Half Title Page

Title Page

Contents

I

This is a Part

CHAPTER 1

Intelligent Decision-Making and Motion Planning for Automated Vehicles

Rick Voßwinkel

IAV GmbH, Chemnitz, Germany

Maximilian Gerwien

IAV GmbH, Chemnitz, Germany

Alexander Jungmann

IAV GmbH, Chemnitz, Germany

Frank Schrödel

Automation Technology & Robotics, Schmalkalden University of Applied Sciences, Germany

CONTENTS

1

[1]We gratefully acknowledge the financial support of this work by the Federal Ministry for Economic Affairs and Energy under the grant 50NA1912

FUTURISTIC AND AUTONOMOUS VEHICLES are currently a trendy topic with a strong growing market, see e.g. [1]. These systems promise a high level of automation. Through the last three decades, this topic has been drawing great attention from both academia and industry. Current assistance systems on the automotive market primarily support the driver through monitoring and warning functions or simple vehicle guidance tasks on well-defined areas such as parking lots or highways. The presented article focuses on (rule-based and Artificial Intelligence (AI)-based) decision-making and (model-based and model-free) motion planning. Perception tasks like object-fusion, lane and free-space detection are neglected.

1.1 INTRODUCTION

The core component for realizing Automated Driving (AD) beside the actual vehicle platform and the sensors for perceiving the environment is the Automated Driving System (AD-System), which enables a vehicle to perform distinct Dynamic Driving Tasks (DDTs) in a distinct Operational Design Domain (ODD). Briefly speaking, an ODD defines environmental conditions and restrictions, respectively, the presence or absence of certain traffic, and roadway characteristics [2]. A DDT, in turn, refers to the maneuvers to be performed, such as lane-keeping, lane-changing, stopping at a traffic light, or parking in a parking lot.

To increase the level of driving automation, AD-Systems have to be extended in two dimensions. On the one hand, continuous automation without relying on human drivers can only be achieved by incorporating high-level decision-making techniques that enable a vehicle to select, plan, and perform driving maneuvers to fulfill the parent mission according to the concrete situation currently prevailing. On the other hand, the scope of the respective ODD an AD-System can handle without human intervention has to be continuously expanded to be able to deal with a broader range of environmental conditions.

The functional components of an AD-System can usually be clustered into sensing (also referred to as perception), planning, and acting [3]. The sensing cluster is responsible for sensing and understanding the environment, e.g., by detecting lanes or objects such as other traffic participants or traffic signs. The planning cluster is responsible for implementing the deliberative behavior of a vehicle. Based on the observed environment, the planning cluster chooses the actual course of action. The planning cluster is typically divided into three different decision-making and planning levels: strategic planning for navigation and anticipatory driving tasks, tactical planning for orchestrating appropriate maneuvers given concrete situations, and operational planning for computing the actual path as well as trajectories according to the previously selected maneuver. The acting cluster is finally responsible for performing the previously chosen actions based on control strategies. As a result, inputs for the on-board vehicle control are achieved.

Let us consider a lane change as concrete maneuver. Performing a lane change on a highway requires at least knowledge about lanes and objects in the environment. Assuming that all the information can be robustly perceived, the planning cluster

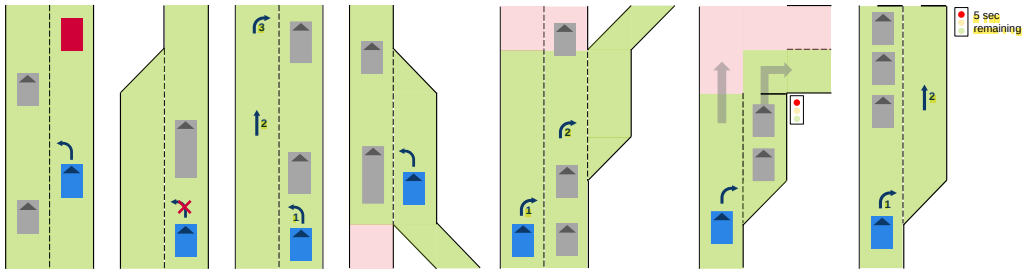


Figure 1.1: Lane change maneuvers for different reasons in different situations.

has to decide when a lane change might be the right course of action (e.g., when a vehicle in front is driving too slow). Furthermore, the planning cluster has to validate if a lane change is feasible (e.g., by checking that no other vehicles are blocking the target lane or won't block it soon). And finally, the planning cluster has to decide how to perform the lane change, e.g., by computing a path with a velocity profile that can be subsequently translated into a trajectory as input for a low-level control algorithm.

Since a lane change is a rather fundamental maneuver, it has to be performed for different reasons, in different situations, and under different environmental conditions. When we switch from the well-structured highway ODD to a far more complex urban ODD with more complex traffic rules, additional traffic participants, crossings, traffic lights, etc., the AD-System still has to be able to decide when, if, and how to perform a maneuver. Figure 1.1 shows lane change maneuvers of an ego vehicle (blue) in different situations. While the reason for performing a lane change differs from situation to situation, the overall goal is always the same: The ego vehicle has to follow a route (green lanes) to reach its final destination. Let us have a closer look at the different situations:

Figure 1.1a: The ego vehicle has to change the lane due to an obstacle (red) on the current lane. The adjacent lane might be completely or partially blocked by other vehicles (gray).

Figure 1.1b: The ego vehicle wants to change the lane due to a slowly driving vehicle in front. Changing the lane, however, would result in a critical situation, since the adjacent lane ends in x meters. As a consequence, performing a lane change - although desired - might not be a good choice.

Figure 1.1c: The ego vehicle wants to change the lane due to a slowly driving vehicle in front. After passing the first vehicle, the ego vehicle should usually adhere to the right-hand drive and change back to the right lane. However, to not unsettle human drivers by odd driving behavior and to keep a constant overall traffic flow, the ego vehicle might rather stay on the left lane and pass other slow vehicles before changing back to the right lane.

Figure 1.1d: The ego vehicle has to change the lane since the current lane ends in

x meters. The adjacent lane might be completely or partially blocked by other vehicles. This situation usually - but not exclusively - occurs when driving on a highway.

Figure 1.1e: The ego vehicle has to perform multiple, successive lane changes to leave the current road in x meters (e.g., when leaving a highway). Adjacent lanes, however, might be completely or partially blocked by other vehicles.

Figure 1.1f: The ego vehicle has to change the lane in order to turn right in x meters. The corresponding traffic light, however, is currently red. As a consequence, the right lane is blocked by vehicles that are waiting for the traffic light to turn green. The ego vehicle has to wait until the right lane is not completely blocked anymore.

Figure 1.1g: The ego vehicle is approaching a crossroad controlled by traffic lights. Both lanes are valid for staying on the route. The current lane, however, is blocked in x meters due to vehicles waiting for the traffic light to turn green. Since the corresponding traffic light is still red for another 5 seconds, the ego vehicle might change the lane to not completely stop but keep a smooth traffic flow before the traffic light turns green.

Scenarios (a)-(e) are neither restricted to highway nor urban environments but can occur in different environments with different environmental conditions. Scenarios (f) and (g), in turn, can usually be found in urban settings.

The described situations were not arbitrarily defined by us to simply motivate the necessity for intelligent decision-making and planning approaches. The situations represent challenges that we already partially tackled in past projects and that we especially face in our latest projects. Two of our recently completed projects are SYNCAR [4] and HarmonizeDD [5]. In SYNCAR we constructed an experimental vehicle by equipping a series vehicle with additional sensors, computing hardware, and communication capabilities. Furthermore, we successfully designed, implemented, and tested innovative driving functions for cooperative, automated driving in an urban environment. The same experimental vehicle was used in HarmonizeDD for successfully testing our designed and implemented connected, automated driving functions in mixed traffic scenarios. In this context, mixed traffic refers to the interaction between both automated and non-automated vehicles in the same scenario. For more details, please refer to [6]. In the running project AutoAkzept [7, 8] we are focusing on driver acceptance and minimizing uncertainty. Besides new HMI-concepts, driving behaviors for automated vehicles are examined. The objective of the project OPA^{3}L [9] is to automate recurrent maneuvers in known areas and in particular to present possible solutions for cooperative maneuvers in such areas.

While SYNCAR, HarmonizeDD, AutoAkzept, and OPA^{3}L address automated driving solutions for private transport, two of our latest and still running projects focus on public transport: ABSOLUT [10] and HEAT [11]. The overall vision of both projects is the development of so-called people mover, i.e., shuttles that drive autonomously on public roads to transport a limited amount of passengers. In ABSOLUT, IAV is in charge of realizing planning and acting algorithms that can cope with

low-speed scenarios to drive in an exhibition environment with lots of people around (Leipzig trade fair) as well as medium speed scenarios for driving on public roads. For HEAT, in turn, IAV is responsible for the entire shuttle development including electric embedded architecture, climate, chassis control, exterior design, sensor concept, and - of course - automated driving functionality. The HEAT shuttle will drive in the harbor city in Hamburg at the ITS world congress 2021.

While each project - in its own right - poses a great challenge, we make huge efforts to follow a holistic development policy that allows us to gradually extend and improve the capabilities of our AD-System with each project. One very essential part is the development of intelligent decision-making and planning (cf. Section 1.4) as well as motion planning (cf. Section 1.5) approaches for coping with the significantly increasing complexity across all past and current projects while simultaneously keeping future expandability in mind. Without an all-encompassing functional architecture (cf. Section 1.3), however, this long-term strategy would not be feasible at all.

1.2 OPTIMIZATION – A KEY INGREDIENT FOR AUTOMATED DRIVING

The scope of this section is to introduce optimization as a fundamental aspect for AD. We introduce some optimization basics and motivate that optimization is used and sometimes necessary in nearly all parts of an AD-System such as sensor fusion, data acquisition, maneuver planning, and control optimization.

To achieve robust and comfortable performance, we need optimal behavior in the individual steps of the AD-System. The question now is: What is meant by the term optimization? Several people might have a different understanding of what optimization means. Neither considering different designs or configurations and choosing the best one nor following some suggestions or rules to improve the overall behavior is the way we understand optimization. In fact, we use the term optimization from a mathematical point of view: Optimization describes a systematical procedure to find an optimal (at least a local one) solution to a given problem.

The initial point of every optimization is a proper problem formulation such as

$$\begin{aligned} &\min\ J = F(\dots)\ \text{s.t.} \\ &\sum_i g_i(\dots) = 0 \\ &\sum_j h_j(\dots) > 0, \end{aligned} \tag{1.1}$$

with the cost functional $J = F(\dots)$. This cost functional defines the main optimization goal. The arguments of the function F depend on the particular problem. Often, arguments like time, system states (velocity, acceleration, jerk, ...), and more abstract ones like comfort are considered. In other words, we define with the cost functional what is important for us: Do we want to drive sportive or defensive? Do we have to follow the shortest or the fastest route? Do we prefer a comfortable driving behavior over a strict and precise driving behavior? The functions g_i and h_i define inequality and equality constraints. Such constraints are used to describe physical

limitations or other saturations and conditions resulting from law or comfort considerations. The method or algorithm we choose to solve a given optimization problem is determined by their properties. While linear and convex problems are generally very effectively solved with gradient-based methods [12], we need more sophisticated algorithms to find global optima of nonlinear problems. Furthermore, we need to differ between a parameter optimization, where we want to find an optimal set of parameters solving the optimization problem and an optimal control problem, e. g. [13], where we are looking for an optimal input function for a given problem.

The first real systematic technique for optimization was proposed by Fermat [14]. Nowadays, setting the first derivatives to zero to compute extrema is one of the basics of every textbook on calculus and a frequent application of differentiation. After Fermat, the iterative approaches of Newton [15] and Gauss were the next steps to search for an optimum [16]. This iterative idea is picked up several times, like in the steepest gradient method, which can be traced back to Riemann and Cauchy [17], the celebrated simplex method of Dantzig [18] or Quasi-Newton methods [19]. At the beginning of the 1980s, meta-heuristic approaches, like genetic algorithms [20], simulated annealing [21], and particle swarm optimization [22] became popular. These algorithms are inspired by nature or other heuristics and have some benefits, such as global optimization and a gradient-free computation. Nevertheless, they need a lot of sample points to converge to the optimum which gives a computational barrier for applications, even they generally support parallel computing.

