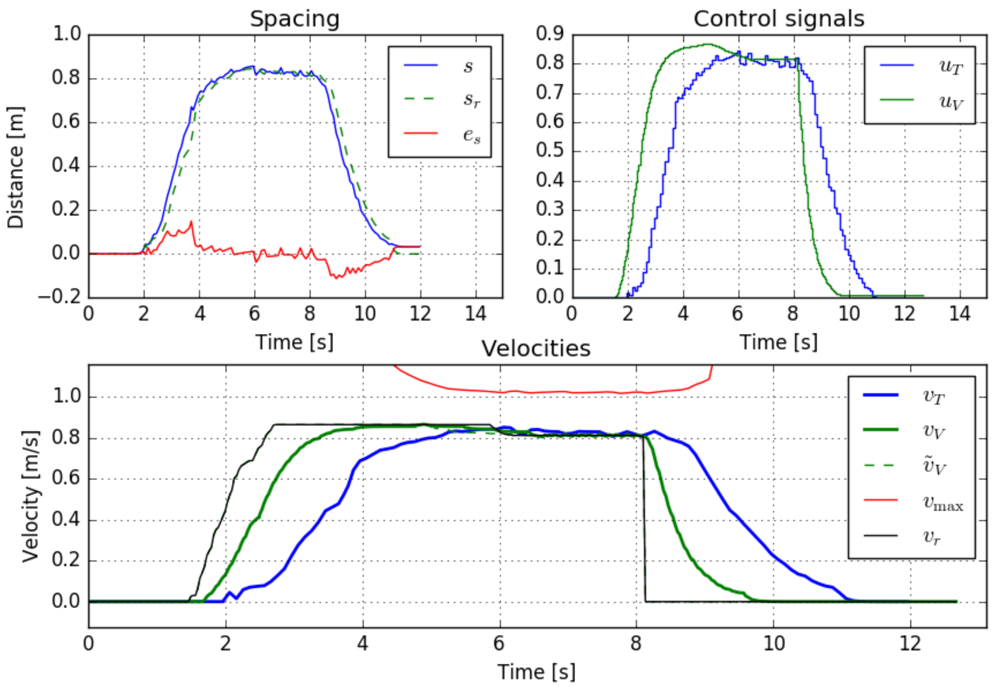

Figure 4.2: Results when VV is commanded by the operator to perform a step-like maneuver. VV and TR have identical dynamics and motion constraints. The measured and reference spacings are denoted s and s_r respectively and $e_s = s - s_r$ is the spacing error. The throttle commands for each vehicle are denoted u_T and u_V, while v_T, v_V are the vehicles' velocities, $\tilde{v}_v$ is the saturated reference velocity for VV, $v_{\max}$ is the estimated maximum velocity of TR and v_r is the operator input.

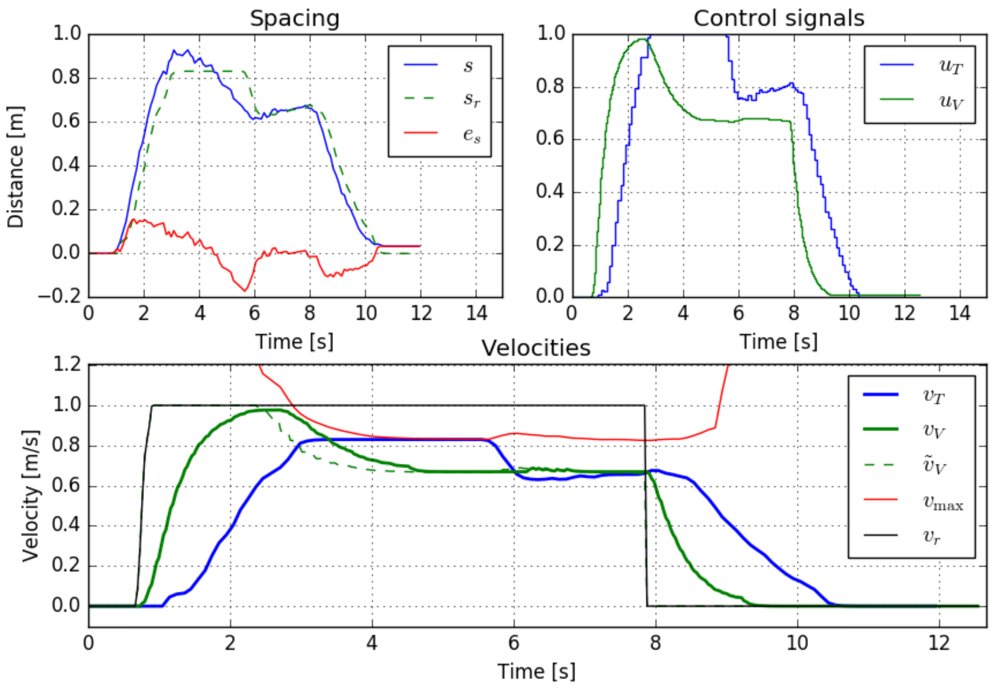

Figure 4.3: Results when VV is commanded by the operator to perform a step-like maneuver and TR is simulated to have a maximum velocity of 80% of that of VV. The measured and reference spacings are denoted s and s_r respectively and $e_s = s - s_r$ is the spacing error. The throttle commands for each vehicle are denoted u_T and u_V, while v_T, v_V are the vehicles' velocities, $\tilde{v}_v$ is the saturated reference velocity for VV, $v_{\max}$ is the estimated maximum velocity of TR and v_r is the operator input.

