Optimal Trajectories

Moritz Werling

Zusammenfassung New active driver assistance systems that work at the road- and navigation level as well as automated driving face a challenging task. They must permanently calculate the vehicle input commands (such as those for the steering, brakes, and the engine/powertrain) to realize a desired future vehicle movement, a driving trajectory. This trajectory has to be optimal in terms of some optimization criterion (in general a trade-off between comfort, safety, energy effort, and travelling time), needs to take the vehicular dynamics into account, and must incorporate lane boundaries or the predicted free space amidst (possibly moving) obstacles. This kind of optimization can be mathematically formulated as a so-called optimal control problem. To limit the calculation effort, the optimal control problem is usually solved only on a limited prediction interval (starting with the current time) leading to a receding horizon optimization. The chapter illustrates this practically proven approach in detail. Furthermore, the three general principles of dynamic optimization known from control theory and robotics are presented, namely calculus of variations, direct optimization, and dynamic programming. Furthermore, their application to driver assistance systems and automated driving is exemplified and the high practical relevance is supported by the given literature. Finally, the respective advantages and limitations of the optimization principles are discussed in detail proposing their combination for more involved system designs.

1 Introduction

Advanced collision avoidance systems, lane keeping support, traffic jam assistance, and remote valet parking; they all operate on the actuators to relieve the drivers of the lateral and/or longitudinal vehicle control or make it safer for them. Well-

Moritz Werling
BMW Group, München and Karlsruhe Institut of Technology, Karlsruhe, E-mail: moritz.werling@bmw.de

defined tasks, such as staying in the middle of the marked lane while following the vehicle ahead, can be handled by a set-point controller [16]. And yet standard automated parking maneuvers already require a calculated trajectory[1] that must be adapted to the available parking space. As systems need to cover more and more situations, the number of degrees of freedom increase, which makes a trajectory parameterization very complex, especially when the vehicle must take numerous obstacles into account. This calls for a systematic approach based on mathematical optimization (as opposed to heuristic approaches such as potential field and elastic bands methods, see e.g. [29, 6], with their inherent limitations, cf. [28]. In the chapter at hand, we address real-time trajectory optimization[2], a task that an automated vehicle faces when it travels through its environment, also referred to as motion planning in robotics [30, 31]. The focus will be on methods that engage with the longitudinal and lateral vehicle movement. However, the results can be transferred to novel warning systems that can also benefit from an optimal trajectory prediction, see e.g. [11]. Generally speaking, a trajectory optimization method is sought that can handle both structured (e.g. streets) and unstructured environments (parking lots), one that works amongst cluttered static obstacles and in moving traffic as well as exhibits a natural, human-like, anticipatory driving behavior. Using more technical terms, the method should be easy to implement, parameterize, adapt, scale well with the number of vehicle states and the length of the optimization horizon, incorporate nonlinear, high fidelity vehicle models, combine the lateral and longitudinal motion, be complete [3], allow for both grid maps and object lists representations (with predicted future poses) of the obstacles, be numerically stable, and transparent in its convergence behavior (if applicable). Also, the calculation effort must be low to allow for short optimization cycles on (low performance) electronic control units so that the vehicle can quickly react to sudden changes in the environment. Unfortunately, there is no such single method that has all these properties. And, most likely, there will never be one. However, different optimization methods can be combined to get as close as possible to the above requirements. The next section therefore gives a closer look into the basic principles of trajectory optimization and their application.

2 Dynamic Optimization

When engineers speak about optimization, they usually refer to static optimization, in which the optimization variables $\boldsymbol{p}$ are finite, also called parameters (e.g., finding the most efficient operating point of an engine). Then optimal refers to some well-defined

[1] More precisely: a path, which does not have any time dependency

[2] Notice, that in control theory the term trajectory planning usually implies that there is no feedback of the actual system states on the trajectory. The dynamical system is then only stabilized by a downstream trajectory tracking controller, which is not always advisable. We therefore use the term trajectory optimization instead to be independent of the utilized stabilization concept.

[3] A complete algorithm always finds the solution if it exists.

optimization criterion, usually the minimization of a cost function $J(\boldsymbol{p})$ (e.g., fuel consumption per hour). Trajectory optimization is different in that the optimization variables are functions $\boldsymbol{x}(t)$ of an independent variable t, usually time. It is also called dynamic or infinite-dimensional optimization. Evaluating $\boldsymbol{x}(t)$ therefore requires a cost functional (a "function of a function"), which quantifies the "quality" of the trajectory $\boldsymbol{x}(t)$ by a scalar value. Due to the vehicular focus, a special case will be considered, one that requires the trajectory $\boldsymbol{x}(t)$ to be consistent with some dynamical system model which has an input u. Without such model, the optimization cannot incorporate the inherent properties and physical limitations of the vehicle. This special case of dynamic optimization is called an optimal control problem (e.g., [33]).

2.1 Optimal Control Problem

A fairly general formulation of the optimal control problem (OCP) reads:

Minimize the cost functional

$$J(\boldsymbol{u}(t)) = \int_{t_0}^{t_f} l(\boldsymbol{x}(t), \boldsymbol{u}(t), t)\,\mathrm{d}t \;+\; V(\boldsymbol{x}(t_f), t_f) \tag{1a}$$

subject to the system dynamics

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), t), \quad \boldsymbol{x}(t_0) = \boldsymbol{x}_0 \tag{1b}$$

as well as the equality and inequality constraints

$$\boldsymbol{g}(\boldsymbol{x}(t_f), t_f) = \boldsymbol{0} \tag{1c}$$

$$\boldsymbol{h}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \leq \boldsymbol{0}, \quad \forall t \in [t_0, t_f]\ . \tag{1d}$$

In other words, for our (possibly nonlinear and time variant) system with state $\boldsymbol{x} \in \mathbb{R}^n$ and input $\boldsymbol{u} \in \mathbb{R}^m$ we seek on the interval $t \in [t_0, t_f]$ the input trajectory $\boldsymbol{u}(t)$ that minimizes the cost functional J while steering (in the truest sense of the word) the system from its initial state $\boldsymbol{x}_0$ to an end state $\boldsymbol{x}(t_f)$, so that the equality and inequality constraints are fulfilled at all times. The optimal input trajectory is usually denoted by $\boldsymbol{u}^*(t)$ and the resulting state trajectory by $\boldsymbol{x}^*(t)$, see Fig. 1. Notice that J comprises integral costs l and end point costs V and is not only a functional of the input $\boldsymbol{u}(t)$ but also of the states $\boldsymbol{x}(t)$. Furthermore, if the final time t_f is not given, it becomes part of the optimization, so that the length of the trajectory will also be optimized.