Nowadays, Machine Learning (ML) approaches became more and more relevant. These data-driven methods construct a meta-model based on sample points and are able to learn how to solve detection, classification, planing, control, or prediction problems. In the ML domain, three types of learning have prevailed [23, 24, 25]:

Supervised Learning: Supervised Learning considers input and desired output data to minimize the error between the predicted outcome and the labeled output data.

Unsupervised Learning: Unsupervised Learning approaches only consider the input data and try to find structure and patterns in the provided data-set.

Reinforcement Learning: Reinforcement Learning (RL) techniques learn the desired behavior by interacting with the environment (put simply: learning by doing) [26]. Reward functions evaluate the outcome of each action.

ML techniques are mostly inspired by human learning behavior and psychological models.

Optimization can be found in nearly every part of the AD structure. Beginning with modeling of the vehicle dynamics, which is generally based on minimizing the cumulated modeling errors (e. g., [27]). The part of the introduced AD-System which is nearest to the vehicle is the control. Optimization either direct to the control strategy [28] or indirect regarding the control quality (e. g., [29]) is significant for the successful implementation. The next layer is the operational planning, which mainly consists of a trajectory planning module or a Model Predictive Control (MPC). Since

optimization is the key feature of these layers, they will be described in more detail later on. The decision-making levels are the levels that decide which action to choose based on the presented environment and the interpreted situational context with a long-time prediction horizon. The optimization goal here, is to maximize the expected reward or value function to get the most beneficial policy over all actions [30, 31]. The whole structure is closed via sensor fusion and perception. In this part, we can found optimization as well, like Kalman filters or Moving Horizon Estimation (MHE) [32, 33].

1.3 FUNCTIONAL ARCHITECTURES FOR AUTOMATED DRIVING

Choosing an appropriate functional architecture for AD is fundamental for making the right decisions in line with requirements regarding the intended driving behavior. Mastering the complexity of AD, especially in urban traffic scenarios, while simultaneously adhering to functional safety requirements is a challenging task that has to be considered in the architecture and the methods of decision-making from scratch.

Two classical architecture approaches can be pointed out in automated systems [34, 35, 36]. The first one is referred to as function-based, top-down, or knowledge-driven architecture. It uses an explicit world model for decision-making and planning as well as couples deliberative agents with explicit knowledge. An overall mission for each agent is decomposed into sub-goals within a hierarchical control structure. That is, higher layers in the hierarchy create sub-goals for the lower layers. This includes the way of sense-model-plan-act paradigm such as in the standard NASRAM architecture [37]. In consequence, the high-level side has long planning cycles and no fast reactions to dynamic environment changes. Deliberative agents are well suited for structured and highly predictable environments. This approach provides also the benefits of mission-oriented planning and high-level intelligence. However, top-down architectures require a general world model that is hard to represent and also computationally expensive. Therefore the world representation is often reduced to topological and semantic maps. Furthermore, purely deliberative agents are not sufficient to realize intelligent behavior because of over-simplification [38, 39]. The problem of static world representation is the absence of intuitive interpretation and solutions in dynamic environments.

The second architecture approach, in turn, focuses on reactive agents. Reactive agents implement decision-making and planning functionality by bottom-up or data-driven models that react to their sensory input directly without high-level planning and overall missions [34, 38, 39, 40]. The aim is a timely robotic response in dynamic and unstructured worlds. The main advantage is the robustness of the directly coupled approach of perception and action (behavior). For a purely reactive approach to be applicable, however, actions (or driving functions in our case) have to be explicitly defined and made available in advance, leading to a deficiency of flexibility in the architecture. For an example of this approach, see also the subsumption architecture [41].

Different hybrid architectures were developed to combine the benefits of both deliberative and reacting agents and to eliminate or at least minimize the drawbacks of

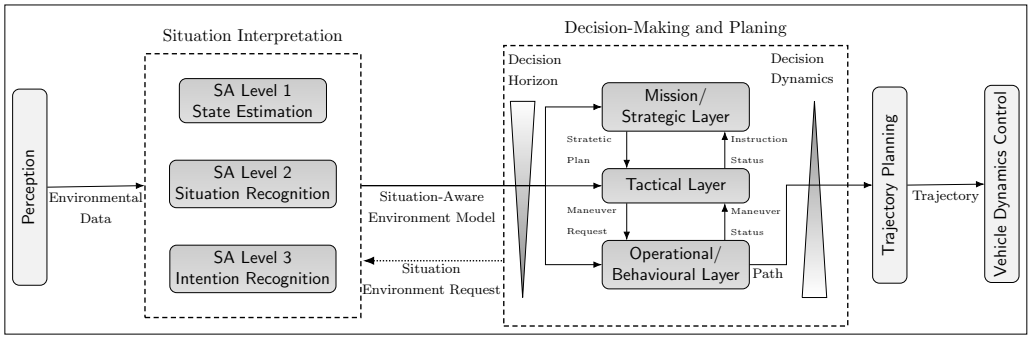


Figure 1.2: Functional architecture for our AD-System.

each approach. In AD hybrid models are in fact the most common approach and gain a lot of attention in research [42, 35, 43, 44, 45]. Reactive and deliberative functions are typically combined within a three-layered architecture [46, 47]: the deliberative layer, the sequencing layer, and the reactive layer. The deliberative layer is responsible for long-term strategic mission planning including mission adaption. The sequencing layer, in turn, coordinates the interaction between the deliberative and the reactive layer to execute appropriate tactics using context-dependent rules. The term tactic, in this context, refers to a pre-defined, ordered, and structured set of actions. The reactive layer acts with metric information and numerical control. The difficulty lies in the interface of these different layers because they work asynchronously, with different time scales and data representations.

Choosing the most appropriate functional architecture for AD usually depends on the application and the corresponding requirements. Early work in this context was inspired by the real-time control system domain and might be interpreted as Active Seeing (AS) [48, 49, 50, 51]. These architectures were the breakthrough for developing functional hybrid architectures for AD and are particularly characterized by their hierarchical planning. Furthermore, the well-known DARPA challenge gave rise to various hybrid architectures [52, 53, 54, 55], where each architecture was specialized for solving the course defined by the challenge. Urban scenarios, however, were only partially considered. Later architectures focused more on urban environments. The first experience was collected with projects like Stadtpilot by TU Braunschweig [56] or the PRORETA 3 project by TU Darmstadt [57]. Further research in functional architectures for AD finally led to a 3-layered decision-making approach consisting of a strategical, a tactical, and an operational layer [58, 59, 60].

In our work, we gradually adapt and refine this three-layer approach to serve as holistic architecture that incorporates any AD functionality that is already available, currently in development, or might be developed in the future. By doing so, we ensure that the performance and consequently the level of automation of our AD-System increases systematically over time. Figure 1.2 gives a rough overview of our functional architecture. The Perception cluster is responsible for perceiving the environment and will not be further discussed in this chapter. The Situation Interpretation cluster translates the environmental data into an environmental model (or situation model)

according to the decision-making and planning level that requires the respective information. Information maximization and reduction (e.g., by selection or abstraction), as well as information representation, are key aspects. Both clusters are indeed tightly coupled and allow for situation-aware decision-making and planning. Section 1.4 gives a detailed overview of the two function clusters and the respective layers of hierarchy within the clusters while Section 1.5 introduces concrete algorithms for motion planning on the operational layer. The Trajectory Planning cluster translates geometric paths with velocity annotations into trajectories or even alternative representations (cf. Section 1.5). The Vehicle Dynamics Control cluster is finally responsible for generating control variables such as steering angle and acceleration.

1.4 SITUATION INTERPRETATION AND DECISION-MAKING

The decision-making process for intelligent high-level planning vehicles requires a hierarchical and situation-aware representation of the world. The focus of this section is presenting methods that handle, in a safe manner, decisions based on uncertain and incomplete information. Therefore we present the underlying optimization problem in information and decision theory and give a short overview of different applications and tasks in automated driving.

1.4.1 Environment Presentation and Situation Interpretation

Situational interpretation is the elaboration of the situational context, whereby not only the current situation is considered, but also past conditions create a certain awareness of a certain situation. Awareness of a situation thus involves a focus on possible resources and possible future events. This time-dependent and uncertain process is also called Situation Awareness (SA). The concept of SA has its origins in psychology [61, 62, 63] and serves as the basis for any situation-related decision and risk assessment. The model [61] originally has three levels, which can be subdivided with regard to the decision-making process:

SA Level 1: Estimation and representation of the current environment and entities.

SA Level 2: Comprehension, classifying and combining with mission-related goal to the current situation.

SA Level 3: Prediction and intention detection of future situations and status.

Figure 1.2 shows the adapted concept of SA included in the functional architecture of our AD-System. Thus the situation interpretation and thus the different levels serve as a service for the various decision-making processes.

The difficulty in the SA Level 1, to present a suitable environment, lies in the adequate combination to one consistent Comprehensive Environment Model (CEM) [64] that acts as a generic abstraction layer for the decision-making process. Since the environment representation can be metric, topological, static, dynamic, and semantic, a lot of effort has to be done to combine those representations to provide each functionality the corresponding information and requirements [65]. Each Decision Layer

subsumes a solid situation interpretation of the scenario and situation depending on the task and the goals of each decision-making layer. But not the hole CEM is necessary for each decision layer such that hierarchically structured information has to be considered.

SA Level 2 generally conveys an understanding of the specific scenario and can recognize the situation in that scenario. The main task is to classify and recognize, based on SA Level 1, different scenarios and situations. To achieve this goal three different approaches are conceivable. First, information from static data like roadmaps provides the situation or the scenario. Second, expert-systems with pre-defined rules divide situations [66, 67, 68]. As a third point, SA Level 2 learns in a supervised manner to classify [69] pre-defined scenarios.

The SA Level 3 will estimate and predict future states in a special situation [65, 70, 71, 72, 73]. Besides the prediction of future trajectories and movements of an object in the situation also the prediction of maneuver, interactions, and indicators like collision-risk or time-to-x values are tasks in this level.

Problem Formulation of Situation-Aware Interpretation

There are five main factors of uncertainty in robotics and thus also in AD-Systems [74]. The inherently unpredictable environment, the perception by sensors, the inaccuracy of actuators, every system model and each calculation with the computer system comes with uncertainties. In consequence, uncertainties have to be considered and could be also modelled. Therefor we have to define a stochastic process. Based on the stochastic process, the focus is on information-based cost functions from which the underlying optimization task can be derived.

A stochastic process is characterized by the fact that the random variable X depends on a parameter t. In most cases, the parameter t represents the time. A stochastic process $X = \{X_t, t \in T\}$ is thus a family of random variables. In the following it is always assumed that T is the index set for the time. The realization of X_t for all points of time $t \in T$ results in a real function $x = x(t)$, which is also called trajectory or time sequence process. The trajectories can be discrete or continuous where the state space contains measurable metric or symbolic states [75].

Important for the perception and measurement of a stochastic process is the information which each measurement receives for a realization $P(X = x)$. With the assumption that not every realization receives the same amount of information, the Shannon information

$$H(X = x) = \log \frac{1}{P(X = x)} = -\log P(X = x) \tag{1.2}$$

can be expressed. If the information measure were zero, the process would be fully predictable from the information a-priori available from $P(X)$. Thus the variable x would also be deterministic and no longer stochastic.