4.2 Lateral Control

The performance of the lateral controller is demonstrated Figure 4.4. The left part shows the paths taken by VV, which is maneuvered manually with aggressive steering, and TR, which applies both longitudinal and lateral control to track VV's path. The lateral control errors are shown in the top right part and the executed steering angles at the bottom right. Parameters are given in Table 4.2, where the model parameters are based on the rough estimations performed in [57] for a similarly sized vehicle. The optimization problem was solved around 80 times and the solver performance is shown in Table 4.3 in terms of computation time. The sampling period h and horizons n_p and n_c were chosen with regards to the solve time.

h	n_p	n_c	q	r	$\delta_{\max}$	C_α	l_f	l_r	m	I_z
0.15 s	2	2	0.05	0.02	35°	$5.0\ \frac{\text{N}}{\text{rad}}$	0.16 m	0.16 m	6.0 kg	$0.07\ \text{kg}\cdot\text{m}^2$

Table 4.2: Lateral control parameters

Mean solve time	0.077 s
Standard deviation	0.033 s

Table 4.3: Optimization performance

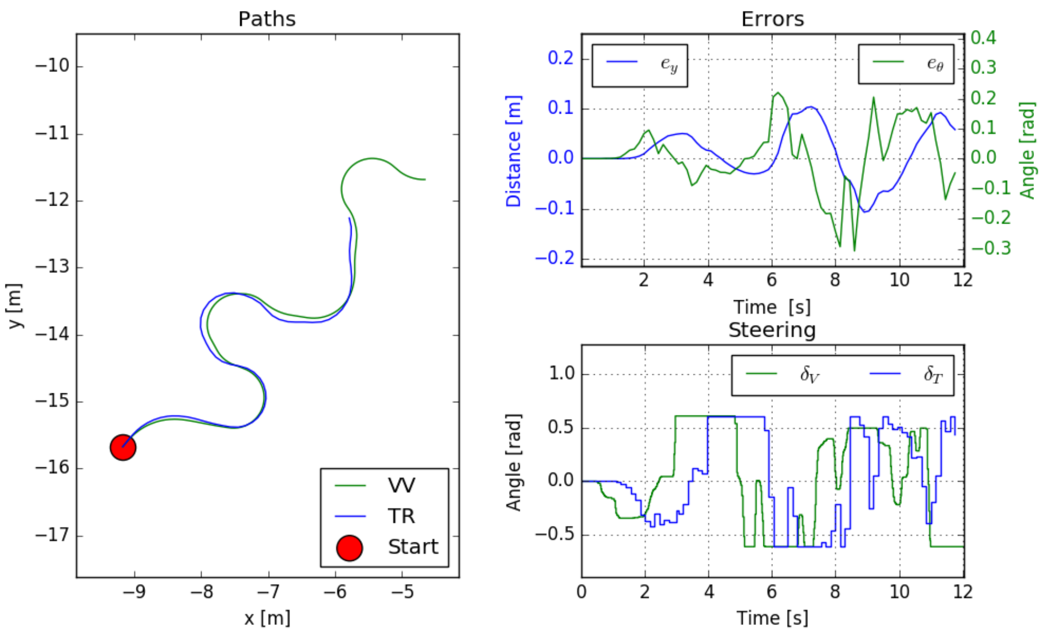

Figure 4.4: Lateral controller performance during aggressive maneuvering

4.3 Path Planning

Figures 4.5 and 4.6 display the outcome in two cases when the path planner makes 100 attempts to generate feasible paths in different conditions. A binary gridmap is used to represent the location of obstacles and arrows represent the initial poses z_0, z_1 and z_2. The algorithm parameters used in each case are given in Table 4.4. The differences are highlighted in bold, with case 2 having larger weight on path length and smaller mutation variances. Quantitative results about the performance of the planner are shown in Table 4.5. The cases that are demonstrated represent the following scenarios:

- Case A: A trivial planning task. No obstacles and no complex maneuvers.
- Case B: The case where VV has cut a corner in the virtual environment and TR must adapt the route to avoid a collision.
- Case C: A situation with a narrow set of feasible solutions due to the constraints on curvature and distance to obstacles.
- Case D: A case where VV is inside or behind an obstacle and TR must plan a path to accomplish a route that is still similar to the one taken by VV.

Parameter	Case 1 (fig. 4.5)	Case 2 (fig. 4.6)	Unit
(α_0, β_0)	(0.2, 0.2)	(0.2, 0.2)	-
N	20	20	-
$\kappa_{\max}$	2.5	2.5	1/m
$(\sqrt{\sigma_{x_1}}, \sqrt{\sigma_{y_1}}, \sqrt{\sigma_{\theta_1}})$	$(0.25, 0.25, 0.05)$	$(\mathbf{0.1}, \mathbf{0.1}, \mathbf{0.01})$	m
$(\sqrt{\sigma_{x_2}}, \sqrt{\sigma_{y_2}}, \sqrt{\sigma_{\theta_2}})$	$(0.1, 0.1, 0.05)$	$(\mathbf{0.1}, \mathbf{0.1}, \mathbf{0.01})$	m
$(\sqrt{\sigma_\alpha}, \sqrt{\sigma_\beta})$	$(0.01, 0.01)$	$(0.01, 0.01)$	m
$(c_L, c_d, c_\chi, c_\kappa)$	(100, 500, 1000, 100)	(**500**, 500, 1000, 100)	-
f_0	100	100	-
ρ_{safe}	0.2	0.2	m
p	0.4	0.4	-
(λ_{z_1}, k_{z_1})	$(0.1, 3.0)$	$(\mathbf{0.05}, \mathbf{2.0})$	-
(λ_{z_2}, k_{z_2})	$(0.1, 3.0)$	$(\mathbf{0.05}, \mathbf{2.0})$	-
$(\lambda_{\alpha\beta}, k_{\alpha\beta})$	$(0.05, 3.0)$	$(\mathbf{0.05}, \mathbf{2.0})$	-
m	20	20	-
n	3	**6**	-
δ_F	20	20	-

Table 4.4: Path planning parameters and units where applicable

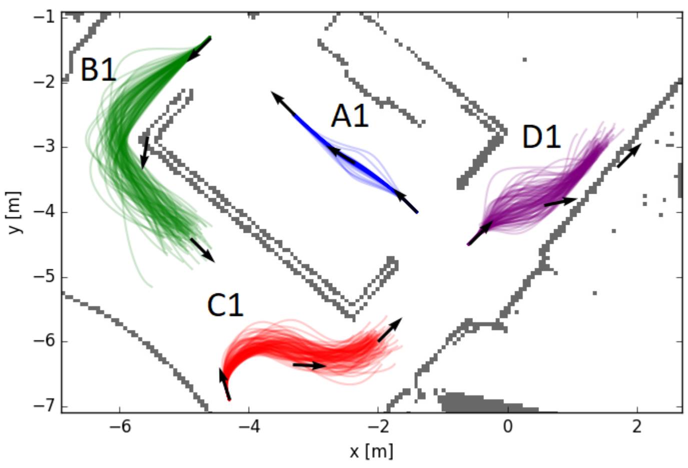

Figure 4.5: Case 1: The output paths with 100 attempts for each case A-D. Arrows indicate initial input poses.