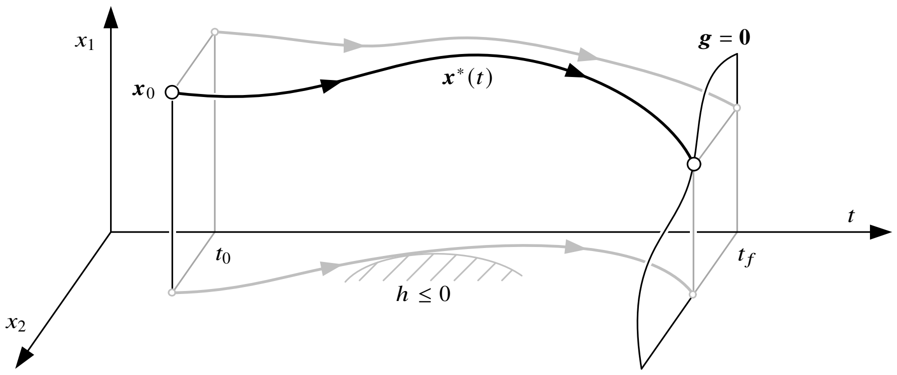

Abb. 1 Example of an optimal trajectory $x^*(t) \in \mathbb{R}^2$ with end constraints $g = 0$ at a fixed end time t_f and inequality constraint $h \le 0$ for x_2

2.2 Problem Formulation for DAS and Automated Driving

Coming back to the automotive application, equation (1b) of the previous section describes the dynamics of the vehicle. This state space model also includes the planar motion, either relative to some reference such as the lane center, see Sect. 3.1.2, or relative to a stationary origin, see Sect. 3.3.4. Undesired vehicle motion such as deviations from the lane center, detours, dangerous vehicle states (e.g., large slip angles), or uncomfortable jerks, e.g., caused by hectic steering actions will be penalized in the cost functional (1a). The free space prediction between the generally moving obstacles can be described by the inequality constraints (1d). And the end constraints (1c) can be utilized to require that the optimized vehicle trajectory will be aligned to the road at the end of the optimization interval. As will be explained in Sect. 5, this final state plays an important role for the stability of the "replanning" algorithm, therefore special costs $V(x(t_f), t_f)$ can be introduced in (1a).

3 Solving the Optimal Control Problem

All known approaches to the OCP can be assigned to one of the following three principles, see e.g. [8].

3.1 Approach I: Calculus of Variations

The classical approach to the OCP is *calculus of variations*, which delivers valuable insight in the solution.

3.1.1 Theoretical Background: Hamilton Equations

Static optimization problems can be tackled by *differential calculus*. It is well known that the first derivative of a function $J(\boldsymbol{p})$ is equal to zero at a minimum (or any other stationary point). For multivariate problems we can write $\nabla J(\boldsymbol{p}) = 0$ which leads to a set of (algebraic) equations that the optimum $\boldsymbol{p}^*$ must satisfy. The extension to problems with equality constraints requires the method of so-called *Lagrange-multipliers*, yielding first order necessary conditions for optimality. Analogously for dynamic optimization, variational calculus requires that the first variation of the functional $J(\boldsymbol{u}(t))$ vanishes for the optimal control function $\boldsymbol{u}^*$ which is often written as $\delta J(\boldsymbol{u}(t)) = 0$. In order to incorporate the system dynamics in the OCP, which constitute (differential) equality constraints, we can also apply the Lagrange-multiplier method. This yields a set of differential equations, the so-called *Hamilton equations*:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t) \tag{2a}$$

$$\dot{\boldsymbol{\lambda}} = -\frac{\partial l}{\partial \boldsymbol{x}} - \left[\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right]^{\mathrm{T}} \boldsymbol{\lambda} \tag{2b}$$

$$\boldsymbol{0} = \frac{\partial l}{\partial \boldsymbol{u}} + \left[\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right]^{\mathrm{T}} \boldsymbol{\lambda} \tag{2c}$$

They are *first order necessary conditions* for our OCP (leaving out the inequality constraints (1d)). The function $\boldsymbol{\lambda}(t)$ constitutes the Lagrange-multipliers, here called *co-states*. Besides the initial condition

$$\boldsymbol{x}(t_0) = \boldsymbol{x}_0 \tag{3}$$

the optimal trajectory must fulfill the (algebraic) *transversality conditions*, depending on whether the end state $\boldsymbol{x}(t_f)$ is constraint by (1c) and/or the final time t_f is given, see e.g. [33]. The simplest condition requires a fixed end state $\boldsymbol{x}_f$ at a given end time t_f so that

$$\boldsymbol{x}(t_f) = \boldsymbol{x}_f \tag{4}$$

Either way, this results in a *boundary value problem*, which in general needs to be solved numerically. This so-called *indirect approach* is very accurate but, from experience, not as flexible as the direct approach that we will introduce in Sect. 3.2. However, for simple OCPs the resultant boundary value problem can be solved analytically, leading to fast computable optimal trajectory primitives with broad applications (see Sect. 4).

3.1.2 Example Application: Automated Lane Change

We will now apply the described method to the generation of optimal lane change primitives. The lateral motion across the road can be modelled as a triple integrator system with states $\boldsymbol{x} = [x_1, x_2, x_3]^{\mathrm{T}}$, namely the position, the lateral velocity, and the lateral acceleration, respectively, all within the reference frame of some curve, see Fig. 2.
Then, the system dynamics are described by

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, u) = [x_2, x_3, u]^{\mathrm{T}} \tag{5}$$

where u represents the lateral jerk, which is the third derivative of the lateral position. We will now seek for the optimal system input $u^*(t)$ that transfers the integrator system from its initial state

$$\boldsymbol{x}(0) = \boldsymbol{x}_0 \tag{6}$$

to a given end state

$$\boldsymbol{x}(t_f) = \boldsymbol{0} \, , \tag{7}$$

at a given end time t_f. Amongst all trajectories we seek for the one that minimizes the integral of the jerk-square, that is

$$J = \int_0^{t_f} \frac{1}{2} u(t)^2 \mathrm{d}t \tag{8}$$

so that the movement feels most pleasant to the passengers. Notice that the final cost here has no influence on the solution due to the fixed end state, so that we can set $V = 0$ in (1a).
With $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \lambda_3]^{\mathrm{T}}$, evaluating the so-called control equation (2c) we obtain

$$0 = u + \lambda_3 \quad \Rightarrow \quad u = -\lambda_3 \; . \tag{9}$$

The co-state equation (2b) yields

$$\dot{\boldsymbol{\lambda}} = \begin{bmatrix} \dot{\lambda}_1 \\ \dot{\lambda}_2 \\ \dot{\lambda}_3 \end{bmatrix} = \begin{bmatrix} 0 \\ -\lambda_1 \\ -\lambda_2 \end{bmatrix}$$

and therefore, we get

$$\begin{aligned} \dot{\lambda}_1 &= 0 &&\Rightarrow \quad \lambda_1(t) = -c_1 \\ \dot{\lambda}_2 &= -\lambda_1 &&\Rightarrow \quad \lambda_2(t) = c_1 t + c_2 \\ \dot{\lambda}_3 &= -\lambda_2 &&\Rightarrow \quad \lambda_3(t) = -\frac{1}{2} c_1 t^2 - c_2 t - c_3 \, , \end{aligned} \tag{10}$$