Two main tasks can be pointed out in situation aware interpretation. At first the well-known problem to model a dynamic system, dynamic situation, or just situation aspects from observed data set Y_t. The aim is to estimate X_t from the previous values Y_s, where at any time t only $\{Y_s : s < t\}$ previous values can be observed. A solution,

for the so-called filter problem, is a modeling task that calculates the system dynamics and the observation dynamics explicitly or implicitly. This can be expressed with the state space system

$$x_t = f(x_{t-1}, u_{t-1}, \epsilon|\theta) \tag{1.3}$$
$$y_t = g(x_t, \sigma|\theta). \tag{1.4}$$

A parametrization of the functions f and the observation dynamics g is necessary. The solution to the filter problem and the modeling of the stochastic system can be very different depending on the methodology. The following aspects have to be considered:

- Are probability distributions known from the process or sub-processes?
- What properties are subordinated to the process (Markov property, Bayesian process etc.)?
- What parameter estimation techniques are used?

The second main task that can be pointed out is the selection of good data. Especially data-driven classifiers or planning algorithms need more quality in the data instead of huge data amounts for finding the optimal solution. An observer that is actively involved in the perception of actions when these are performed by a demonstrator is called AS [51, 76]. Besides the integration of active seeing in the functional architecture in AD, which has to be organized in a hierarchical and distributed manner, a measure of information gain is needed to find the right queries for each decision level and a suitable information-based cost function. To maximize the information gain and to minimize the amount of needed data in the sense of a-priori knowledge and posterior data is also called Active Learning (AL) [77].

The Information Gain (IG) indicates how much information a feature gives us about the class. This is normally expressed by calculating the entropy

$$H(X) = -\sum_i f(x_i) \log_2 f(x_i) \tag{1.5}$$

of a stochastic process X. Calculating the minimum of entropy is maximizing the IG.

An information-based cost function can be described with similar information measures and will also model the notions of IG. In the case of modeling the distribution with the parameter-set θ, the expected value of the IG is also known as the Kullback-Leibler (KL) divergence and describes the mutual information between an a-priori distribution $P(x)$ and the posterior distribution $Q(X|\theta)$. The expression

$$D_{KL}(Q\|P) = \sum_i Q(x_i|\theta) \log_2 \frac{Q(x_i|\theta)}{P(x_i)} \tag{1.6}$$

measures how different the distributions are, where D_{KL} is zero if and only if both distributions are equal. The optimization criterion

$$\min J = D_{KL}(Q\|P) \tag{1.7}$$

has to be minimized to find the optimal model.

This leads to a modeling problem where not only uncertainty is considered but also the non-linearity and the dependency of different states. A lot of applications in AD research use this approach directly or indirectly to predict future states to observe intention or maneuver symbolically [65], but also in a metric manner to predict future trajectories and interactions of surrounding participants [70, 71, 72, 73].

In the sense of AL and choosing unseen data, we also can use the KL divergence. Modeling the KL divergence of the posterior measures the amount of IG expected from the query (where $x^{'}$ is the queried data):

$$\max J = D_{KL}(Q(\theta|x, x^{'})\|Q(\theta|x)) \tag{1.8}$$

Here the goal is to choose a query that maximizes the KL divergence between posterior and prior data, such that the largest KL divergence between updated posterior probability and the current posterior probability represents the largest gain.

In the model for the solution of the filter problem and the active learning part the KL divergence is usually chosen, because there the posterior and the prior have to be compared.

1.4.2 Intelligent Decision-Making for Maneuver Selection

Decision-making is essential for planning, selecting, and finally performing maneuvers. In Figure 1.2, the decision-making and planning cluster is hierarchically structured into multiple layers:

Mission Layer: The mission layer provides all long-time goals and tasks that come with the intention of the human driver. It encompasses the global graph search algorithm to find the route in the road network under consideration of constraints like the fastest ways or resource-saving.

Strategic Layer: The strategic layer includes the graph search algorithms to find the optimal route of roads in a predefined horizon and even driving lane-specific segments to reach the driving destination. Vehicle-to-Everything (V2X) information can be used to avoid construction sites or highly driven roads or to adapt the driving behavior concerning critical weather forecasts. The strategical layer and the mission layer mainly incorporate global sensor information for decision-making.

Tactical Layer: The tactical layer can be considered as the interface between the high-level planning side and the low-level behavioral planning side, where the data structure between these layers may vary. This layer uses global and intern sensor information as the foundation of decision-making. The tactical layer is responsible for modifying the planned lane-specific navigation in such a way that it fits with the driving maneuvers of other traffic participants. The abstract lane change advises from the strategical layer is rendered into a concrete collision-free lane change at a spatial location and point in time. Therefore the different possible spatial points have to be observed and selected for the

optimal gap depending on constraints like driving style, alternative navigation routes, or V2X communication. Also, this layer generates a sequence of predefined operational functions for achieving more complex maneuvers such as overtaking maneuvers or it may generate acceleration and deceleration recommendations for the operational layer. Depending on the operational layer and the implemented behaviors, the tasks of the tactical layer can be very different and includes symbolic (maneuver selection) and also metric (e.g., advise of acceleration) decisions. In any case, these decisions always have to take prevailing and emerging situations into account.

Operational Layer: The operational layer subsumes everything necessary to translate maneuvers selected on the tactical decision-making layer into valid paths and trajectories. The corresponding driving functionality may also receive parameter sets from the tactical layer to realize the same maneuver in different scenarios and to adapt optimization functions. More details regarding the operational layer can be found in Section 1.5.

Top-Down Decomposition

For better understanding the interaction between the different layers, let us consider the situation depicted Figure 1.1e. First and foremost, the ego vehicle has one overall mission: To follow a given route (indicated by green lanes) to reach a final destination. While the decision and planning horizon usually comprises the entire area between the current position and final position, the decision dynamics (i.e., the frequency of adjusting the route due to unforeseen events) is generally rather low.

To complete a mission (i.e., to follow a dynamically changing route), the strategic layer decomposes the route into a sequence of so-called complex maneuvers such as “follow course of road” or “turn left”. In Figure 1.1e, we have a transition between two complex maneuvers: While the ego vehicle is still performing the complex maneuver ”follow the course of the road”, it has to prepare for the next complex maneuver “take exit”. Formally speaking, while performing the current complex maneuver, the ego vehicle has to make sure that the preconditions of the subsequent complex maneuver are fulfilled in time. In this concrete situation, the ego vehicle has a very restricted time frame for the first lane change to meet the precondition (drive on right lane) for taking the exit.

The tactical layer is responsible for decomposing complex maneuvers into sequences of basic maneuvers (or simply: maneuvers). A lane change is a basic maneuver that serves as one of many building blocks for realizing complex maneuvers. In Figure 1.1e we have two explicitly indicated basic maneuvers in terms of the two-lane changes. However, we can see a complete chain of basic maneuvers. The ego vehicle is currently performing the maneuver “follow lane” while waiting until the preconditions for performing the lane change are met - either by passively observing how the situation evolves or by actively trying to change the current situation. After performing the first lane change, the ego vehicle follows the new lane. In either case, the ego vehicle is continuously performing the complex maneuver “follow the course of the road” on the strategic layer.

To execute a basic maneuver, the tactical layer interacts with the operational layer, which provides driving behaviors or actions as the most atomic building blocks in this hierarchy. The maneuver "follow lane" in Figure 1.1e, e.g., can be further decomposed into at least two parallel actions: One for longitudinal driving behavior such as Adaptive Cruise Control (ACC) and one for lateral driving behavior for keeping the lane. In the past, there have been different approaches to making decisions about the choice of behaviour or the execution of the planning maneuver, and currently there are no uniform solutions or an optimal solution. The tactical layer in particular plays a special role, as the interface to the operational layer has to be defined and this is where usually encounter difficulties. This is also due to the fact that planning and decision-making depend on the functional architecture and also to the fact that they work asynchronously, with different time scales and data representations. According to Arkin [34] there are four main interface strategies for the different hybrid architectures (selection, consultation, adaptation, shifting). In automated driving the Selection and adaptation can be observed. The selection strategy is basically the sequential generation of various predefined behavioral functions. Consequently, the predefined set are atomic functions that are selected and combined at the tactical layer to fully meet the mission objectives. The adaption is regarded as an adaptation system in which the tactical layer continuously changes the behavioral function in the light of changing conditions, scenarios and situations. This can be achieved by parameterization or consulting for scenario-dependent constraints.

Composition Strategies and Methods

By using building blocks to compose complex functionality on a higher hierarchy layer, we can flexibly apply different composition methods on different layers while taking decision dynamics and horizons into account. Composition, in this context, refers to coming up with a plan to achieve the goal on the next higher hierarchy layer. Since automated vehicles have to deal with a non-deterministic, highly dynamic environment, planning, executing a plan, and revising a plan might also be tightly interwoven - especially on the lower hierarchy layers.

A very common and straightforward approach to compose high-level functionality is to encode all possible plans in state machines when the AD-System is designed. Roughly speaking, depending on a current state (or situation) and according to predefined transitions with transition conditions between the states, a state machine clearly defines in which situation and under which condition an action (e.g., a complex or basic maneuver) may be selected. Although this approach is indeed very intuitive, it is also very restricted: Knowledge, which was not encoded into the state machines during design time, is not available when the AD-System is online and running. That is, the developers and domain experts have to consider all situations that may occur in the desired application and precisely define all relevant transitions and corresponding conditions in advance. For a very restricted ODD and a limited set of DDTs, this approach usually works pretty well, since a thorough investigation of possible situations and necessary maneuvers is still manageable, while state machines that encode the result of the investigation are still maintainable. With increasing AD

functionality, however, this approach is not feasible anymore and should be replaced by AI approaches in terms of Automated Planning and ML techniques.

Purely symbolic AI planning approaches can be used to separate domain knowledge from the actual planning process, to facilitate online planning [30]. This strategy is quite similar to solving the service composition problem in the service-oriented computing domain. For example, image processing services are interpreted as building blocks to compose more complex image processing functionality, where the functionality of building blocks is described based on ontologies [78]. In our context, the functionality of maneuvers can be described to define 1) under which condition a maneuver may be performed and 2) what kind of effects performing a maneuver may have. Having functional specifications of maneuvers available, an online search algorithm can identify a valid sequence of maneuvers that transforms the current situation into the desired situation. In any case, identifying whether performing a maneuver in a situation at hand is valid or not is not explicitly defined in terms of static transitions between pre-defined states anymore, but can be flexibly computed online.

Purely symbolic approaches, however, have multiple drawbacks. While they indeed facilitate flexible and automated composition of high-level functionality, they do not work well on lower layers where less abstract environmental models are required, e.g., to cope with uncertainty and higher dynamics. Furthermore, symbolic approaches usually only allow for identifying valid maneuvers. Whether one valid maneuver is more appropriate then another valid maneuver in a specific situation cannot be evaluated. That is, purely symbolic planning algorithms do not rate or rank alternative solutions to decide which is the best one.

The normative decision problem can be described mathematically [30], such that there is a choice of various actions and the environment is presented with the set of states. The actions to the environment may influence the states; such that states have a conditionally probability distribution. For the rating of each action, there is a utility function which specifies the payoff if the state is located and an action is chosen. So, the task of normative decision theory is to select the action that maximizes the expected utility.

A straightforward solution for this decision-making problem is to incorporate hand-crafted, domain-specific heuristics to determine the value of a maneuver given the current situation. Yet again, we face the challenge that developers and domain experts have to define such heuristics in advance. Alternatively, symbolic planning approaches can be extended [79] or replaced by reward-based RL techniques based on Markov Decision Processes (MDPs) [80], which is a general framework to model planning and decision making problems. Formally, a classical MDP $\mathcal{M}$ is a quintuple

$$\mathcal{M} = (\mathbb{T}, \mathbb{S}, \mathbb{A}, p_t(\cdot|x,a), r_t(x,a)),$$

where the ingredients are defined as follows:

- $\mathbb{T} = \{1, 2, ..., N\}, N \in \mathbb{N}$, is a discrete finite set of decision epochs with $t \in \mathbb{T}$ representing a point in time when a decision is made.