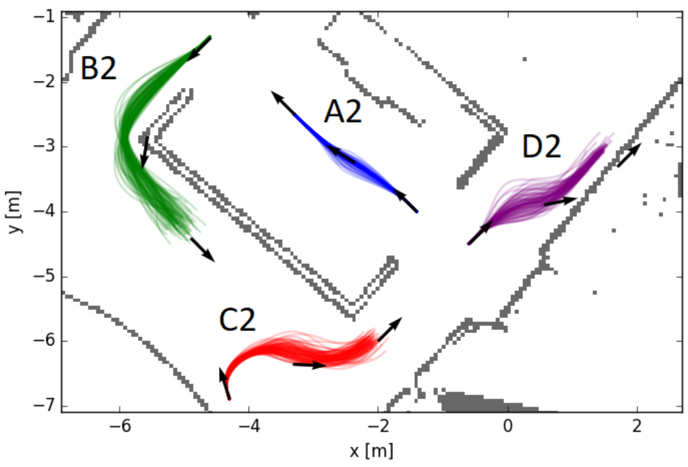

Figure 4.6: Case 2 : The output paths with 100 attempts for each case A-D. Arrows indicate initial input poses.

Parameter	A1	B1	C1	D1	A2	B2	C2	D2
Fails	0	2	0	6	0	0	4	18
Successes	100	98	100	94	100	100	96	82
Max iteration*	9	9	15	15	13	20	20	20
Min iteration*	3	4	4	4	6	7	7	6
Mean iterations*	3.1	5.1	6.4	7.1	6.1	10.0	11.6	12.3
Std. iterations*	0.6	1.3	2.5	2.7	0.7	2.6	3.3	3.3
Mean time [ms]	76	95	109	128	111	163	190	211
Std. time [ms]	20	32	33	52	21	35	48	54

* Excluding failed attempts

Table 4.5: Quantitative results of the different cases shown in Figures 4.5 and 4.6. The fields show the number of times path planning failed or was successful, the highest and lowest number of iterations required in a successful case, the mean number and standard deviation of iterations needed in a successful case and mean computation time and associated standard deviation.

4.4 Complete System

Figures 4.7 and 4.8 illustrate the overall teleoperation performance when motion control and path planning is combined. In the example, an operator maneuvers VV in a static environment represented by a binary gridmap. The operator occasionally makes maneuvering mistakes such that VV goes inside obstacles. Two such segments are highlighted in the smaller plots to the right in Figure 4.7. Figure 4.8 displays velocity and steering data over the course of the demonstration. Parameters are similar to those given in Tables 4.1, 4.2 and case 1 of Table 4.4. A video of a similar demonstration is available at `https://youtu.be/ZYqT2FalNhs`.

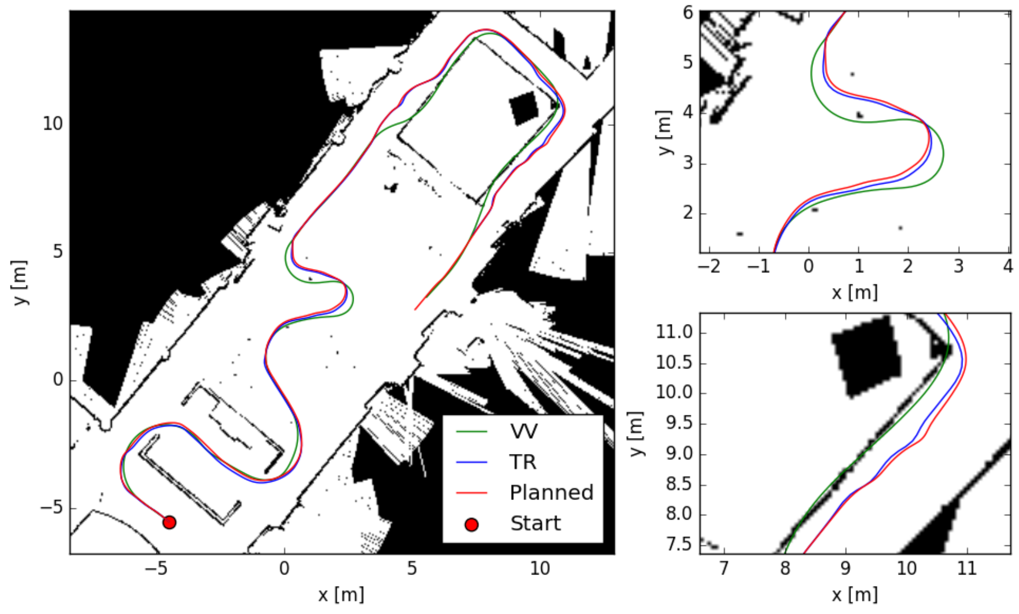

Figure 4.7: Tracking results for the complete system.