With λ_3 from (10) it can be seen from (9) that the optimal control input function is a third order polynomial with yet unknown integration constants $c_1,\ c_2$, and c_3. Substituting $u(t)$ in the state equation (5) we find the optimal trajectory to

$$\dot{x}_3 = \lambda_3 \quad \Rightarrow \quad x_3(t) = \frac{1}{6}c_1t^3 + \frac{1}{2}c_2t^2 + c_3t + c_4 \tag{11}$$

$$\dot{x}_2 = x_3 \quad \Rightarrow \quad x_2(t) = \frac{1}{24}c_1t^4 + \frac{1}{6}c_2t^3 + \frac{1}{2}c_3t^2 + c_4t + c_5 \tag{12}$$

$$\dot{x}_1 = x_2 \quad \Rightarrow \quad x_1(t) = \frac{1}{120}c_1t^5 + \frac{1}{24}c_2t^4 + \frac{1}{6}c_3t^3 + \frac{1}{2}c_4t^2 + c_5t + c_6 \tag{13}$$

with additional integration constants c_4, c_5, and c_6. To comply with the initial state (6) and end constraint (7) the integration constants need to be chosen accordingly. As (10)-(13) are linear with respect to the constants this can be done by simple linear algebra, leading to the trajectory (a) in Fig. 2
Similarly, we can find the solution to the OCP with a free end state (and a free end time). In this case we require the end state to be as close to (and as soon at) the reference as possible by setting

$$V(x) = k_1x_1(t_f)^2 + k_2x_2(t_f)^2 + k_3x_3(t_f)^2, \qquad k_i > 0$$

$$(V(x) = k_1x_1(t_f)^2 + k_2x_2(t_f)^2 + k_3x_3(t_f)^2 + k_tt_f, \qquad k_i, k_t > 0)$$

The new transversality conditions will only lead to a different end state and end time, see (b) and (c) in Fig 2. However, the optimal function class, that is the 5th-order polynomial, will stay the same.

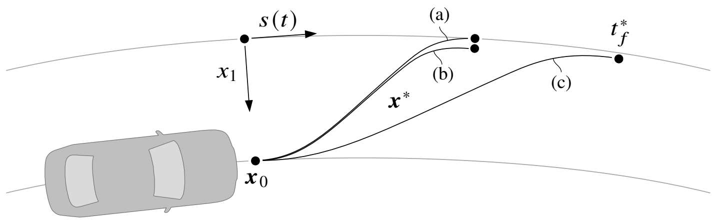

Abb. 2 Optimal lane changes for (a) a given end time and end state; (b) a given end time and a free end state; (c) a free end time and a free end state

3.1.3 Further Readings

The previous calculations can be analogously carried out for the longitudinal movement, which leads to a very comfortable braking characteristic [23]. The longitudinal and lateral movements can also be combined by a regular local and temporal sampling of target states across and along the street. This leads to a reactive algorithm [47], which prevents collisions with static and moving obstacles widely used in academia and industry. The variational approach was generalized to problems with input constraints, known as *Pontryagin's minimum principle*. As for the vehicular application, it yields the shortest path connecting two poses of a vehicle with a limited turn radius [10, 43, 5]. Even more involved is finding variational solutions to problems with state constraints such as the optimal braking application in Fig. 3 from [48]. Numerical solutions to the first order necessary conditions can be found by so-called indirect methods (see e.g. [18]), which provide very accurate results in general. Contrary to direct methods, as described in the sequel, they need to determine initial conditions for the co-states, which makes the application challenging for an automotive online application.

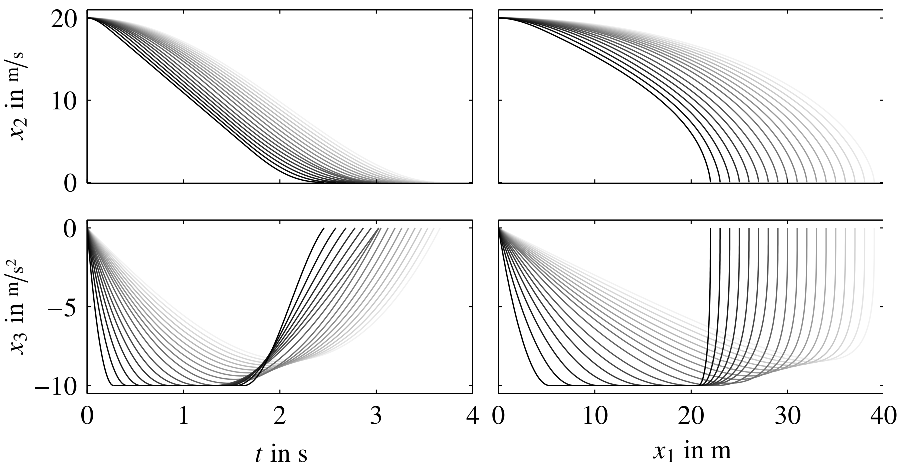

Abb. 3 Minimal jerk-square minimal time stopping trajectories with acceleration constraint with $a \geq -10$m/s²

3.2 Approach II: Direct Optimization Techniques

Direct optimization is probably the most widely explored approach in *model-predictive control*. It approximates the dynamic optimization problem of the OCP to a static one, as the latter can be efficiently solved by well-established numerical solvers. The essence of this approach is a *finite-dimensional parameterization* of the input, state, or output trajectory.

3.2.1 Theoretical Background: Finite Parameterization Approximation

We will now introduce a very common method called single shooting. In a first step, we chose a finite dimensional parameterization of the input

$$\boldsymbol{u}(t) = \boldsymbol{\psi}(t, \bar{\boldsymbol{u}})$$

such as a piecewise constant interpolation (see Fig. 4), a polynomial, or a spline. The input trajectory is therefore fully described by the finite parameter vector $\bar{\boldsymbol{u}}$. The system dynamics (1b) now read

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{\psi}(t, \bar{\boldsymbol{u}}), t), \quad \boldsymbol{x}(t_0) = \boldsymbol{x}_0 \ ,$$

This constitutes an initial value problem, which can be solved by an ordinary differential equations solver. We denote the resultant trajectory by

$$\boldsymbol{x}(t) = \boldsymbol{\phi}(t, \bar{\boldsymbol{u}}).$$

Furthermore, it is standard practice that the inequality constraints are only required to hold at N discrete equidistant points in time $t_i, i = 1, \ldots, N$, so that the number of inequality constraints is also finite.
Thus, the OCP was transformed to the following static optimization problem:

Minimize the cost function

$$J(\bar{\boldsymbol{u}}, t) = \int_{t_0}^{t_f} l(\boldsymbol{\phi}(t, \bar{\boldsymbol{u}}), \boldsymbol{\psi}(t, \bar{\boldsymbol{u}}), t)\, \mathrm{d}t + V(\boldsymbol{\phi}(t_f, \bar{\boldsymbol{u}})) \tag{14a}$$

subject to the equality and inequality constraints

$$\boldsymbol{g}(\boldsymbol{\phi}(t_f, \bar{\boldsymbol{u}}), t_f) = \boldsymbol{0} \tag{14b}$$
$$\boldsymbol{h}(\boldsymbol{\phi}(t_i, \bar{\boldsymbol{u}}), \boldsymbol{\psi}(t_i, \bar{\boldsymbol{u}})) \leq \boldsymbol{0}, \quad i = 0, \ldots, N \ . \tag{14c}$$

Loosely speaking, for a first guess $\bar{\boldsymbol{u}}_0$ the system will be simulated in a “single shoot” for $t \in [t_0, t_f]$ starting from $\boldsymbol{x}_0$. Then, the total costs J are evaluate as well as the constraints $\boldsymbol{g}$ and $\boldsymbol{h}$ in (14a), (14b), and (14c), respectively. These values are then fed back to a numerical solver, which repeats this procedure by a variation of $\bar{\boldsymbol{u}}$ to conclude how to modify the parameter so that J gets smaller without violating $\boldsymbol{g} = \boldsymbol{0}$ and $\boldsymbol{h} \leq \boldsymbol{0}$. When the solution does not significantly change any more or a certain number of iterations have been reached, the optimization will be terminated.

3.2.2 Example Application: Emergency Obstacle Avoidance

The direct optimization based on a non-linear model has been successfully demonstrated in [46]. However, the computational demand is too high for today’s electronic control units, since a sequence of *quadratic program* (QP) need to be solved. In order

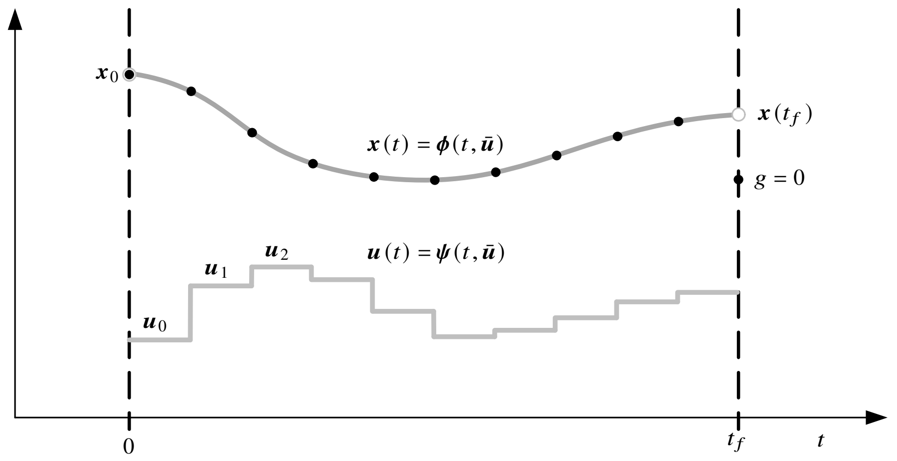

Abb. 4 Finite parameterization of the input and sampling of the constraints [39]

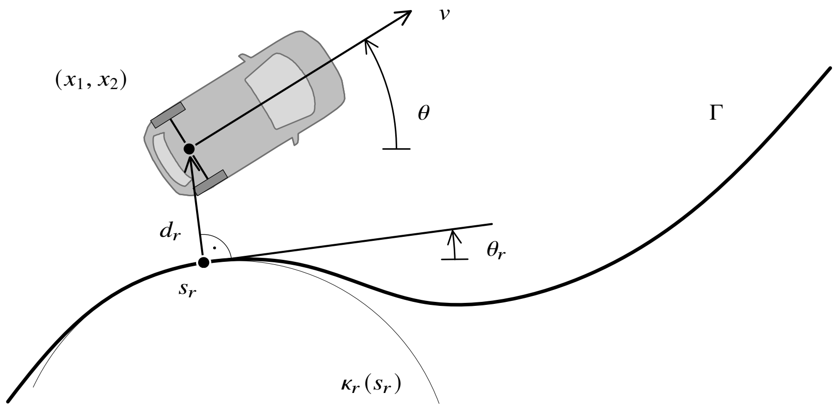

Abb. 5 Kinematic vehicle model with respect to a given reference curve Γ

to reduce the computational effort to a few milliseconds, [12, 7], for instance, approximate the formulation of trajectory optimization directly as a QP and a solution can be found in a single step. We now sketch the approach from [22], which has the potential to cover a wide range of series applications. In doing so, we focus on the lateral motion along a reference curve Γ such as a lane center.

We assume in a first step, that the longitudinal profile $v(t)$ is given from an upstream algorithm. The difference in orientation $\theta - \theta_r$, see Fig. 5, is typically smaller than 20° [45] and d_r is small with respect to the reference curve's radius $r_r = \frac{1}{\kappa_r}$. We therefore model $\sin(\alpha) \approx \alpha$ and $\cos(\alpha) \approx 1$. Furthermore $\frac{d_r}{r_r} = d_r \kappa_r \ll 1$, which give us for the system state $\boldsymbol{x} = [d_r, \theta, \kappa, \theta_r, \kappa_r]^{\mathrm{T}}$ the following linear time-variant model

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}_C(t)\boldsymbol{x}(t) + \boldsymbol{B}_C(t)u(t) + \boldsymbol{E}_C(t)z(t),\ \boldsymbol{x}(t_0) = \boldsymbol{x}_0 \tag{15}$$

where

$$\boldsymbol{A}_C(t) = \begin{bmatrix} 0 & v(t) & 0 & -v(t) & 0 \\ 0 & 0 & v(t) & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & v(t) \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$
$$\boldsymbol{B}_C(t) = [\,0,0,1,0,0\,]^{\mathrm{T}}, \quad \boldsymbol{E}_C(t) = [\,0,0,0,0,1\,]^{\mathrm{T}},$$

Notice, that the state has been extended by the instantly driven curvature κ of the reference point, see Fig. 5, to ensure continuous curvatures, as well as the reference value κ_r and its integral θ_r of Γ. The latter simulates the curvature influence as a known disturbance.