- $\mathbb{S}$ is a discrete finite set of states with $x_t \in \mathbb{S}$ being the state occupied at decision epoch $t \in \mathbb{T}$.
- $\mathbb{A} = \bigcup_{x \in \mathbb{S}} \mathbb{A}_x$ is a discrete finite set of actions with $\mathbb{A}_x$ being the set of possible actions in state $x \in \mathbb{S}$.
- $p_t(x'|x,a) \in [0,1]$ is the transition probability at decision epoch $t \in \mathbb{T}$ for transitioning from state $x \in \mathbb{S}$ into state $x' \in \mathbb{S}$ when performing action $a \in \mathbb{A}_x$.
- The expected reward at decision epoch $t \in \mathbb{T}$ for being in state $x \in \mathbb{S}$ and performing action $a \in \mathbb{A}_x$ is defined by

$$r_t(x,a) = \sum_{x' \in \mathbb{S}} r_l(x,a,x') \cdot p_t(x'|x,a)$$

  with $r_l(x,a,x')$ being the immediate reward for transitioning to a successor state $x' \in \mathbb{S}$.

A policy $\pi = (\pi_1, ..., \pi_t, ..., \pi_{N-1})$ specifies for each decision epoch $1 \leq t < N$ a decision rule π_t in a deterministic ($\pi_t : \mathbb{S} \to \mathbb{A}$) or in a stochastic ($\pi_t : \mathbb{S} \to P(\mathbb{A})$) manner. Solving an MDP is equivalent to finding an optimal policy π^* that maximizes the discounted cumulative reward

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1},$$

with the discount factor γ, $0 \leq \gamma \leq 1$.

The expected cumulative reward of a policy π is usually defined as a value function, where a distinction is made between *state-value* and *state-action-value* functions. The state-value function

$$V^{\pi}(x) = E_{\pi}\{R_t | x_t = x, \pi\} \tag{1.9}$$

represents how good it is to be in a state x. It depends on the policy that performs the actions. The state-action-value function

$$Q^{\pi}(x,a) = E_{\pi}\{R_t | x_t = x, a_t = a\} \tag{1.10}$$

is an indication for how good it is to chose an action a while being in state x. Regarding the state-value function (1.9), the optimal policy π^* for an MDP is defined as

$$\pi^* \in \arg\max_{\pi \in \Pi} V^{\pi}(x), \tag{1.11}$$

where Π is the set of all possible policies. The optimal policy for the state-action-value function (1.10) is achieved by

$$\pi^* \in \arg\max_{a \in \mathbb{A}_x} Q^{\pi}(x,a). \tag{1.12}$$

If complete knowledge of the environment is available, i.e., all elements of an

MDP are known, dynamic programming algorithms such as Value Iteration or Policy Iteration can be applied to compute π^* [26]. If complete knowledge is not available, RL techniques can be applied in order to maximize the *expected* cumulative reward in the long run. In this case, the algorithm has to actively learn through the experience of interactions with the environment and only the current possible states and actions can be observed.

MDPs are indeed a sound mathematical framework for sequential decision-making. Classical MDPs consider discrete state and action spaces as well as discrete time intervals for performing actions. Furthermore, classical MDPs incorporate transition probabilities to model uncertainty when performing actions (e.g., when a maneuver does not bring the desired result). However, classical MDPs assume that the environment is always completely observable. For that reason, Partially Observable Markov Decision Processes (POMDPs) additionally incorporate observation probabilities to not only model the uncertainty when acting but also when perceiving and trying to interpret the situation [81]. Another relevant extension to classical MDPs are Semi-Markov Decision Processes (SMDPs). SMDPs consider the duration of actions to be randomly distributed. This is essential for AD, in particular for the lowest hierarchy layers, since performing the same maneuver usually takes different time.

When a decision-making problem is modeled as a MDP, it can be solved using RL techniques, either directly or by trial and error. Even hierarchical learning becomes possible [82]. Solving a MDP is equivalent to finding or approximating an optimal policy that maximizes the cumulative reward in the long run. If complete knowledge of the environment is available, i.e. the entire model is known, dynamic programming algorithms such as Value Iteration or Policy Iteration can be applied to directly compute the optimal policy. Otherwise, model-free approaches like Q-Learning are available [83].

1.5 ANYWHERE, ANYTIME, ANYWISE: MOTION PLANNING PROBLEM

1.5.1 Path Representation and Trajectory Generation Problem

As a result of the decision-making process illustrated in Section 1.4 we know what needs to be done. This section focuses on the realization of these maneuvres. In order to compute proper driving trajectories, we have to place ourselves in the environment and have to determine the situation around us, especially road conditions including lanes, road side units, other road users, and many more. For that purpose AD-Vehicles are equipped with a bunch of sensors, like Lidar, cameras, and ultrasonic sensors. The data of these sources are fused and offer us a good idea of the situation around us. With that in hand and requested maneuver a suitable path can be computed.

A common method for path calculation is the potential field method. The potential field method is inspired by the physical phenomena of magnetic fields and was described by Khatib [84] for the first time. Initialized with a starting point and an endpoint, repulsive forces, and attractive forces are introduced to create a potential

field by forming the negative gradient of a potential function. In a general potential field, algorithms can fulfill real-time requirements, but they lead often to local minima. Thus a non-optimal path is computed or the algorithm cannot find a proper solution. Beyond that, potential field algorithms are not able to handle kinematic constraints.

In the context of graph search-based algorithms, the congruence space of the vehicle is approximated as a grid or lattice. In [85] is the Dijkstra algorithm used to adaptively determine the way-points according to the road situation. The A* algorithm is a grabbing heuristic search algorithm. In practical applications, the performance is related to the resolution of the map restricted. In 2008, Hybrid-A* was introduced in the Standford self-driving Junior vehicle used in the DARPA Urban Challenge [86]. Under the consideration of curvature and kinematic constraints, the state Lattice algorithms were developed and successfully applied in unstructured environments [87]. However, the incremental properties of the search algorithm lead to an exponential growth of the computing complexity.

In our case is the path determined based on the earlier chosen maneuvers. Thus, if we want to stay in the lane, the path is the middle of the lane. If a lane change is necessary, a smooth function is fitted from one lane to the other, etc. This piecewise and maneuver-based path computation gives us the possibility to set up a smart interface between the different planning layers and enables a fast path adjustment due to new environment data.

The path consists of four parts. The first one is the reference which is located in the middle of the driveable corridor. The left and right boundaries of this corridor give the second and the third part, see Figure 1.3. The last one is a velocity profile, which represents the maximum permitted velocity in a step-function like representation. The path consists of points based on the arc-length containing these four pieces of information.

Now let us consider a lane change scenario, cf. Figure 1.1, to illustrate the path concept. While we are keeping the lane, the reference is the middle of that lane and the left and right boundaries are the boundaries including some safety distances. The velocity profile is determined by the traffic rules or other limitations (e. g. driver input). If a lane change is requested a new drivable corridor is generated. Thus the reference is planned from the middle of the actual lane to the middle of the desired lane and the boundaries are appropriately adjusted. The velocity profile is a-priori not modified, but it might be necessary, which is addressed in the following.

Adaptations to the planned driving path are necessary to ensure safety, comfort, and something often called ”human-like driving”. Thus, the path is subsequently adjusted based on different factors of influence. One of these factors is the Green Light Optimal Speed Advisory (GLOSA) vehicle guidance concept, which explicitly considers the current and the predicted states of multiple traffic lights ahead, including lane-use recommendations. This information can be used to adapt the vehicle's dynamics. Consequently, the vehicle can drive in urban situations with a high driving comfort as well as much more energy-efficiency, which can be used to enlarge the range of electric vehicles or save fuel while additionally improving travel time. The Predictive Velocity Adaption (PREVA) represents a predictive longitudinal vehicle

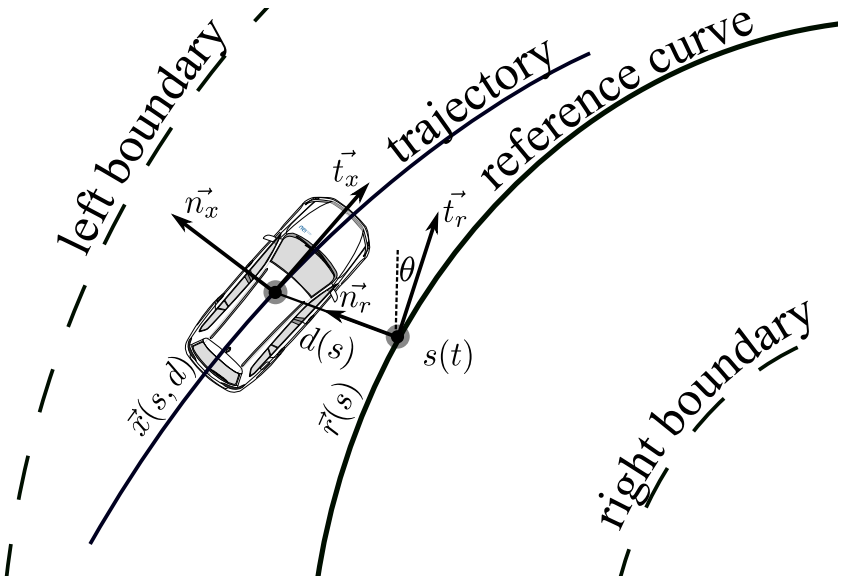


Figure 1.3: Frenét Space: trajectory and reference

guidance concept. Here adaptions of the reference velocity are realized based on road curvature or width, weather and road conditions as well as driver-related impacts, like mood or uncertainty. A safety layer compares the driving path against aggregated clearance information (i.e. low-level sensor data) to ensure collision-free driving. If necessary this layer will adapt the driving path to lead the vehicle to a full stop. An obstacle avoidance algorithm realizes to overtake small objects on the lane without needing to do a full lane change.

For the subsequent calculation, it is necessary to determine an analytical description of the point-cloud describing the path. Based on Beziér-Splines, a piece-wise polynomial description of the path is computed and called reference trajectory. These polynomials are represented as functions of the arc-length covered along the reference trajectory.

1.5.2 Proven Planing Approach

In this section, a model-free trajectory planning algorithm is introduced, which calculates an optimal longitudinal and lateral trajectory for the vehicle guidance based on the previously discussed reference path. For sure there are a bunch of approaches to tackle trajectory computation. In [88] and [89] trajectories are generated by searching in a discrete state lattice. The principle is that the state space is discretized, generating a large number of target states. In combination with optimization control strategies, this can generate jerk-optimal trajectories [90]. However, the high number of different combinations results in a sub-optimal solution with a considerable computational effort.

Another possibility gives the optimization of kinematic variables. This procedure is based on the mass point model and minimizes kinematic or geometric quantities

[91]. Various functions such as polynomials, sigmoids, or splines for trajectory planning are used followed by optimization. In [92] and [93] a trajectory was planned using continuous curvature polynomials. This algorithm first discretizes points on the road and connects them with curvature polynomials. By numerical optimization, an optimal trajectory is generated. However, the curvature is dependent on the speed and acceleration, the polynomials used and the cost-functional are not matched so that stability cannot be guaranteed [94].

MPC allows us to solve an optimal control problem cyclically by the feedback of the current system state and allows us to predict the behavior of the road users. By means of this basic idea, this can be applied to the motion planning of vehicles. In the work of [95] and [96] the planning is implemented with the MPC, which determines an optimal trajectory including a quality functional. In the work of [97] an approach based on the MPC for longitudinal and lateral guidance is developed. Due to the special formulation for trajectory optimization, a follow-up driving behavior on curved scenes can be realized. This planning is based on the Frenét coordinates and causes an efficient calculation for longitudinal and lateral guidance. The model-predictive method uses system models to calculate not only the optimal trajectory but also the optimal manipulated variables. The advantage of the MPC is its ability to integrate the constraints of the system. The disadvantage, however, is the computational effort that increases with the prediction horizon. In order to reduce the computational effort, an explicit MPC can be used, which provides an efficient calculation.