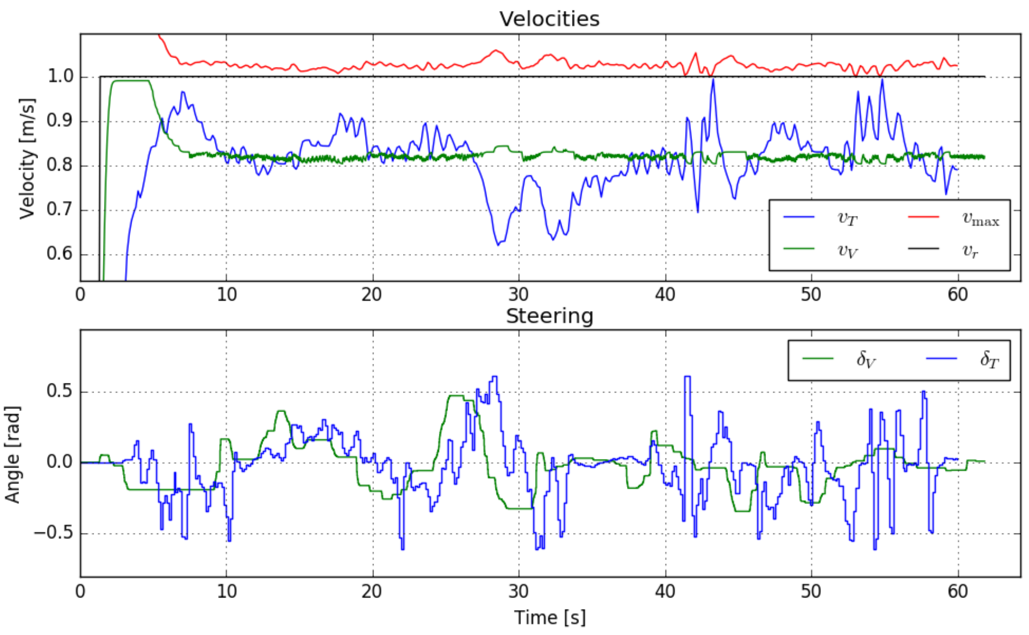

Figure 4.8: Velocity and steering values for the case depicted in Figure 4.7

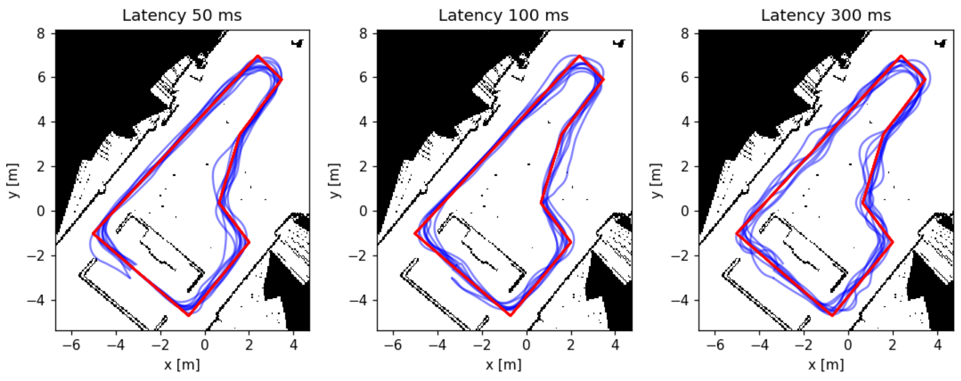

Figure 4.9: Paths produced with direct control and simulated delays as indicated. The path to follow is marked in red. Movement is counterclockwise.

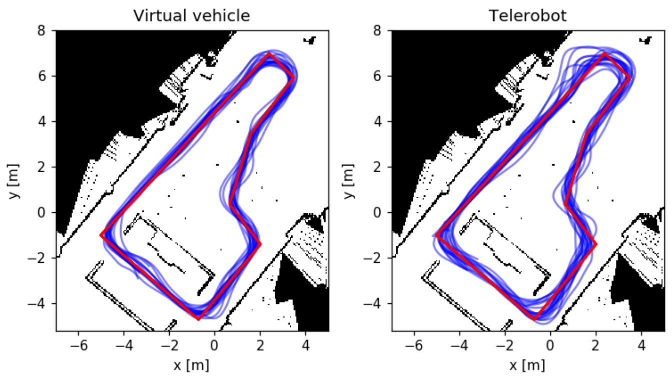

Figure 4.10: Paths produced by VV and TR respectively with shared control using the virtual environment interface. The path to follow is marked in red. Movement is counterclockwise.

4.5 User Trials

To assess the usability of the complete system, user trials were conducted. Ten participants were to teleoperate a vehicle counterclockwise around a predefined loop, first by direct control followed by shared control with the virtual environment interface. The participants were not familiar with the system or the control interface beforehand, but were given a few minutes to try the controls before initiating the trials.

In the direct control case, simulated delays were added to the control input to simulate the effect of network latency. On the first lap this would be 50 ms to approximately correspond to the expected latency of a 5G network. With every new lap, delay was randomly chosen as either 100 ms (representing 4G) or 300 ms (representing a worst case scenario). The vehicles were controlled with a computer mouse interface with a fixed third person view similar to the view shown in Figure 3.4. When using shared control, both VV and TR were presented to the operator in the virtual environment.

The paths that were produced in the different cases are shown in Figures 4.9 and 4.10 and the spread of lap completion times is shown in Figure 4.11. The boxes extend from the lower to upper quartile values, while the whiskers indicate the full range and the orange line shows the median.

After performing the experiment, the participants answered a survey about their subjective impressions of the systems and the questions and outcome is summarized in Table 4.6. The questions were partly derived from the System Usability Scale (SUS). In addition to the survey questions, a few participants remarked that the control interface felt unfamiliar. One person said that it made the hand movements overly tense. Two commented that very little attention was given to the TR in the experiment, which is likely to affect the response to question 7 in the survey.

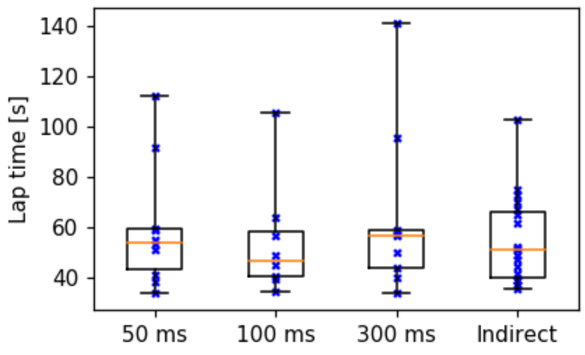

Figure 4.11: The spread of lap completion times under the influence of delay (50, 100 or 300 ms) or with shared (indirect) control.