To minimize the number of constraints, we define the system output based on the idea of approximating the vehicle shape by several circles [51], in our case three, see Fig. 6. With $l_1 = 0, l_2 = \frac{l}{2}, l_3 = l$, we define the circles' center positions with respect to the reference curve Γ by

$$d_i = d_r + l_i \sin(\theta - \theta_r) \approx d + l_i(\theta - \theta_r), \quad i = 1,2,3. \tag{16}$$

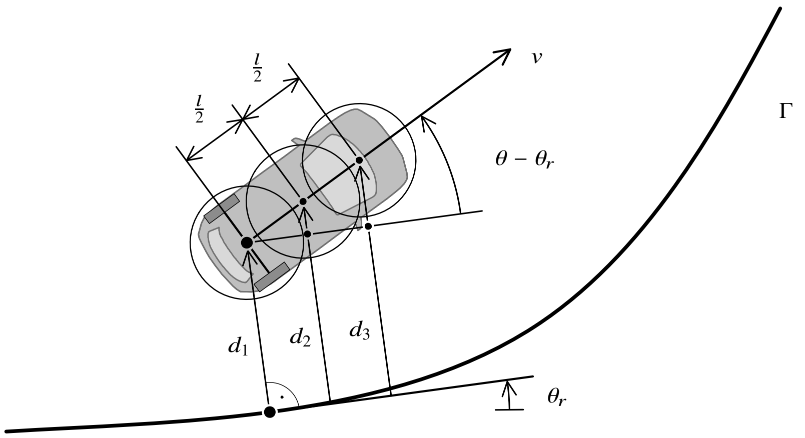

Abb. 6 Visualization of the defined system outputs

Next, we follow the idea of a finite parameterization of the input and transform the continuous-time vehicle prediction model (15) and (**??**) into its equivalent time-discrete form in one step. Assuming the system matrix $\boldsymbol{A}_C(t)$ as well as the system input $u(t)$ and the system disturbance $z(t)$ to be constant within each discretization interval k with length T_s, we can use the *Laplace transformation*, see [22] for details, to get the time-discrete, time-varying system model

$$\begin{aligned} \boldsymbol{x}(k+1) &= \boldsymbol{A}(k)\boldsymbol{x}(k) + \boldsymbol{B}(k)u(k) + \boldsymbol{E}(k)z(k), \\ \boldsymbol{y}(k) &= \boldsymbol{C}(k)\boldsymbol{x}(k), \qquad \boldsymbol{x}(0) = \boldsymbol{x}_0. \end{aligned} \tag{17}$$

As we want to minimize the discretization error caused by the system disturbance $z(k)$ and keep $z(k)$ constant within each discretization interval we define

$$z(k) = \frac{\kappa_r(k+1) - \kappa_r(k)}{T_s} \tag{18}$$

approximating the reference curvature as a forward first order hold.

In the third step, we define upper and lower limits for the system output based on the perceived environment and the vehicle dynamics.

As depicted in Fig. 6, the system output (16) can be interpreted as a normal distance between the reference curve Γ and the center position of each circle. Due to the presence of moving traffic the maximum admissible deviation, that is how far each circle can be shifted form the center to either side before intersecting an object, obstacle or the lane markings, is not only dependent on the vehicle's position s_r along the reference curve Γ, but also on the time step k, namely

$$d_{i,\min}(s_r(k), k) \le d_i(k) \le d_{i,\max}(s_r(k), k)\,,\ i = 1, 2, 3, \tag{19}$$

where $s_r(k)$ is given by the integration of the desired future velocity profile.

Also, we would like to account for the limited turning circle by introducing a fixed time-invariant upper $\kappa_{\max,\delta}$ and lower $\kappa_{\min,\delta}$ for the κ signal in the system output. Additionally, to account for the total wheel traction, we combine these bounds by a time-varying term which is dependent on $v(k)$ so that

$$\underbrace{\max\left(\kappa_{\min,\delta}, \kappa_{\min,\mu}(k)\right)}_{\kappa_{\min}} \le \kappa(k) \le \underbrace{\min\left(\kappa_{\max,\delta}, \kappa_{\max,\mu}(k)\right)}_{\kappa_{\max}},$$

where $\kappa_{\min,\mu}(k) = \kappa_{\min}(v(k), \mu)$ and $\kappa_{\max,\mu}(k) = \kappa_{\max}(v(k), \mu)$ are derived using the relations stated by the so-called *Kamm's circle* based on an the estimated friction parameter μ.

Lastly, as the steering actuator can only supply a certain maximum power, the vehicle's steering rate is limited. We therefore confine the curvature's rate of change, that is the system input, by box constraints

$$u_{\min} \le u(k) \le u_{\max}. \tag{20}$$

In the last step of the problem modelling, we define a quadratic running cost function of the form

$$l(\boldsymbol{x}(k), u(k)) = w_d d_r^2 + w_\theta[\theta - \theta_r]^2 + w_\kappa[\kappa - \kappa_r]^2 + w_u u^2 \tag{21}$$

with $w_d, w_\theta, w_\kappa, w_u > 0$. The first three terms penalize deviations from the reference curve Γ whereas the last term encourages a smoothing behavior. After some straightforward but laborious transformations, we substitute the system state in the

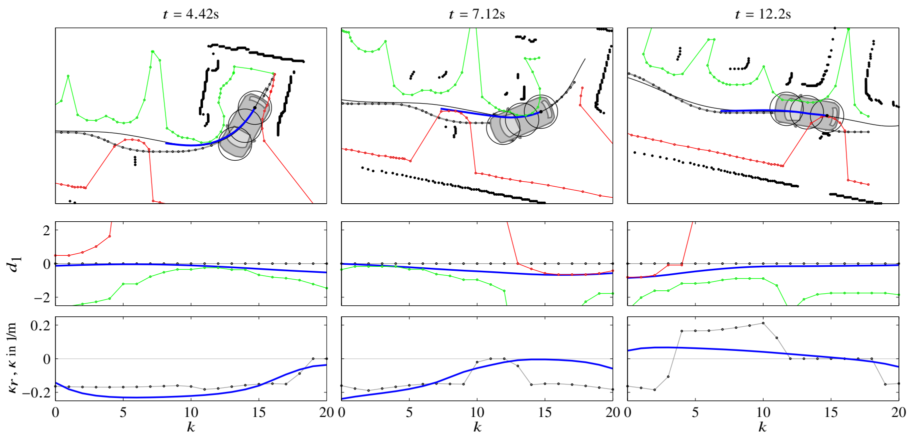

Abb. 7 Bird's eye view of the moving vehicle for the parking scenario at three consecutive instants of time with its correspondent results for the lateral position for the rear axle d_1 and the curvature κ, where the blue bold line represents the optimized trajectory

cost function and the output constraint equation and finally get a quadratic program of the form

$$J = \bar{\mathbf{u}}^{\mathrm{T}} \boldsymbol{H} \bar{\mathbf{u}} + \boldsymbol{F} \bar{\mathbf{u}} \tag{22}$$

$$\boldsymbol{A}_c \bar{\mathbf{u}} \leq \mathbf{b}_c \tag{23}$$

for the input vector $\bar{\mathbf{u}}$, where $\boldsymbol{H}, \boldsymbol{F}, \boldsymbol{A}_c, \mathbf{b}_c$ are suitably sized matrices and vectors as functions of the initial state $\boldsymbol{x}_0$ the disturbance signal $\kappa_r(k)$ and the velocity reference $v(k)$ as well as the predicted environment.