Due to the practical relevance, we will describe a model-free approach based on [90] and an MPC approach [98] more in detail.

In case of the model-free approach and due to the Frenét coordinate system, see Figure 1.3, we can individually solve the lateral and the longitudinal problem. Thus an optimal longitudinal ($s(t)$) and lateral ($d(s)$) trajectory is computed and merged afterwards. Using this Frenét system a position $\vec{x}(s(t), d(s))$ on a specific time t is described by:

$$\vec{x}(s(t), d(s)) = \vec{r}(s(t)) + d(s)\vec{n}_r(s(t)), \tag{1.13}$$

where $s(t)$ is the arc-length, $d(s)$ is the lateral offset on the respective path length and $\vec{n}_r(s(t)) = [-\sin\theta_r(s), \cos\theta_r(s)]^T$ is the normal vector on the path at the point $\vec{r}(s(t)) = [r_1(s(t)), r_2(s(t))]^T$, and $\theta_r(s)$ is the orientation of the path [99]. The main idea of the model-free trajectory is to solve the unconstrained lateral and longitudinal optimization problem analytically using Pontryagin's Minimum Principle [100].

1.5.3 Optimal Trajectory Computation – the Longitudinal Case

The calculation of valid longitudinal trajectories is of significant importance for safety and comfort. Thus velocity profile is determined by solving an Optimal Control Problem (OCP), considering the traffic situation as well as road infrastructure. Such an OCP is a special case of the general optimization problem 1.1. For that purpose the

longitudinal OCP is formulated using the kinematic relations:

$$\min_{u} \quad J_s = k_{t_f} t_f + \int_0^{t_f} \frac{\alpha_s}{2} u_s^2(t) + \frac{\gamma_s}{2} s_3^2(t) \mathrm{d}t \tag{1.14a}$$

$$\text{s.t.} \quad \dot{\boldsymbol{s}}(t) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \boldsymbol{s}(t) + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u_s(t), \tag{1.14b}$$

$$\boldsymbol{s}(0) = \boldsymbol{s}_0 \qquad \text{and} \qquad \boldsymbol{s}(t_f) = \boldsymbol{s}_{t_f}, \tag{1.14c}$$

where the state $s \in \mathbb{R}^3$ represents the arc-length and its first (velocity) and second (acceleration) derivatives of the ego-vehicle. The control input $u_s(t)$ represents the jerk and t_f the final time, which gives the optimization horizon. The three terms of the cost functional J_s contain the jerk, the acceleration as well as the final time. These terms are weighted with the parameters k_{tf}, α_s, $\gamma_s \in \mathbb{R}^+$. The resulting OCP is solved using the Hamiltonian

$$\mathcal{H}\left(t, s^\star, u_s^\star, \lambda\right) = \lambda_1 s_2 + \lambda_2 s_3 + \lambda_3 u_s + \frac{\alpha_s}{2} u_s^2 + \frac{\gamma_s}{2} s_3^2 \tag{1.15}$$

with the Langrange multipliers $\lambda = [\lambda_1, \lambda_2, \lambda_3]$. Following the ideas of the calculus of variations the analytical solution of the optimization problem can be computed as

$$s_1(t) = \sigma_0 + \sigma_1 t + \sigma_2 t^2 + \sigma_3 t^3 + \sigma_4 e^{k_s t} + \sigma_5 e^{-k_s t}, \tag{1.16}$$

where $K_s = \sqrt{\gamma_s / \alpha_s}$. The coefficients $\sigma_0 \dots \sigma_5$ will now be determined using the initial and final conditions

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & k_s & -k_s \\ 0 & 0 & 2 & 0 & k_s^2 & k_s^2 \\ 1 & t_{f,j}^i & t_{f,j}^{i^2} & t_{f,j}^{i^3} & e^{k_s t_{f,j}^i} & e^{-k_s t_{f,j}^i} \\ 0 & 1 & 2t_{f,j}^i & 3t_{f,j}^{i^2} & k_s e^{k_s t_{f,j}^i} & -k_s e^{-k_s t_{f,j}^i} \\ 0 & 0 & 2 & 6t_{f,j}^i & k_s^2 e^{k_s t_{f,j}^i} & k_s^2 e^{-k_s t_{f,j}^i} \end{pmatrix} \begin{pmatrix} \sigma_0^i \\ \sigma_1^i \\ \sigma_2^i \\ \sigma_3^i \\ \sigma_4^i \\ \sigma_5^i \end{pmatrix}$$

$$= \begin{pmatrix} s_0 & \dot{s}_0 & \ddot{s}_0 & s^i & \dot{s}^i & \ddot{s}^i \end{pmatrix}^T. \tag{1.17}$$

While the initial conditions are determined based on the actual position, velocity, and acceleration or based on the previous trajectory, are the final conditions not a-priory clear. Since no constraints are considered in the OCP (1.14a)-(1.14c), theoretically all values can arbitrary increase. To tackle that issue are the final conditions varied and the resulting trajectories filtered due to constraints and sorted based on the cost functional. Beside the indicated degree of freedom, contains the reference for the velocity. Such that the final value for the velocity is clear, but when we will reach it is fittable. For more details see [101].

As depicted in (1.14a), there is no velocity term in the optimization problem, accordingly, it becomes tricky to track the requested velocity at every time instant. One of the solution strategies, to track the requested velocity profile is by artificially partitioning the incoming velocity profile [101].

1.5.4 The Lateral Optimal Control Problem

As in the longitudinal case, a special form of (1.1) and the kinematic relations of the lateral behavior are used to formulate an OCP:

$$\min_{u} \quad J_d = k_{t_f} t_f + \int_0^{t_f} \frac{\alpha_d}{2} u_d^2(t) + \frac{\gamma_d}{2} d_2^2(t) \mathrm{d}t \tag{1.18a}$$

$$\text{s.t.} \quad \dot{\boldsymbol{d}}(s(t)) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \boldsymbol{d}(s(t)) + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u_d(t), \tag{1.18b}$$

$$\boldsymbol{d}(0) = \boldsymbol{d}_0 \qquad \text{and} \qquad \boldsymbol{d}(s(t_f)) = \boldsymbol{d}_{s(t_f)}. \tag{1.18c}$$

As above can the analytical solution

$$\begin{aligned} d_1(s(t)) =& \delta_0 + \delta_1 s + e^{k_d s} \left(\delta_2 \cos\left(k_d s\right) + \delta_3 \sin\left(k_d s\right) \right) \\ &+ e^{-k_d s} \left(\delta_4 \cos\left(k_d s\right) + \delta_5 \sin\left(k_d s\right) \right), \end{aligned} \tag{1.19}$$

where $k_d = \sqrt[4]{\frac{\gamma_d}{4\alpha_d}}$, be achieved using Pontryagin's Minimum Principle. Following the same argumentation as in longitudinal calculations, a linear set of equations is formulated and solved several times. The control references are the path information introduced in section 1.5.1. As a result we get a bunch of lateral trajectories of the form (1.19) specified via different values for $\delta_0, \ldots, \delta_5$. With these two sets of trajectories, one for lateral and one for longitudinal, in hand all combinations are determined. These combinations are trajectories in the plane and can be used as input for the underlying following control. Now the question arises which trajectory should be chosen? For that purpose the resulting 2D Trajectories are filtered due to inherent requirements like physical or comfort limits, e.g. maximum of acceleration or the path boundaries. As a result, a list of possible drivable trajectories is achieved. This list is sorted concerning the overall cost functional

$$J(s(t), d(s)) = k_s J_s(s(t)) + k_d J_d(d(s)), \tag{1.20}$$

containing the lateral and longitudinal ones. The best trajectory is transferred to control.

At first glance, it looks a little bit intricate to compute numerous trajectories and filter them afterward. Nevertheless, it is computational very efficient to solve the underlying linear set of equations using the analytical solution. Such that no online optimization and thus no extensive online optimization solver is needed. An alternative approach is given in the next section.

1.5.5 MPC Based Approach

In this section, we will show how the path following is realized by utilizing an MPC based approach. Here, we consider explicitly a vehicle dynamic and kinematic model. Consequently, the resulting trajectories are more preferable, especial in challenging urban situations (heavy traffic and narrow roads). However, the MPC based approach

is less computational efficient in comparison to the model-free previously discussed approach.

In the case of the MPC, optimization is a central tool to achieve comfortable and even derivable trajectories again. Thus we will come back to the basic definitions of section 1.2. Within the scope of AD, steering a vehicle autonomously along a pre-specified geometric reference path is an important control task. With this reference path in hand, we could directly control the vehicle using the path information as input. Nevertheless, this might lead to uncomfortable driving behavior as well as controller inconsistency. To smooth the path and directly considering comfort requirements optimization techniques are used again. In contrast to the previous section, here we address a MPC approach which directly takes the vehicle model into account.

As mentioned before, a vehicle model is needed in order to set up the MPC. Accordingly, the longitudinal and lateral dynamic of the vehicle is modeled as a continuous-time nonlinear system in the form

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)), \quad \boldsymbol{x}(t_0) = \boldsymbol{x}_0 \tag{1.21a}$$

$$\boldsymbol{y}(t) = \boldsymbol{h}(\boldsymbol{x}(t)) \tag{1.21b}$$

with the maps $\boldsymbol{f} : \mathbb{R}^9 \times \mathbb{R}^2 \to \mathbb{R}^9$ and $\boldsymbol{h} : \mathbb{R}^9 \to \mathbb{R}^3$. The state vector $\boldsymbol{x} = [x,\, y,\, \psi,\, \dot{\psi},\, \beta,\, v,\, v_{ref},\, \delta_s,\, \delta_{s,ref}]^\top$ consists of the vehicle states, like the coordinates of the center of gravity in an inertial frame x and y, the yaw angle ψ, the yaw rate $\dot{\psi}$, the side slip angle β, the velocity v, the target velocity v_{ref}, the steering wheel angle δ_s and the steering wheel target angle $\delta_{s,ref}$. The control input $\boldsymbol{u} = [a_{ref},\, \omega_{s,ref}]^\top$ contains the target acceleration a_{ref} and the steering wheel target angular velocity $\omega_{s,ref}$ of the vehicle. Furthermore, the output vector $\boldsymbol{y} = [x_f,\, y_f,\, \psi]^\top$ includes the coordinates of the middle of the front axle x_f and y_f as well as the vehicle orientation ψ.

More specifically, we are using a single track model according to [27] in order to describe the lateral vehicle behaviour. By assuming a steerable front wheel, we receive the following model equation set

$$\dot{x}(t) = v(t)\cos(\psi(t) + \beta(t)) \tag{1.22a}$$

$$\dot{y}(t) = v(t)\sin(\psi(t) + \beta(t)) \tag{1.22b}$$

$$\ddot{\psi}(t) = -\frac{c_{sf}\, l_f^2 + c_{sr}\, l_r^2}{J_z\, v(t)}\dot{\psi}(t) - \frac{c_{sf}\, l_f - c_{sr}\, l_r}{J_z}\beta(t) + \frac{c_{sf}\, l_f}{J_z}\delta_a(t) \tag{1.22c}$$

$$\dot{\beta}(t) = \left(-1 - \frac{c_{sf}\, l_f - c_{sr}\, l_r}{m\, v(t)^2}\right)\dot{\psi}(t) - \frac{c_{sf} + c_{sr}}{m\, v(t)}\beta(t) + \frac{c_{sf}}{m\, v(t)}\delta_a(t). \tag{1.22d}$$

The parameters l_f and l_r denoting the distance from CG to the front and rear wheel respectively. Furthermore, the parameters c_{sf} and c_{sr} are the cornering stiffnesses of the front and rear wheel, respectively. The total mass and the yaw moment of inertia of the vehicle are denoted by m and J_z.