Questions	**Responses**				
Direct control	**1**	**2**	**3**	**4**	**5**
1. The system was easy to use	0	0	1	8	1
2. I felt confident using the system	0	1	5	4	0
3. I found the system cumbersome to use	0	8	1	0	1
Indirect control	**1**	**2**	**3**	**4**	**5**
4. The system was easy to use	0	0	0	8	2
5. I felt confident using the system	0	1	1	6	2
6. I found the system cumbersome to use	1	8	0	1	0
7. TR behaved like I intended	0	1	3	4	2

Preferences	**Direct**	**Indirect**	**No preference**
8. Which control method did you prefer?	3	7	0

Table 4.6: Survey questions and responses from user experiments. Respondents gave the degree of agreeing with each statement on a scale from 1 to 5 corresponding to strongly disagreeing to strongly agreeing. The numbers to the right of the questions show how many respondents chose each option.

Chapter 5

Discussion

5.1 Motion Control

The longitudinal step responses in Figures 4.2 and 4.3 show that the spacing error is kept below 20 cm during transients and is close to zero when VV has constant velocity. It can also be seen that the active estimation of TR's maximum velocity converges to the true values as intended when they are approached, while being conservative otherwise. It also achieves the goal of ensuring that the vehicles does not diverge as seen in the second case (Figure 4.3). The operator input is constant $v_r = 1$ m/s and VV initially reaches $\approx$1 m/s, but decelerates automatically as $v_{\max}$ converges to $\approx$0.8 m/s. The undershoot of $\approx$18 cm in e_s around time 5 s is likely due to brief integrator windup.

The MPC for lateral control also have acceptable performance, as shown in Figure 4.4, by not deviating more than 10 cm and 0.3 rad $\approx 17°$ from a desired path despite aggressive curvature. It can probably be improved by having longer prediction and control horizons in the MPC. Here, they are both restricted to small values due to poor real-time performance of CVXPY in solving the optimization problem.

The demonstrations are performed at relatively low speeds, which could represent driving on a parking lot or near a road construction work. However, further analysis is needed to asses the performance at higher speeds. Further insights could also be obtained by implementing the controllers on a real system with a cellular network as communication link. A possible extension is to consider vehicle dynamics and path curvature in the longitudinal controller to, for example, adjust the speed when making sharp turns. Other extensions are to allow reversing maneuvers and to use brake commands to control velocity.

5.2 Path Planning

Figures 4.5 and 4.6 along with Table 4.5 illustrates the capabilities of the proposed genetic algorithm for path planning. It is shown that it is able to produce paths that are feasible even when the initial guess is not, such as the cases B and D, which correspond to VV having travelled straight through obstacles. In the simplest case (A) the produced paths have small variations, as desired for TR to follow VV as similarly as possible. The generated paths in the other cases have higher variability due to the inherent randomness of the algorithm. It is seen with the second set of parameters that variability can be reduced by putting larger weight on path length, while decreasing the mutation variances and increasing the required number of consecutive converged solutions for termination. However, this reduces the success rates and increases the computation times. In case D, success rate drops from 94% to 82%. Increasing the maximum number of iterations would increase the success rate, which would however further increase the computation time.

The results indicate that computation times in simple cases (like A) lie in the order of magnitude of 100 ms, while it reaches 200 ms in more challenging cases. It is therefore deemed a viable method for path-planning in real-time, assuming that VV is operated to produce mostly feasible paths.

The results also show that especially cases B, C and D have relatively large variances in the termination point of the generated paths, such that if followed accurately, TR would not end up at the same location at VV. It can be improved by increasing the weight on termination offset as well as reducing the endpoint mutation variance. However, it is only a minor issue, as the computation times allow the vehicle to replan the route about 5 to 12 times per second with the mean endpoint being close to the desired one.

It is likely that variations in parameters, algorithm and path representation could improve path planning performance. A suggestion for future work is to conduct a thorough examination on the rich literature that exist on genetic algorithms to potentially improve convergence. Nevertheless, the obtained results illustrate the potential of a genetic algorithm in this particular application.

5.3 Complete System

A demonstration of the performance when path planning and motion control is combined is shown in Figures 4.7 and 4.8. It is seen that, most of the time, the path of TR coincides approximately with the path taken by VV, but it deviates when VV makes sharp turns or is too close to or inside obstacles. Two noteworthy examples are highlighted in the right part of Figure 4.7. In the top right, TR is seen to successfully avoid colliding with two small narrowly spaced obstacles that VV approached too closely. In the bottom right, VV travels inside a wall for a long stretch and TR successfully chooses a safe path parallel to it. The curvature is however fluctuating, which is in line with the high variability of case D1 in Figure 4.5. When VV eventually emerges, the paths again coincide.

Figure 4.8 shows how the velocity is adjusted to account for the vehicles occasionally pursuing path segments of different lengths. At for example $t = 28\,\mathrm{s}$, TR slows down as it takes a shorter route than VV, whereas at around $t = 55\,\mathrm{s}$, speed is increased to compensate for the longer path it takes when VV travels inside the wall.

Figure 4.10 shows the consistency of the system during the user trials. VV is maneuvered by test participants along a given path free from obstacles. TR manages to follow VV without failure in every case but deviates significantly several times at the U-turn in the top-right. The reason for this is not clear, but could be due to the fact that many participants slowed down significantly at the U-turn such that TR approached VV, which would give less headway for the path planner. Furthermore, TR is seen to perform 90 degree turns more variably than VV. It tends to prefer the shorter routes achieved by cutting the corners, which is expected as the genetic algorithm promotes shorter paths.