The minimization constitutes a quadratic program, which needs to be updated and solved as the vehicle drives along the reference curve, see e.g. the parking scenario in Fig. 8.

3.2.3 Further Readings

Single shooting has been intensively studied for many vehicular applications such by [27, 12, 25, 49, 17, 40]. Nevertheless, many other numerical methods exist such as *multi-shooting* [4] and *collocation* [24] that might gain in importance in the future when dealing with instable vehicle models in challenging driving conditions. As for all direct optimization methods they lead to a static optimization problem, which can be most successfully solved by *sequential quadratic programming* (SQP) techniques or *interior point methods* (IP), see [38]. As an alternative to a finite parameterization of the input, for flat systems [44] the flat output can be parameterized instead [26], which has been successfully demonstrated in complex inner-city scenarios by [53].

3.3 Approach III: Dynamic Programming

Certain tasks such as parking in several moves or finding the way through multiple moving obstacles are *combinatorial* (non-convex) *problems*. They cannot be tackled by (local) *direct optimization methods* as the latter rely on an initial solution. However, *dynamic programming* is a principle from [1] that significantly reduces the computational burden of these combinatorial problems, so that the *global optimum* can be efficiently obtained. It is therefore no surprise that it can be found in numerous (discrete) optimization algorithms.

3.3.1 Theroetical Backgorund: Bellman's Principle of Optimality

Richard *Bellman's principle of optimality* states:

Definition 1 "An *optimal policy* has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision."

In other words, an optimal trajectory is composed of optimal sub-trajectories. This can be understood by looking at Fig. 9, which shows the optimal trajectory $\boldsymbol{x}^*(t)$ in black connecting $\boldsymbol{x_0}$ with $\boldsymbol{x_f}$. The sub-trajectory (b), bringing $\boldsymbol{x^*}(t_1)$ to $\boldsymbol{x_f}$, must be optimal, too. Otherwise, there would be another sub-trajectory (c) (grey), that would lead to a better total trajectory (a)+(c), which is contrary to the optimal trajectory $x^*(t)$ comprising (a)+(b).

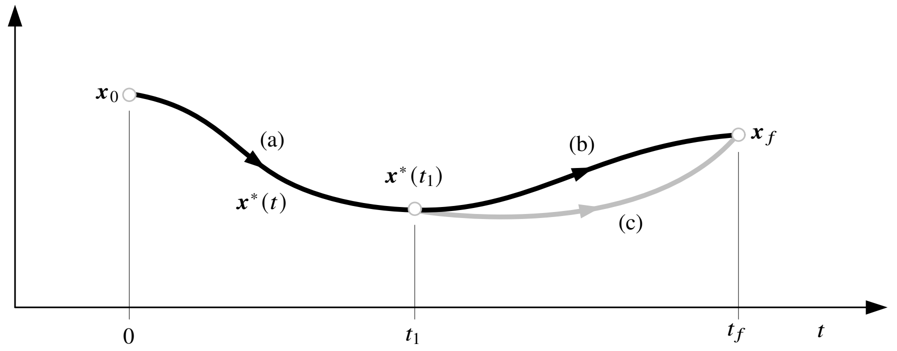

Abb. 8 Illustration of Bellman's Principle of Optimality

The largest practical benefit from this principle is received when time-discretizing the OCP. We therefore consider the time-discrete process

$$\boldsymbol{x}(k+1) = \boldsymbol{f}(\boldsymbol{x}(k), \boldsymbol{u}(k), k), \quad k = 0, \dots, k_f - 1 \tag{24}$$

and seek for the optimal steering sequence $\boldsymbol{u}^*(k)$ that minimizes the cost function

$$J = \sum_{k=0}^{k_f-1} l(\boldsymbol{x}(k), \boldsymbol{u}(k), k) \ . \tag{25}$$

Notice, that constraints can be easily incorporated in (25) by setting the costs $l = \infty$ when the equality or inequality equations are violated. A basic element of dynamic programming is memorizing the costs of sub-trajectories. Therefore, we define the so-called *cost-to-go*

$$G = \sum_{\kappa=k}^{k_f-1} l(\boldsymbol{x}(\kappa), \boldsymbol{u}(\kappa), \kappa)$$

(notice the difference between κ and k), which incur when going from an intermediate state $x(k)$ all the way to k_f.
At this point, it should be noticed that the minimal cost-to-go, denoted by G^*, is nearly as good as the solution u^* itself, as will be seen later. The minimal cost-to-go can be found in *Bellman's Recursion Formula*

$$G^*(\boldsymbol{x}(k), k) = \min_{\boldsymbol{u}(k)} \{ \, l(\boldsymbol{x}(k), \boldsymbol{u}(k), k) + G^*(\boldsymbol{f}(\boldsymbol{x}(k), \boldsymbol{u}(k), k), k+1) \, \} \, ,$$

which can be derived from the principle of optimality in a few steps. In words, it relates the minimal cost-to-go of the state $\boldsymbol{x}(k)$ at the k^{th} step, namely $G^*(\boldsymbol{x}(k), k)$, to the subsequent minimal cost-to-go $G^*(f(\boldsymbol{x}(k), \boldsymbol{u}(k), k), k+1)$ in consideration of the best choice of the possible inputs $\boldsymbol{u}(k)$ with its associated stage cost $l(\boldsymbol{x}(k), \boldsymbol{u}(k), k)$. The formula is used to break down the multi-staged OCP into simpler single-staged optimizations, which numerous algorithms exploit to their advantage.

3.3.2 Dynamic Programming in a Temporal Decision Process

We assume that the system input can only take discrete values from a given set, which depends on the current state and the current time, that is $\boldsymbol{u}(k) \in \mathcal{U}(\boldsymbol{x}(k), k)$. The set ensures that the input will transfer the system from one discrete state $\boldsymbol{x}(k) \in \mathcal{X}(k)$ to the next. We therefore get a multi-stage decision process, such as the simple example in Fig. 10 with three discrete states in each time step (except for the goal state $\boldsymbol{x}_f$).