The lateral vehicle dynamic is represented in form of a first-order plus integrator model

$$\dot{v}(t) = \frac{1}{T_v}(v_{ref}(t) - v(t)) \tag{1.23a}$$

$$\dot{v}_{ref}(t) = a_{ref}(t) \tag{1.23b}$$

where a_{ref} is the target acceleration and T_v is the time constant. This model represents the dynamics of the engine and braking system, as well as the utilized low-level controller of the vehicle. For more details of the utilized dynamic single-track model, please refer to [98].

To finalize the model description, we also have to consider the sets of state and input constraints, which results due to the physical limitations of the vehicle. These sets of constraints are given by $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^9$ and $\boldsymbol{u} \in \mathcal{U} \subseteq \mathbb{R}^2$.

The primary goal of the MPC vehicle guidance approach is to create an output sequence of the system (1.21), which realizes a proper tracking of the geometric reference path. To realize a path tracking, we have to define the tracking error. Therefore, we are utilizing the deviation from the path as

$$\boldsymbol{e}(t) := \boldsymbol{h}(\boldsymbol{x}(t)) - \boldsymbol{p}(s(t)). \tag{1.24}$$

Accordingly, the path-following problem can be formulated as follows. For the given system (1.21) and the reference path, we want to design a controller that computes an optimal $\boldsymbol{u}(\cdot)$ as well as $s(\cdot)$ and guarantees:

1. Path convergence: The system output $\boldsymbol{y}$ converges to the set $\mathcal{P}$ such that $\lim_{t\to\infty} \|\boldsymbol{e}(t)\| = 0$.
2. Velocity convergence: The path velocity $\dot{s}(t)$ converges to a predefined evolution $\dot{s}_{ref}(t) \geq 0$ such that $\lim_{t\to\infty} \|\dot{s}(t) - \dot{s}_{ref}(t)\| = 0$.
3. Constraint satisfaction: The state and input constraints $\mathcal{X}$ and $\mathcal{U}$ are satisfied $\forall t \in [t_0, \infty)$.

A common procedure for the definition of a path-following MPC is the usage of a path parameter s in form of a virtual state. In the current case, the time evolution of these virtual state is described by a differential equation, named timing law. We defined timing law as a single integrator

$$\dot{s}(t) := \vartheta(t), \tag{1.25}$$

where $\vartheta \in \mathcal{V} \subset \mathbb{R}$ is an additional (virtual) control input of the MPC.

We are formulating the path-following problem in form of a standard optimization problem, as initially sketched in equation 1.1 and concertized in equation 1.14a. In order to solve these problem, a sampled-data MPC approach is used, similar to [102, 103, 104]. This means, we obtain the system input via the repetitive solution of an OCP at every discrete sampling time instance $t_k = k\,T_s$ in a receding horizon fashion. In the following, the predicted system states and inputs are denoted by $\bar{\boldsymbol{x}}(\cdot)$

and $\bar{\boldsymbol{u}}(\cdot)$. The cost functional to be minimized for the prediction horizon T_p is given by

$$J(\boldsymbol{x}(t_k),\, s(t_k), \bar{\boldsymbol{u}}(\cdot), \bar{\vartheta}(\cdot)) = \tag{1.26}$$
$$\int_{t_k}^{t_k+T_p} \left\| \begin{matrix} \bar{\boldsymbol{e}} \\ a_{lat}(\bar{\boldsymbol{x}}) \end{matrix} \right\|_Q^2 + \left\| \begin{matrix} \bar{\boldsymbol{u}} \\ \bar{\vartheta} - \vartheta_{ref} \end{matrix} \right\|_R^2 \mathrm{d}\tau + \left\| \begin{matrix} \bar{\boldsymbol{e}} \\ a_{lat}(\bar{\boldsymbol{x}}) \end{matrix} \right\|_P^2$$

where $Q = \mathrm{diag}(q_x, q_y, q_\psi, q_a)$, $P = \mathrm{diag}(p_x, p_y, p_\psi, p_a)$ and $R = \mathrm{diag}(r_a, r_\omega, r_\vartheta)$ are positive definite weighting matrices. Furthermore, $a_{lat}(\boldsymbol{x}) = v(t)\,\dot{\psi}(t)$ is an approximation of the lateral acceleration of the vehicle. The resulting OCP solved at every sampling time instance is

$$\min_{\bar{\boldsymbol{u}}(\cdot),\bar{\vartheta}(\cdot)} J(\boldsymbol{x}(t_k), s(t_k), \bar{\boldsymbol{u}}(\cdot), \bar{\vartheta}(\cdot)) \tag{1.27a}$$

subject to the constraints

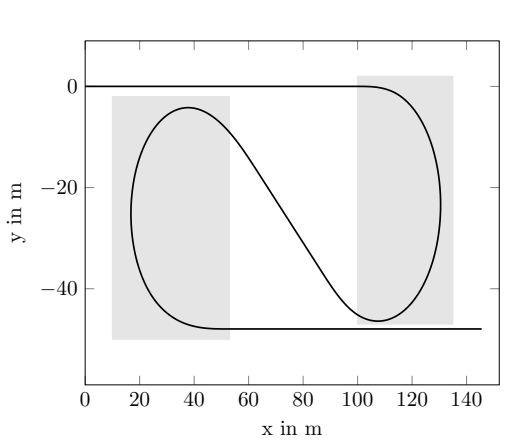
$$\dot{\bar{\boldsymbol{x}}}(\tau) = \boldsymbol{f}(\bar{\boldsymbol{x}}(\tau), \bar{\boldsymbol{u}}(\tau)), \quad \bar{\boldsymbol{x}}(t_k) = \boldsymbol{x}(t_k) \tag{1.27b}$$
$$\dot{\bar{s}}(\tau) = \bar{\vartheta}(\tau), \quad \bar{s}(t_k) = s(t_k) \tag{1.27c}$$
$$\bar{\boldsymbol{e}}(\tau) = \boldsymbol{h}(\bar{\boldsymbol{x}}(\tau)) - \boldsymbol{p}(\bar{s}(\tau)) \tag{1.27d}$$
$$\bar{\boldsymbol{u}}(\tau) \in \mathcal{U}, \quad \bar{\boldsymbol{x}}(\tau) \in \mathcal{X} \tag{1.27e}$$
$$\bar{s}(\tau) \in [0, s_{max}], \quad \bar{\vartheta}(\tau) \in \mathcal{V} \tag{1.27f}$$
$$\boldsymbol{h}_c(\bar{\boldsymbol{x}}(\tau), \bar{\boldsymbol{u}}(\tau)) \leq \boldsymbol{0} \tag{1.27g}$$

which have to hold for all $\tau \in [t_k, t_k + T_p]$. The constraints (1.27b) and (1.27c) are representing the dynamics of the system (1.21) and the time law (1.25) with their respective initial conditions. Due to the fact that we want to realize a proper path-following behaviour, we try to minimize the deviation of the system output from the path. Therefore, the path deviation (1.24) is stated by (1.27d). Equations (1.27e) and (1.27f) representing the state and input constraints, including the virtual state $\bar{s}$ and the virtual input $\bar{\vartheta}$. Finally, (1.27g) defines further constraints with the constraint function $\boldsymbol{h}_c$. The optimal input trajectory $\bar{\boldsymbol{u}}_k^\star(\cdot)$ which results from the OCP is applied to the system (1.21)

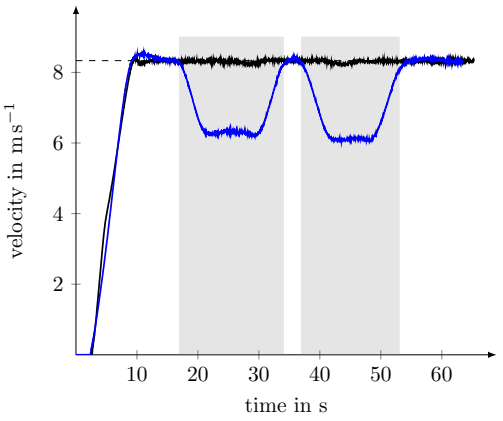
$$\forall t \in [t_k, t_k + T_s) : \boldsymbol{u}(t) = \bar{\boldsymbol{u}}_k^\star(t). \tag{1.28}$$

More in detail, the input trajectory $\bar{\boldsymbol{u}}_k^\star(\cdot)$ is transmitted to the follow-up controller. The follow-up controller is realized as a classical PID based cascade controller, which handles the longitudinal and lateral tracking control problem independently of each other. The main objective of the follow-up controller is the transformation of the trajectory $\bar{\boldsymbol{u}}_k^\star(\cdot)$ into a suitable control variable for the vehicle interface (vehicle CAN) as well as the elimination of disturbance effects.

At each new time instance $t_k + T_s$, the optimization horizon is shifted forward and the OCP from equation (1.27) is solved again for new initial conditions and the whole process is repeated.



(a) Reference path



(b) Achieved velocity for the kinematic trajectory planning approach (—) as well as the MPC(—)

Figure 1.4: Test drive data of the trajectory planning approach and MPC

After introducing the two approaches for the calculation of suitable driving trajectories, the question arises what is better or which one should be used in which situation? While this question is generally hard to answer, we illustrate how the algorithms act on the following use case.

To briefly illustrate the differences between the two approaches introduced in the Sections 1.5.2–1.5.4 and 1.5.5 a small case study is introduced. Thus, real test data are presented and the influences of the two different planning strategies (MPC and trajectory planning approach) of the driven velocity is considered. For that purpose we drove the path shown in Figure 1.4a and a velocity of $v = 30\,\mathrm{km\,h^{-1}} \approx 8.3\,\mathrm{m\,s^{-1}}$. For sure are the lateral and the longitudinal problems are solved, but the achieved differences on the lateral behavior are not so descriptive. Such that the longitudinal tracking is investigated more in detail. Figure 1.4b shows that the velocity tracking of the trajectory planning approach is quite well. There is just some noise on the velocity, which is a result of the measurement disturbances coming from the ground as well as the environment. At first glance we see that the velocity tracking of the MPC is much worse. There are two sections (gray shaded) where the velocity significantly decreased, while the set value remains fix. Certainly this is a feature and no bug. Based on the gray shaded areas in Figure 1.4 we see that the break-in of the velocity occurs in the curves of the path. In the cost functional 1.26 is the lateral acceleration (a_{lat}) weighted and thus influences the the request velocity of the MPC. Driving in the curves lead to an increasing of the lateral acceleration. In order to limit this increasing is the velocity decreased. In other words, the minimum of the cost functional 1.26 is affected by the velocity error as well as the lateral acceleration. Thus a compromise results.

To tackle the previously teased question let us sum up some properties of the two approaches. The MPC has the advantage that it takes the dynamic model of the vehicle into account. In a nutshell, this leads to a more comfortable and eas-

ier trackable trajectory. However, online optimization is needed, which results in a high computational load and might lead to suboptimal or non-valid solutions. The trajectory planning approach is undemanding in the needed resources since the optimization problem is solved analytical and transformed into solving a linear set of equations. Although we need to calculate a bunch of trajectories. Due to the considered kinematic model, all dynamic effects need to be addressed in the underlying controller.

Thus the concrete choice for one of the algorithms significantly depends on the hardware available, the ODD, the DDT as well as the requirements regarding comfort and tracking quality.

1.6 CONCLUSION

Futuristic and autonomous vehicles are currently a trendy topic with great attention from both academia and industry. We were presenting a survey, which offers an overview of automated driving architectures and functionalities. As an introduction to the topic, we started with a detailed analysis of one concrete automated driving task (lane change). The task of realizing a lane change serves as a running example. Following these, we were showing that mathematical optimization is a key technology to realize automated driving functionalities. Therefore, we were formulating a generalized optimization problem and discussing, which automated driving functions are using such optimizing methods. Following these, we were discussing concrete functional architectures for automated driving applications, realized by different companies and universities.

After these discussions, we were going more in detail and focusing on environment representation, situation-aware interpretation, and (rule-based and AI-based) intelligent decision-making sub-functions. Here, we were starting with the formal problem statements and discuss realization aspects. Finally, we were presenting one model-free and one model-based motion planning approach. Also here, we are starting with the formal problem statements and discuss thereafter concrete realization aspects. All these discussed automated driving sub-functions were already implemented and tested in IAVs test vehicles in open traffic situations in Germany.