In a real world implementation, implications of a dynamic environment with moving obstacles must be considered. One particular issue arises with the separation between VV and TR. An example is passing a crosswalk that is unoccupied for VV but is used by pedestrians when TR reaches the same point. This motivates having a small vehicle separation, which, on the other, hand provides less headway for the path planner to anticipate the "future" intentions of the operator and execute motion accordingly. With higher levels of autonomy however, the severity of this issue decreases. TR could for example be equipped with an active predictive safety system as proposed in [43] to automatically stop for the pedestrians and alert the operator of the override. VV could then be teleported back to the location of TR or paused until TR catches up. This requires an efficient design for human-machine interaction and would be an interesting topic for future work.

5.4 User Trials

Relating to the overall scope of the thesis, making the task of teleoperation easier for the operator, user trials were conducted to evaluate the proposed system's usability when compared to direct control. Figure 4.9 illustrates clearly how driving performance degrades with increased time-delay when using direct control. Latencies 50 ms and 100 ms, which correspond to typical latencies using 5G and 4G respectively, seem to have negligible impact on driving performance. However, in agreement with previous research, a latency of 300 ms greatly increases the difficulty to maneuver as indicated by the greater oscillations of the taken paths.

Figure 4.10 shows the paths of VV and TR when using shared control with the virtual environment interface. Comparing VV's and TR's paths, it seems VV is more accurate in adhering to the predefined path than TR, which deviates in particular near segments of high curvature as an effect of the genetic algorithm. It does however not contain the oscillatory behaviour observed in the case of 300 ms latency with direct control.

The spread of lap completion times as shown in Figure 4.11. The slowest lap times occurred with the 300 ms latency, further consolidating the notion of it being the most difficult case. The fastest times were recorded in the case of 100 ms latency, which can be due to this delay occurring in the participants' second or third lap, rather than the first, which invariably had latency of 50 ms. The lap times in the shared control case seem to cluster into two distinct groups. This could be an indication of two different control strategies where participants either supervised TR carefully while maneuvering VV or did not give TR any attention at all.

Turning to the survey results given in Table 4.6, the participants found the systems similarly cumbersome and easy to use, but reported higher confidence when using indirect control. Noteworthy is that not all participants experienced the worst case of 300 ms latency as it was chosen randomly, which likely affected the reported confidence with direct control.

For shared control, a majority of participants reported that TR behaved like intended. Several however remarked that they had not given TR any attention during the experiment, which may explain the three neutral responses in question 7. One participant remarked explicitly that TR behaved in unexpected ways, which could correspond to the odd behaviours observed at the U-turn in Figure 4.10. 70% of the participants reported an overall preference for the shared control method.

These results indicate, despite a small sample size, that the method proposed in this work provides benefits for usability in the presence of large delays, but performs on par with or slightly worse than direct control when the delay is small. The main issue seems to be the path planner, which occasionally creates paths deviating significantly from VV's path even when there are no feasibility conflicts. It is likely that a further investigation into the path planning problem in this application can improve the performance.

Also, the operator interface has a big impact on usability. In the assessment presented here, participants used a simple computer mouse interface for providing controls, and the graphical user interface relied on the visualization tool RVIZ. The view was fixed as a third-person perspective behind the vehicle and obstacles where presented on a two dimensional gridmap on top of which the three dimensional vehicles operated as shown in Figure 3.4. For a real system with a dynamic environment, it would also be necessary to recreate the actual scene in the virtual environment based on sensor data. Potentially, this would not require a high-resolution real-time video stream from the telerobot, which could reduce the amount of data that must be communicated. An in-depth investigation on how to design an efficient operator interface would be an interesting topic for further work.

On a final note, having a virtual environment as suggested in this thesis, opens up many possibilities for operator assistance. VV could for example employ obstacle avoidance or lane keeping functions to make it impossible to collide or drive off-road in the virtual environment. Providing VV with greater autonomous capabilities could also allow for more of a supervisory approach with the operator providing waypoints or approximate hand-drawn paths that could be dynamically edited as a basis for decisions.

Chapter 6

Conclusion

To conclude, this thesis explores a concept for vehicle teleoperation, grounded in the idea of having a virtual environment as an intermediate interface between the operator and the telerobot. The main benefit is that the effect of network latency on the operator's driving performance is eliminated. It also enables potentially safer teleoperation as the telerobot acts fully autonomously to mimic the motion of the virtual vehicle while avoiding obstacles. The proposed system could be enhanced by functions for operator assistance directly in the virtual environment to further simplify the driving task and promote safety.

The thesis focuses on the control aspect of the concept and aims at establishing a method for the telerobot to execute commands to achieve a motion similar to that of the virtual vehicle, while ensuring that the vehicles do not diverge and reducing the impact of operator errors. This is divided into three subproblems; longitudinal control, lateral control and path planning. For longitudinal control, a concept with inspiration from adaptive cruise control is proposed to have the telerobot maintain a constant time-headway to the virtual vehicle using a PI controller. To prevent the vehicles from diverging, the maximum velocity of the virtual vehicle is limited with regards to the maximum velocity of the telerobot, which is estimated continuously in real-time. Lateral control is accomplished with a model predictive controller to stabilize the telerobot to the planned path. Path planning uses a genetic algorithm to take advantage of having access to a, in most cases, feasible initial guess based on the path of the virtual vehicle.

The suggested combination of longitudinal and lateral control is demonstrated in simulation to achieve good tracking performance at low speed. This can however probably be improved further by using an MPC solver with better real-time performance than is used in this work.

The suggested path planning algorithm succeeds in producing feasible paths in real-time, even in cases where the virtual vehicle passes through obstacles. It however introduces a relatively high degree of path variability, which can cause the planned paths to be suboptimal. It is seen that the variability is affected by parameter choices and can be reduced at the cost of success rate and computation time. The implemented algorithm does however not take full advantage of the rich literature in genetic algorithms and it is possible it can be improved with other choices of initial population, fitness function and selection and mutation methods, as well as including recombination or employing a different path representation. An in-depth analysis of path planning methods within the context of the proposed concept is suggested as a topic for future work.

Usability is evaluated through user trials, where 10 participants were asked to perform a teleoperation task, first by direct control with simulated communication latency and then by using the system proposed in the thesis. Afterwards, they were surveyed on their subjective experience. The trials show that the virtual interface system is superior to direct control in the presence of large delays, while being slightly worse when the delay is small, mostly due to characteristics of the path planning algorithm. The participants however reported having higher confidence when using the virtual interface system and a majority reported an overall preference the same.