Even though the naive approach of evaluating all $3^3 = 27$ possibilities would here be feasible, this is clearly prohibitive for realistically sized problems due to an exponential runtime of $O(m^{k_f})$, where m is the number of state transitions and k_f the optimization horizon. However, we can apply the recursion formula, start with the last stage $k = k_{f-1}$, and work our way back to the first step. In doing so we calculate the optimal cost-to-go of every state of each stage by contemplating all possible transients and memorize it. This way it can be accessed when evaluating the optimal cost-to-go of the previous stage. Every stage can be done in $O(mn)$, where

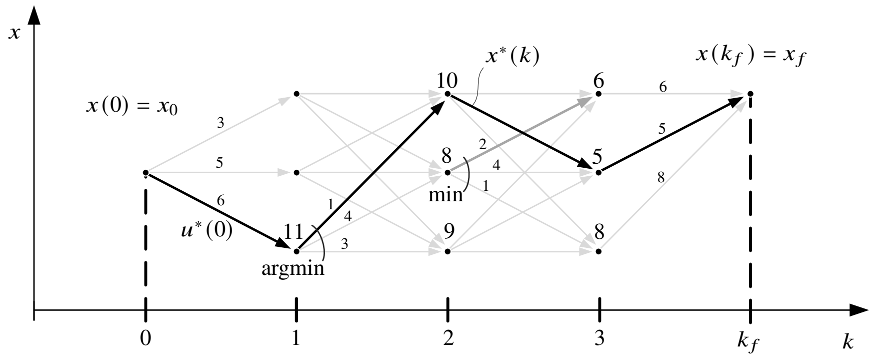

Abb. 9 Discrete decision process with the optimal trajectory

m and n equals the number of its inputs and states. This adds up to a total runtime of $O(mnk_f)$, which is only linear in the length of the optimization horizon. Alg. 6 summarizes the algorithm. The optimal input and state sequence $\boldsymbol{u}^*(k)$ and $\boldsymbol{x}^*(k)$

Algorithm 1 Dynamic Programming

```
1: $G^*(\boldsymbol{x}_f, k_f) \leftarrow 0$
2: **for** $k = k_f - 1$ **to** 0 **do**
3:     **for all** $\boldsymbol{x} \in \mathcal{X}$ **do**
4:         $G^*(\boldsymbol{x}(k), k) = \min_{\boldsymbol{u}(k)} \{ l(\boldsymbol{x}(k), \boldsymbol{u}(k), k) + G^*(\boldsymbol{f}(\boldsymbol{x}(k), \boldsymbol{u}(k), k), k+1) \}$
5:     **end for**
6: **end for**
```

can now be found by an efficient forward search starting at $\boldsymbol{x}_0$ and alternating

$$\boldsymbol{u}^*(k) = \operatorname*{argmin}_{\boldsymbol{u}(k)} \{ l(\boldsymbol{x}(k), \boldsymbol{u}(k), k) + G^*(\boldsymbol{f}(\boldsymbol{x}(k), \boldsymbol{u}(k), k), k+1) \} \tag{26}$$

and (24) for every stage to the end, see black arrows in Fig. 10.

3.3.3 Example Application: Optimal Overtaking

A simplified problem setup and its solution can be found in Fig. 11. Here, the lateral motion is optimized, so that the car can proceed at a constant speed without getting too close to the other vehicles. The system state comprises of the lateral position and velocity, hence $\boldsymbol{x}(k) = [d(k), v_{\text{lat}}(k)]^{\mathrm{T}}$, which is discretized with $\Delta d = 0.5\text{m}$ and $\Delta v_{\text{lat}} = 0.5\text{m/s}$ at $t_k = k \cdot \Delta t$ with $k = 1, \ldots, 10$ and $\Delta t = 1.0\text{s}$. Furthermore, the input set $\mathcal{U}$, i.e., all lateral acceleration profiles, is chosen as a first order polynomial with $-10.0\text{m/s}^2 < u(t) < 10.0\text{m/s}^2$. And lastly, the stage costs are defined as

$$l\left(\boldsymbol{x}(k),u(k),k\right)=\int_{t_k}^{t_k+\Delta t}u(t)^2\mathrm{d}t+k\left[d(k)-d_{\text{nearest}}(k)\right]^2+C_{\text{collision}}\left(\boldsymbol{x}(k),u(k),k\right)$$

with $k = 1.0$, where $d_{\text{nearest}}(k)$ denotes the lateral coordinate of the lane center closest to $d(k)$. The cost term $C_{\text{collision}}\left(x(k),u(k),k\right)$ equals to zero if the vehicle does not collide within the interval $[t_k, t_k + \Delta t]$ and else equals to infinity. Due to the few system states with their coarse discretization, the (global) solution is found within a few milliseconds.

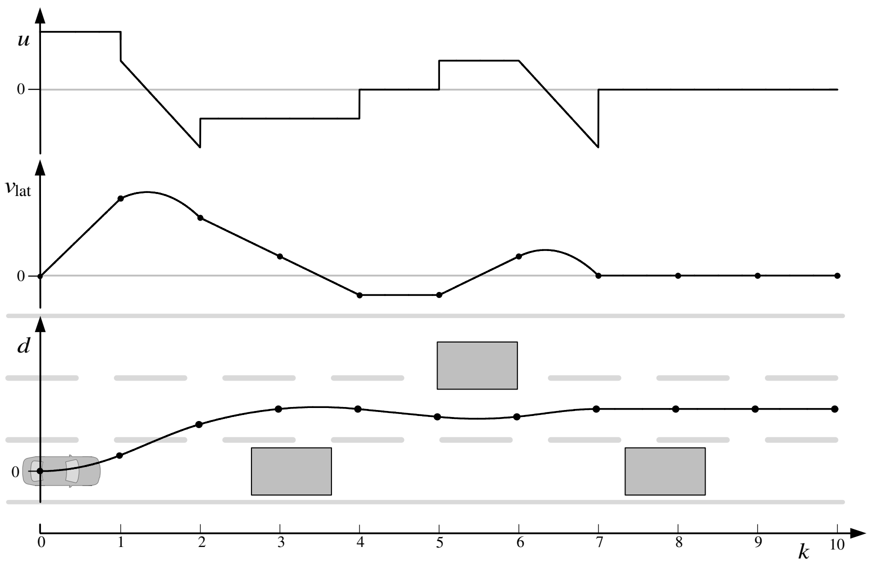

Abb. 10 Dynamic programming example

3.3.4 Further Readings

The presented iteration scheme is especially suited for structured dynamic environments as shown by [50, 35], as well as [20]. As can be seen in Fig. 13, for natural, human like trajectories it is most advantageous to align the sampling along and across the road course. The costs usually penalize not only jerky, accelerant movements, and the approximation to obstacles, but also deviations from the desired road center. As opposed to that, in unstructured environments, such as parking lots, there is no preferred direction, so that the sampling uniformly covers the $[x, y]$ plane as shown in Fig. 12. As static obstacles dominate the problem, temporal aspects are usually neglected meaning that it is not important at what time a location is visited by the vehicle (see [37, 52, 41, 34, 9]). This drastically simplifies the problem, however, the iteration cannot be applied anymore due to the loss of the processing sequence given by k (the decision graph becomes cyclic). Therefore, a substitute order needs to be chosen, one that still leads to the optimum: Investigating the state (expanding the node) with the lowest cost-to-come leads to *Dijkstra's algorithm*, unlike considering the state with the lowest (under)estimate on the total cost, which results in *A* algorithm*, an informed search. The latter is especially efficient in unstructured environments, as the costs are dominated by the covered distance, which can be well estimated by the Euclidean distance (the so-called heuristic), see [2].