In our future work, we will gradually realize and expand our situation-aware maneuver planning approach by adopting a two-pronged strategy. On the one hand, from the short-term application and project perspective, respectively, we will focus on more straightforward decision-making approaches (e.g., in terms of hierarchical state-machines), but extend the number of available basic maneuvers and improve the existing maneuvers and driving behaviors to be applicable in more complex situations. For our shuttle project ABSOLUT, e.g., the lane change maneuver has to be revised to be applicable for both highway and more complex urban scenarios. On the other hand, while extending the number of maneuvers and driving behaviors with each project, we will step-by-step replace the straightforward decision-making and planning approaches by more intelligent approaches that, among others, can cope with uncertainties, allow for online planning on different levels of abstraction, and can even adapt to new and unforeseen situations.

The urban scenarios considered in the project OPA^{3}L light this progress in decision-making and planning techniques. Furthermore is the MPC a significant topic of the sequel development. Such that an intelligent velocity preprocessing and path adjustments to realize an obstacle avoidance functionality within the MPC, as well as an object-based control, are essential parts of our actual and future work.

Acronyms

ACC Adaptive Cruise Control. xxiv

AD Automated Driving. xii, xv–xviii, xxi, xxii, xxiv, xxvii, xxxiii

AD-System Automated Driving System. xii, xiii, xv, xvi, xviii–xx, xxiv

AI Artificial Intelligence. xii, xxv, xxxvii

AL Active Learning. xxi, xxii

AS Active Seeing. xviii, xxi

CEM Comprehensive Environment Model. xix, xx

DDT Dynamic Driving Task. xii, xxiv, xxxvii

GLOSA Green Light Optimal Speed Advisory. xxviii

IG Information Gain. xxi, xxii

KL Kullback-Leibler. xxi, xxii

MDP Markov Decision Process. xxv–xxvii

MHE Moving Horizon Estimation. xvii

ML Machine Learning. xvi, xxv

MPC Model Predictive Control. xvi, xxx, xxxii–xxxiv, xxxvi, xxxviii

OCP Optimal Control Problem. xxx–xxxii, xxxiv, xxxv

ODD Operational Design Domain. xii, xiii, xxiv, xxxvii

POMDP Partially Observable Markov Decision Process. xxvii

PREVA Predictive Velocity Adaption. xxviii

RL Reinforcement Learning. xvi, xxv, xxvii

SA Situation Awareness. xix

SMDP Semi-Markov Decision Process. xxvii

V2X Vehicle-to-Everything. xxii, xxiii

Bibliography

[1] Liu, Zongwei, Jiang, Hao, Tan, Hong, and Zhao, Fuquan. An overview of the latest progress and core challenge of autonomous vehicle technologies. *MATEC Web Conf.*, 308:06002, 2020.

[2] SAE International. *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*, June 2018.

[3] Alexander Jungmann, Christian Lang, Florian Pinsker, Roland Kallweit, Mirko Taubenreuther, and Matthias Butenuth. Artificial intelligence for automated driving – quo vadis? In *Automatisiertes Fahren 2019*, pages 117–134. Springer Fachmedien Wiesbaden, 2020.

[4] SYNCAR - Synchronized Automated Driving in Urban Areas. https://www.synchrone-mobilitaet.de/en/projects/syncar.html, 2020. Accessed: 2020-08-04.

[5] HarmonizeDD - Continuous Support of Connected and Automated Driving for Mixed Traffic With Heterogeneously Equipped Vehicles. https://www.synchrone-mobilitaet.de/en/projects/harmonizedd.html, 2020. Accessed: 2020-08-04.

[6] Rico Auerswald, Roman Busse, Markus Dod, Richard Fritzsche, Alexander Jungmann, Michael Klöppel-Gersdorf, Josef F. Krems, Sven Lorenz, Franziska Schmalfuß, Sabine Springer, and Severin Strobl. Cooperative driving in mixed traffic with heterogeneous communicationsand cloud infrastructure. In *Proceedings of the 5th International Conference on Vehicle Technology and Intelligent Transport Systems*, pages 95–105, 2019.

[7] Automation ohne Unsicherheit zur Erhöhung der Akzeptanz automatisierten und vernetzten Fahrens – AutoAkzept. https://www.bmvi.de/SharedDocs/DE/Artikel/DG/AVF-projekte/autoakzept.html, 2020. Accessed: 2020-08-17.

[8] Uwe Drewitz, Klas Ihme, Michael Oehl, Frank Schrödel, Rick Voßwinkel, Franziska Hartwich, Cornelia Hollander, Anna-Antonia Pape, Tobias Fleischer, Sonja Cornelsen, Andreas Lüdtke, Daniela Gräfing, and Alexander Trende. Automation ohne Unsicherheit: Vorstellung des Förderprojekts AutoAkzept zur Erhöhung der Akzeptanz automatisierten Fahrens. In *10. VDI Fachtagung Mensch-Maschine-Mobilität*, pages 1–19, November 2019.

[9] OPA^{3}L - Optimal Assistierte, hoch Automatisierte, Autonome und kooperative Fahrzeugnavigation und Lokalisation. http://www.math.uni-bremen.de/zetem/cms/detail.php?id=19184, 2020. Accessed: 2020-08-17.

[10] ABSOLUT - Anspruchsvoll, Ganzheitlich, Intelligent. https://absolut-projekt.de/, 2020. Accessed: 2020-08-04.

[11] HEAT - Hamburg Electric Autonomous Transportation. https://bit.ly/2BXUUkR, 2020. Accessed: 2020-08-04.

[12] Elijah Polak. *Optimization: Algorithms and Consistent Approximations*. Springer, 1997.

[13] Leonid T. Aschepkov, Dmitriy V. Dolgy, Taekyun Kim, and Ravi P. Agarwal. *Optimization: Algorithms and Consistent Approximations*. Springer International Publishing, 2016.

[14] Pierre de Fermat. *Pierre de Fermats Abhandlungen über Maxima und Minima (1629)*. Akademische Verlagsgesellschaft m. b. h. Leipzig, 1934.

[15] John Wallis. A treatise of algebra, both historical and practical. *Philosophical Transactions of the Royal Society of London*, 15(173):1095–1106, 1685.

[16] Ding-Zhu Du, Panos M. Pardalos, and Weili Wu. History of optimization. In *Encyclopedia of Optimization*, pages 1538–1542, Boston, 2009. Springer US.

[17] Svetlana S. Petrova and Alexander D. Solov'ev. The origin of the method of steepest descent. *Historia Mathematica*, 24(4):361 – 375, 1997.

[18] George B. Dantzig and Mukund N. Thapa. *Linear Programming 1: Introduction*. Springer, New York, 1997.

[19] Charles G. Broyden. Quasi-newton methods. In *Numerical Methods for Unconstrained Optimization*, pages 87–106, London, 1972. Academic Press.

[20] John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, Cambridge, 1992.

[21] Scott Kirkpatrick, Danial Gelatt, and Mario P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[22] James Kennedy, Russel C. Eberhart, and Yuhui Shi. *Swarm Intelligence*. Morgan Kaufmann / Academic Press, San Francisco/ San Diego, 2001.

[23] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, 2006.

[24] Kevin P Murphy. *Machine Learning: A Probabilistic Perspective*. MIT press, Cambridge, 2012.

[25] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, 2016.

[26] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, 1998.

[27] Rajesh Rajamani. *Vehicle Dynamics and Control*. Springer, New York, 2012.

[28] Ali Boyali, Seiichi Mita, and Vijay John. A Tutorial On Autonomous Vehicle Steering Controller Design, Simulation and Implementation. *ArXiv*, abs/1803.03758, 2018.

[29] Pan Zhao, Jiajia Chen, Yan Song, Xiang Tao, Tiejuan Xu, and Tao Mei. Design of a Control System for an Autonomous Vehicle Based on Adaptive-PID. *International Journal of Advanced Robotic Systems*, 9(2):44, 2012.

[30] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated planning: theory & practice*. Morgan Kaufmann, San Francisco, 2004.

[31] Wilko Schwarting, Javier Alonso-Mora, and Daniela Rus. Planning and decision-making for autonomous vehicles. *Annual Review of Control, Robotics, and Autonomous Systems*, 2018.

[32] Christopher V. Rao, James B. Rawlings, and Jay H. Lee. Constrained linear state estimation—a moving horizon approach. *Automatica*, 37(10):1619 – 1628, 2001.

[33] Afnan Alofi, Anwaar A. Alghamdi, Razan Faisal Alahmadi, Najla Aljuaid, and M Hemalatha. A review of data fusion techniques. *International Journal of Computer Applications*, 167:37–41, 2017.

[34] Ronald C. Arkin. *Behavior-Based Robotics*. MIT press, Cambridge, 1998.

[35] Mohamed Salah Hamdi. *Entwurf adaptiver lernender Roboter*. PhD thesis, Universität Hamburg, 1999.

[36] Robin R. Murphy. *Introduction to AI robotics*. MIT press, Cambridge, 2019.

[37] James S. Albus, Harry McCain, and Ronald Lumia. NASA/NBS standard reference model for telerobot control system architecture (NASREM). Technical report, 1989.

[38] Rodney Brooks. Achieving artificial intelligence through building robots. Technical report, 1986.

[39] Rodney Brooks. Intelligence without representation. *Artificial intelligence*, 47(1-3):139–159, 1991.

[40] Tucker Balch and Ronald C. Arkin. Behavior-based formation control for multirobot teams. *IEEE Transactions on Robotics and Automation*, 14(6):926–939, 1998.

[41] Rodney Brooks. A robust layered control system for a mobile robot. *IEEE Journal on Robotics and Automation*, 2(1):14–23, 1986.

[42] James S. Albus. The NIST real-time control system (RCS): an approach to intelligent systems research. *Journal of Experimental & Theoretical Artificial Intelligence*, 9(2-3):157–174, 1997.

[43] Malik Ghallab, Dana Nau, and Paolo Traverso. The actors view of automated planning and acting: A position paper. *Artificial Intelligence*, 208:1–17, 2014.

[44] Felix Lotz. *Eine Referenzarchitektur für die assistierte und automatisierte Fahrzeugführung mit Fahrereinbindung (English Title: A Reference Architecture for Assisted and Automated Vehicle Guidance with Driver Interaction)*. PhD thesis, Technische Universität Darmstadt, 2017.

[45] Félix Ingrand and Malik Ghallab. Deliberation for autonomous robots: A survey. *Artificial Intelligence*, 247:10–44, 2017.

[46] Erann Gat. On three-layer architectures. *Artificial intelligence and mobile robots*, 195:210, 1998.

[47] Rachid Alami, Raja Chatila, Sara Fleury, Malik Ghallab, and Félix Ingrand. An architecture for autonomy. *The International Journal of Robotics Research*, 17(4):315–337, 1998.

[48] Ernst Dieter Dickmanns, Reinhold Behringer, Dirk Dickmanns, Thomas Hildebrandt, Markus Maurer, Frank Thomanek, and Joachim Schiehlen. The seeing passenger car'vamors-p'. In *Proceedings of the Intelligent Vehicles' 94 Symposium*, pages 68–73. IEEE, 1994.

[49] Ernst Dieter Dickmanns. *Dynamic vision for perception and control of motion*. Springer Science & Business Media, 2007.

[50] Markus Maurer. *Flexible Automatisierung von StraBenfahrzeugen mit Rechnersehen*. PhD thesis, Universität der Bundeswehr München, 2000.

[51] Martin Pellkofer. *Verhaltensentscheidung für autonome Fahrzeuge mit Blickrichtungssteuerung*. PhD thesis, Universität der Bundeswehr Münche, 2003.