All in all, this work outlines and evaluates a novel teleoperation concept using a virtual environment interface. It can be further expanded by evaluating the teleoperation system on a real vehicle, considering dynamic environments, allowing reversing maneuvers, adding operator assistance in the virtual environment, studying path planning more in-depth as well as studying possible ways to design the operator interface for efficient communication between human and telerobot.

Driving is an immensely complex task to automate, and teleoperation is a necessary means to reap the benefits of partially autonomous vehicles until full automation is realized. It however brings its own set of challenges, which can compromise safety if not accounted for. This thesis has presented one way of approaching the teleoperation problem to relieve it of some of its known deficits.

Bibliography

[1] The SAE On-Road Automated Vehicle Standards Committee, "Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles," tech. rep., SAE International, 2018.

[2] A. Davies, "Waymo has taken the human out of its self-driving cars," *Wired*, Nov 2017. Accessed May 16, 2020 at https://www.wired.com/story/waymo-google-arizona-phoenix-driverless-self-driving-cars/.

[3] Wired, "Inside uber's self-driving car." [YouTube video], Sep. 14 2016. Accessed May 16, 2020 at https://www.youtube.com/watch?v=18uWvm6W4bc.

[4] Scania Group, "Introducing scania axl, a cableless autonomous concept truck." [YouTube video], Sep. 24 2019. Accessed May 17, 2020 at https://www.youtube.com/watch?v=8bX48KVeN2U.

[5] Volvo Buses, "Live demonstration of an autonomous bus in depot – volvo buses." [YouTube video], Nov. 25 2019. Accessed May 17, 2020 at https://www.youtube.com/watch?v=1crg11NwqDE.

[6] Einride, "Einride pod autonomous mode." [YouTube video], May. 11 2018. Accessed May 17, 2020 at https://www.youtube.com/watch?v=xr7sC9zPWDc.

[7] C. Badue, R. Guidolini, R. V. Carneiro, P. Azevedo, V. B. Cardoso, A. Forechi, L. Jesus, R. Berriel, T. Paixão, F. Mutz, L. Veronese, T. Oliveira-Santos, and A. F. De Souza, "Self-driving cars: A survey," 2019.

[8] C.-Y. Chan, "Advancements, prospects, and impacts of automated driving systems," *International Journal of Transportation Science and Technology*, vol. 6, no. 3, pp. 208–216, 2017.

[9] M. M. Morando, Q. Tian, L. T. Truong, and H. L. Vu, "Studying the safety impact of autonomous vehicles using simulation-based surrogate safety measures.(research article)," *Journal of Advanced Transportation*, vol. 2018, 2018.

[10] R. Hoogendoorn, B. van Arem, and S. Hoogendoorn, "Automated driving, traffic flow efficiency, and human factors," *Transportation Research Record*, vol. 2422, no. 2422, pp. 113–120, 2014.

[11] L. Truong, C. Gruyter, G. Currie, and A. Delbosc, "Estimating the trip generation impacts of autonomous vehicles on car travel in victoria, australia," *Transportation*, vol. 44, no. 6, pp. 1279–1292, 2017.

[12] D. Milakis, B. van Arem, and G. van Wee, "Policy and society related implications of automated driving: a review of literature and directions for future research," *Journal of Intelligent Transportation Systems: technology, planning, and operations*, pp. urn:issn:1547–2450, 2017.

[13] P. Sigal, "Autonomous vehicles undergo 'reality check'," *Automotive News Europe*, Feb 2020. Accessed May 17, 2020 at https://europe.autonews.com/automakers/autonomous-vehicles-undergo-reality-check.

[14] A. Davies and A. Marshall, "Are we there yet? a reality check on self-driving cars," *Wired*, Apr 2019. Accessed May 17, 2020 at https://www.wired.com/story/future-of-transportation-self-driving-cars-reality-check/.

[15] F. Jiang, "The automated vehicle traffic control tower," 2019.

[16] L. Kang, B. Qi, and S. Banerjee, "Augmenting self-driving with remote control: Challenges and directions," pp. 19–24, 02 2018.

[17] P. Sawers, "Einride demos a single teleoperator taking control of multiple autonomous trucks," *Venturebeat*, April 2020. Accessed May 18, 2020 at https://venturebeat.com/2020/04/07/einride-demos-a-single-teleoperator-taking-control-of-multiple-autonomous-trucks/.

[18] A. Davies, "Nissan's path to self-driving cars? humans in call centers," *Wired*, May 2017. Accessed May 18, 2020 at https://www.wired.com/2017/01/nissans-self-driving-teleoperation/.

[19] A. Davies, "Self-driving cars have a secret weapon: Remote control," *Wired*, Feb 2018. Accessed May 18, 2020 at https://www.wired.com/story/phantom-teleops/.

[20] A. Davies, "The war to remotely control self-driving cars heats up," *Wired*, Jan 2019. Accessed May 18, 2020 at https://www.wired.com/story/designated-driver-teleoperations-self-driving-cars/.

[21] B. Paden, M. Cap, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 33–55, 2016.

[22] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE Access*, vol. 8, pp. 58443–58469, 2020.

[23] P. Ross E., "The audi a8: the world's first production car to achieve level 3 autonomy," *IEEE Spectrum*, Jul 2017. Accessed May 19, 2020 at https://spectrum.ieee.org/cars-that-think/transportation/self-driving/the-audi-a8-the-worlds-first-production-car-to-achieve-level-3-autonomy.

[24] T. Fong and C. Thorpe, "Vehicle teleoperation interfaces," *Autonomous Robots*, vol. 11, no. 1, pp. 9–18, 2001.

[25] S. Gnatzig, F. Chucholowski, T. Tang, and M. Lienkamp, "A system design for teleoperated road vehicles," vol. 2, 07 2013.

[26] J. Chen, E. Haas, and M. Barnes, "Human performance issues and user interface design for teleoperated robots," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 37, no. 6, pp. 1231–1245, 2007.

[27] S. Neumeier, E. A. Walelgne, V. Bajpai, J. Ott, and C. Facchi, "Measuring the feasibility of teleoperated driving in mobile networks," in *2019 Network Traffic Measurement and Analysis Conference (TMA)*, pp. 113–120, IFIP, 2019.

[28] R. Inam, N. Schrammar, K. Wang, A. Karapantelakis, L. Mokrushin, A. Feljan, and E. Fersman, "Feasibility assessment to realise vehicle teleoperation using cellular networks," in *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, pp. 2254–2260, Institute of Electrical and Electronics Engineers Inc., 2016.

[29] M. Agiwal, A. Roy, and N. Saxena, "Next generation 5g wireless networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 1617–1655, 2016.

[30] T. Sheridan, "Space teleoperation through time delay: review and prognosis," *IEEE Transactions on Robotics and Automation*, vol. 9, no. 5, pp. 592–606, 1993.

[31] "Curiosity rover gets a boost from artificial intelligence.," *Nature*, vol. 546, no. 7660, pp. 578–578, 2017.

[32] T. Fong, J. Rochlis Zumbado, N. Currie, A. Mishkin, and D. L. Akin, "Space telerobotics: Unique challenges to human–robot collaboration in space," *Reviews of Human Factors and Ergonomics*, vol. 9, no. 1, pp. 6–56, 2013.

[33] S. Chien and K. L. Wagstaff, "Robotic space exploration agents," *Science Robotics*, vol. 2, no. 7, p. eaan4831, 2017.

[34] A. D. Dragan and S. S. Srinivasa, "A policy-blending formalism for shared control," *The International Journal of Robotics Research*, vol. 32, no. 7, pp. 790–805, 2013.

[35] S. Javdani, H. Admoni, S. Pellegrinelli, S. S. Srinivasa, and J. A. Bagnell, "Shared autonomy via hindsight optimization for teleoperation and teaming," *The International Journal of Robotics Research*, vol. 37, no. 7, pp. 717–742, 2018.

[36] Y. Gao, F. J. Jiang, X. Ren, L. Xie, and K. H. Johansson, "Reachability-based human-in-the-loop control with uncertain specifications," in *Proceedings of 21st IFAC World Congress*, 2020.

[37] C. Brooks and D. Szafir, "Balanced information gathering and goal-oriented actions in shared autonomy," vol. 2019-, pp. 85–94, IEEE Computer Society, 2019.

[38] D. Sadigh, S. Sastry, S. Seshia, and A. Dragan, "Information gathering actions over human internal state," vol. 2016-, pp. 66–73, Institute of Electrical and Electronics Engineers Inc., 2016.

[39] E. You and K. Hauser, "Assisted teleoperation strategies for aggressively controlling a robot arm with 2d input," vol. 7, pp. 354–361, MIT Press Journals, 2012.

[40] S. M. LaValle, "Rapidly-exploring random trees : a new tool for path planning," 1998.

[41] A. Hosseini, T. Wiedemann, and M. Lienkamp, "Interactive path planning for teleoperated road vehicles in urban environments," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pp. 400–405, IEEE, 2014.

[42] T. Suzuki, Y. Amano, T. Hashizume, and N. Kubo, "Vehicle teleoperation using 3d maps and gps time synchronization," *IEEE Computer Graphics and Applications*, vol. 33, no. 5, pp. 82–88, 2013.

[43] A. Hosseini and M. Lienkamp, "Predictive safety based on track-before-detect for teleoperated driving through communication time delay," in *2016 IEEE Intelligent Vehicles Symposium (IV)*, vol. 2016-, pp. 165–172, IEEE, 2016.

[44] T. Tang, P. Vetter, S. Finkl, K. Figel, and M. Lienkamp, "Teleoperated road vehicles – the "free corridor" as a safety strategy approach," *Applied Mechanics and Materials*, vol. 490-491, no. Mechanical Design and Power Engineering, pp. 1399–1409, 2014.

[45] H. Kam, S.-H. Lee, T. Park, and C.-H. Kim, "Rviz: a toolkit for real domain data visualization," *Telecommunication Systems*, vol. 60, no. 2, pp. 337–345, 2015.

[46] R. Rajamani, *Vehicle Dynamics and Control.* Mechanical Engineering Series, Boston, MA: Springer US, 2006.

[47] A. Eskandarian, *Handbook of Intelligent Vehicles.*, vol. 1-2. 2012.

[48] B. Wittenmark, K.-E. Årzén, and K. J. Åström, *Computer Control: An Overview.* 2002.

[49] E. Kim, J. Kim, and M. Sunwoo, "Model predictive control strategy for smooth path tracking of autonomous vehicles with steering actuator dynamics," *International Journal of Automotive Technology*, vol. 15, no. 7, pp. 1155–1164, 2014.

[50] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.

[51] H. Peng and M. Tomizuka, "Lateral control of front-wheel-steering rubber-tire vehicles," 1990.

[52] J.-w. Choi, R. Curry, and G. Elkaim, "Continuous curvature path generation based on bezier curves for autonomous vehicles," *IAENG International Journal of Applied Mathematics*, vol. 40, 05 2010.

[53] C. Lamini, S. Benhlima, and A. Elbekri, "Genetic algorithm based approach for autonomous mobile robot path planning," *Procedia Computer Science*, vol. 127, pp. 180–189, 2018.

[54] B. Song, Z. Wang, and L. Sheng, “A new genetic algorithm approach to smooth path planning for mobile robots,” *Assembly Automation*, vol. 36, no. 2, pp. 138–145, 2016.

[55] R. Oliveira, “Planning and motion control in autonomous heavy-duty vehicles,” 2014.

[56] D. Whitley, “A genetic algorithm tutorial,” *Statistics and Computing*, vol. 4, no. 2, pp. 65–85, 1994.

[57] Y. Chen and J. Zhu, “Trajectory tracking and control for nonholonomic ground vehicle: Preliminary and experimental test,” p. V003T37A007, 09 2018.