[52] Sören Kammel, Julius Ziegler, Benjamin Pitzer, Moritz Werling, Tobias Gindele, Daniel Jagzent, Joachim Schröder, Michael Thuy, Matthias Goebl, Felix von Hundelshausen, et al. Team AnnieWAY's autonomous system for the 2007 DARPA Urban Challenge. *Journal of Field Robotics*, 25(9):615–639, 2008.

[53] Charles Reinholtz, Dennis Hong, Al Wicks, Andrew Bacha, Cheryl Bauman, Ruel Faruque, Michael Fleming, Chris Terwelp, Thomas Alberi, David Anderson, et al. Odin: Team VictorTango's entry in the DARPA urban challenge. In *The DARPA Urban Challenge*, pages 125–162. Springer, Berlin, Heidelberg, 2009.

[54] Michael Montemerlo, Jan Becker, Suhrid Bhat, Hendrik Dahlkamp, Dmitri Dolgov, Scott Ettinger, Dirk Haehnel, Tim Hilden, Gabe Hoffmann, Burkhard Huhnke, et al. Junior: The stanford entry in the urban challenge. *Journal of field Robotics*, 25(9):569–597, 2008.

[55] Chris Urmson, Joshua Anhalt, Drew Bagnell, Christopher Baker, Robert Bittner, MN Clark, John Dolan, Dave Duggins, Tugrul Galatali, Chris Geyer, et al. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466, 2008.

[56] Tobias Nothdurft, Peter Hecker, Sebastian Ohl, Falko Saust, Markus Maurer, Andreas Reschka, and Jürgen Rüdiger Böhmer. Stadtpilot: First fully autonomous test drives in urban traffic. In *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 919–924. IEEE, 2011.

[57] Eric Bauer, Felix Lotz, Matthias Pfromm, Matthias Schreier, Stephan Cieler, Alfred Eckert, Andree Hohm, Stefan Lüke, Peter Rieth, Bettina Abendroth, Volker Willert, Jürgen Adamy, Ralph Bruder, Ulrich Konigorski, and Hermann Winner. Proreta 3: An integrated approach to collision avoidance and vehicle automation. *at-Automatisierungstechnik*, 60(12):755–765, 2012.

[58] Tobias Nothdurft. *Ein Kontextmodell für sicherheitsrelevante Anwendungen in der autonomen Fahrzeugführung*. PhD thesis, Technische Universität Braunschweig, 2014.

[59] Richard Matthaei and Markus Maurer. Autonomous driving–a top-down-approach. *at-Automatisierungstechnik*, 63(3):155–167, 2015.

[60] Frank Schrödel and Matthias Freese. Concept and Validation of a Guidance Approach for Highly Automated Shuttles. *IFAC-PapersOnLine*, 52(5):359–365, 2019.

[61] Mica Endsley. Toward a theory of situation awareness in dynamic systems. *Human Factors*, 37(1):32–64, 1995.

[62] Mica Endsley. Situation Awareness and Human Error : Designing to Support Human Performance. In *Proceedings of the High Consequence Systems Surety Conference*, 1999.

[63] Thanh Nguyen, Chee Peng Lim, Ngoc Duy Nguyen, Lee Gordon-Brown, and Saeid Nahavandi. A Review of Situation Awareness Assessment Approaches in Aviation Environments. *IEEE Systems Journal*, 13(3):3590–3603, 2019.

[64] Matthias Schreier. Environment representations for automated on-road vehicles. *at-Automatisierungstechnik*, 66(2):107–118, 2018.

[65] Stéphanie Lefèvre, Dizan Vasquez, and Christian Laugier. A survey on motion prediction and risk assessment for intelligent vehicles. *ROBOMECH journal*, 1(1):1, 2014.

[66] Britta Hummel. *Description Logic for Scene Understanding: at the Example of Urban Road Intersections*. PhD thesis, Universität Karlsruhe, 2010.

[67] Michael Huelsen. *Knowledge-based driver assistance systems: traffic situation description and situation feature relevance*. Springer Vieweg, Wiesbaden, 2014.

[68] Stefan Brunner, Markus Kucera, and Thomas Waas. Ontologies used in robotics: A survey with an outlook for automated driving. In *IEEE International Conference on Vehicular Electronics and Safety*, pages 81–84. IEEE, 2017.

[69] Christian Roesener, Felix Fahrenkrog, Axel Uhlig, and Lutz Eckstein. A scenario-based assessment approach for automated driving by using time series classification of human-driving behaviour. In *IEEE 19th international conference on intelligent transportation systems*, pages 1360–1365. IEEE, 2016.

[70] ByeoungDo Kim, Chang Mook Kang, Jaekyum Kim, Seung Hi Lee, Chung Choo Chung, and Jun Won Choi. Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network. In *IEEE 20th International Conference on Intelligent Transportation Systems*, pages 399–404. IEEE, 2017.

[71] Florent Altché and Arnaud de La Fortelle. An lstm network for highway trajectory prediction. In *IEEE 20th International Conference on Intelligent Transportation Systems*, pages 353–359. IEEE, 2017.

[72] Nachiket Deo and Mohan M. Trivedi. Convolutional social pooling for vehicle trajectory prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1468–1476, 2018.

[73] Xin Li, Xiaowen Ying, and Mooi Choo Chuah. Grip++: Graph-based interaction-aware trajectory prediction. *IEEE Intelligent Transportation Systems Conference*, pages 3960–3966, 2019.

[74] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press, Cambridge, 2005.

[75] Achim Klenke. *Wahrscheinlichkeitstheorie*. Springer, Berlin Heidelberg, 2006.

[76] Yiannis Demiris and Bassam Khadhouri. Hierarchical attentive multiple models for execution and recognition of actions. *Robotics and autonomous systems*, 54(5):361–369, 2006.

[77] Mehdi Elahi, Francesco Ricci, and Neil Rubens. A survey of active learning in collaborative filtering recommender systems. *Computer Science Review*, 20:29–50, 2016.

[78] Alexander Jungmann and Bernd Kleinjohann. Automatic Composition of Service-based Image Processing Applications. In *Proceedings of the 13th IEEE International Conference on Services Computing*, pages 106–113, 2016.

[79] Alexander Jungmann and Bernd Kleinjohann. A Holistic and Adaptive Approach for Automated Prototyping of Image Processing Functionality. In *Proceedings of the 21st IEEE International Conference on Emerging Technologies and Factory Automation*, pages 1–8, 2016.

[80] Martin L. Puterman. *Markov decision processes: Discrete stochastic dynamic programming*. Wiley-Interscience, Hoboken, 2005.

[81] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1):99–134, 1998.

[82] Alexander Jungmann, Bernd Kleinjohann, and Willi Richert. A Fast Hierarchical Learning Approach for Autonomous Robots. In *Organic Computing - A Paradigm Shift for Complex Systems*, pages 545–558. Springer, 2011.

[83] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.

[84] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *IEEE International Conference on Robotics and Automation*, pages 500–505, 1985.

[85] Jonathan Bohren, Tully Foote, Jim Keller, Alex Kushleyev, Daniel Lee, Alex Stewart, Paul Vernaza, Jason Derenick, John Spletzer, and Brian Satterfield. Little Ben: The Ben Franklin Racing Team's entry in the 2007 DARPA Urban Challenge. *Journal of Field Robotics*, 25(9):598–614, 2008.

[86] Michael Montemerlo, Jan Becker, Suhrid Bhat, Hendrik Dahlkamp, Dmitri Dolgov, Scott Ettinger, Dirk Haehnel, Tim Hilden, Gabe Hoffmann, Burkhard Huhnke, Doug Johnston, Stefan Klumpp, Dirk Langer, Anthony Levandowski, Jesse Levinson, Julien Marcil, David Orenstein, Johannes Paefgen, Isaac Penny, Anna Petrovskaya, Mike Pflueger, Ganymed Stanek, David Stavens, Antone Vogt, and Sebastian Thrun. *Junior: The Stanford Entry in the Urban Challenge*, pages 91–123. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[87] Thomas M. Howard and Alonzo Kelly. Optimal Rough Terrain Trajectory Generation for Wheeled Mobile Robots. *The International Journal of Robotics Research*, 26(2):141–166, 2007.

[88] Julius Ziegler and Christoph Stiller. Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1879–1884, 2009.

[89] John M. Dolan und Jin-Woo Lee. Matthew McNaughton, Chris Urmson. Motion planning for autonomous driving with a conformal spatiotemporal lattice. In *IEEE International Conference on Robotics and Automation*, page 4889–4895, Shanghai, 5 2011.

[90] Moritz Werling, Julius Ziegler, Sören Kammel, and Sebastian Thrun. Optimal Trajectory Generation for Dynamic Street Scenarios in a Frenet Frame. *IEEE International Conference on Robotics and Automation*, pages 987—-993, 2010.

[91] Martin Keller. *Trajektorienplanung zur Kollisionsvermeidung im Straßenverkehr*. PhD thesis, Technische Universität Dortmund, 2017.

[92] Alanzo Kelly and Bryan Nagy. Reactive nonholonomic trajectory generation via parametric optimal control. *The International Journal of Robotics Research*, 22(7-8):583–601, 2003.

[93] Wenda Xu, Junqing Wei, John M. Dolan, Huijing Zhao, and Hongbin Zha. A real-time motion planner with trajectory optimization for autonomous vehicles. In *IEEE International Conference on Robotics and Automation*, pages 2061–2067, 2012.

[94] Christian Rathgeber. *Trajektorienplanung und -folgeregelung für assistiertes bis hochautomatisiertes Fahren*. PhD thesis, Technischen Universität Berlin, 2016.

[95] Sterling Anderson, Steven Peters, Thomas Pilutti, and Karl Iagnemma. An optimal-control-based framework for trajectory planning, threat assessment, and semi-autonomous control of passenger vehicles in hazard avoidance scenarios. *International Journal of Vehicle Autonomous Systems*, 8:190–216, 2010.

[96] Moritz Werling and Darren Liccardo. Automatic collision avoidance using model-predictive online optimization. In *IEEE 51st IEEE Conference on Decision and Control*, pages 6309–6314, 12 2012.

[97] Benjamin Gutjahr, Christian Pek, Lutz Gröll, and Moritz Werling. Rechenеffiziente Trajektorienoptimierung für Fahrzeuge mittels quadratischem Programm. *at - Automatisierungstechnik*, 64:786 – 794, 2016.

[98] Robert Ritschel, Frank Schrödel, Juliane Hädrich, and Jens Jäkel. Nonlinear Model Predictive Path-Following Control for Highly Automated Driving. *IFAC-PapersOnLine*, 52(8):350 – 355, 2019.

[99] İlhan Mutlu, Matthias Freese, Khaled Alaa, and Frank Schrödel. Case Study on Model Free Determination of Optimal Trajectories in Highly automated driving. In *9th Symposium on Advances in Automotive Control*, 2019.

[100] Magnus R. Hestene. Calculus of Variations and Optimal Control Theory. *John Wiley & Sons, Inc. New York,*, 1966.

[101] Rick Voßwinkel, İlhan Mutlu, Khaled Alaa, and Frank Schrödel. A Modular and Model-Free Trajectory Planning Strategy for Automated Driving. In *European Control Conference*, 2020.

[102] Timm Faulwasser, Benjamin Kern, and Rolf Findeisen. Model predictive path-following for constrained nonlinear systems. In *Proceedings of the 48h IEEE Conference on Decision and Control held jointly with 28th Chinese Control Conference*, pages 8642–8647, 2009.

[103] Timm Faulwasser, Tobias Weber, Pablo Zometa, and Rolf Findeisen. Implementation of Nonlinear Model Predictive Path-Following Control for an Industrial Robot. *IEEE Transactions on Control Systems Technology*, 25(4):1505–1511, 2017.

[104] Martin Böck and Andreas Kugi. Real-time Nonlinear Model Predictive Path-Following Control of a Laboratory Tower Crane. *IEEE Transactions on Control Systems Technology*, 22(4):1461–1473, 2014.