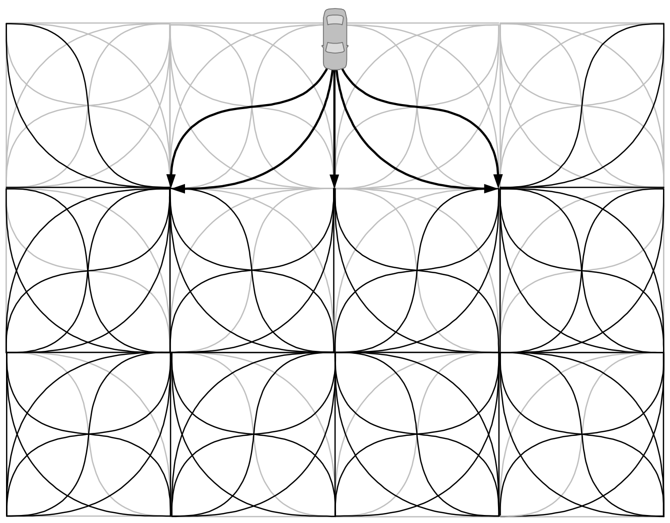
Abb. 11 State lattice for unstructured environments (illustration based on [36])

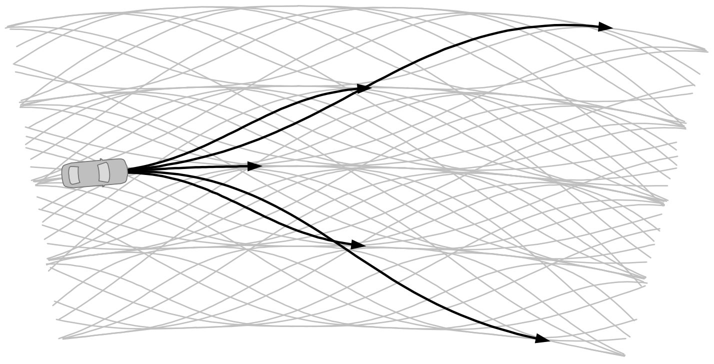
Abb. 12 State lattice for structured environments (illustration based on [35])

4 Comparison of the Approaches

When we compare dynamic programming with direct optimization, we realize that the two approaches possess orthogonal capabilities, see table below. Dynamic programming suffers the so-called *curse of dimension* [1] meaning that the approach does not scale well with the number of states. Its application is therefore limit to models with few system states (<3-4) with a coarse discretization. Direct optimization, however, can incorporate system models with numerous and continuous system states (>4) and is therefore able to directly feed the system input $\boldsymbol{u}(t)$. In turn, direct optimization cannot deal with arbitrary cost functionals due to numerical limitations (local convergence) of the underlying solver. Also, its runtime usually grows exponentially with the number of optimization variables so that the length of the optimization horizon is restricted to a few seconds. As opposed to that, dynamic programming can handle arbitrary costs and will always lead to the global optimum. Furthermore, dynamic programming scales comparably well with the length of the optimization horizon (cf. the linear runtime of the dynamic programming iteration scheme in Sect. 3.3.2). For complex, farsighted trajectory optimization tasks, the two approaches need to be combined. Dynamic programming will then yield only a “rough long-term plan”, which either serves as the reference trajectory (see e.g.[13, 14, 21]) and/or provide an initial guess [26] for the locally working direct optimization method. The latter will take a detailed model of the vehicle into account and improves the dynamic programming solution on a reduced optimization horizon to a feasible trajectory. The optimal state trajectory is either forwarded to a low-level feedback steering/acceleration controller or the optimal input trajectory is directly fed to the vehicle actuators. Closed form solutions from the calculus of variations are thereby often used to speedup dynamic programming. This can be in the form of a heuristic for an informed search (see e.g. [52] or a so-called analytical expansions [9], both of which, roughly speaking, approximate the remaining trajectory and therefore extend the computable optimization horizon, see Tab. 1.

Approach	many states	continuous states	global optimum	long horizon
DP	⊖	⊖	⊕	⊕
DO	⊕	⊕	⊖	⊖
DP + DO	⊕	⊕	⊕	⊕
DP + DO + IO	⊕	⊕	⊕	⊕
DP: Dynamic Programming	DO: Direct Optimization	IO: Indirect Optimization		

Tabelle 1 Comparison and combination of the approaches

5 Receding Horizon Optimization

The receding horizon approach is the gist of *model-predictive control* (MPC, see e.g. [42]), which makes a numerical optimization practical for closed loop control. Therein, in each step t_k, the OCP is solved on a finite horizon T, which calculates the optimal open-loop trajectories $\bar{\boldsymbol{x}}^*(\tau)$ over $\tau \in [t_k, t_k + T]$, see Fig. 14. Only the first part of the optimal control $\bar{\boldsymbol{u}}^*(\tau)$ is implemented on Δt. Right in time the new solution is available of the OCP that has been shifted by Δt. In classical MPC, at each t_k the current plant state is fed back as the new initial state of the OCP. Altogether, this leads to a closed control-loop that anticipates future events, such as input and state saturation, and takes control actions accordingly.

This procedure is completely compatible with trajectory optimization for vehicles. Even more, its replanning mechanism can innately take the limited sensor range and predictability of the other traffic participants into account, which can fundamentally change the OCP from one optimization step to the other. Furthermore, the approach leaves additional degrees of freedom, which can be used to increase the overall robustness of the closed loop system. Firstly, the prediction model may not only include the plant but also the underlying fast low-level feedback or feedforward controllers, which are intended to simplify the resultant optimization model (1b). And secondly, the initial state of the OCP does not necessarily have to be the actual vehicle state but can also be the optimal trajectory of the last step sampled at the current time - or a combination of both. This works as long as the fed-back desired states of the optimal trajectory are tracked by an above-mentioned low-level feedback controller. However, as soon as components of the current vehicle state are used, that show up in the optimization constraints, so-called *slack variables* need to be introduced in the optimization. They transform the otherwise hard inequalities into *soft constraints* as they are referred to in the MPC lingo. This is required as otherwise the slightest disturbance or model uncertainty would ultimately lead to an initial state that the OCP with hard constraints does not have a solution for. MPC theory offers even more. It also deals with *stability* issues of the closed-loop system such as in [19]. More precisely, even for time-invariant system dynamics, constraints, and costs (typical of conventional control problems), the OCP solution changes over time due to the receding horizon, as indicated by Fig. 14. In the worst case the differences of the consecutive solutions build up and the system destabilizes. Well-established MPC-schemes therefore propose an augmentation of the cost functional and constraints, e.g., by a special terminal constraint (*invariant set* [3]) and cost (*control Lyapunov function* [19]) in order to guarantee stability. These schemes can also be transferred to the automotive application. Even more interesting for collision avoidance are the *permanent feasibility* guarantees, which can prevent the sub-optimal short-sighted solutions from "dead ends" also known as *inevitable collision states* (ICS, [15]) in robotics, but this is ongoing research (e.g. **which???**(Althoff et al. 2012),[32]).

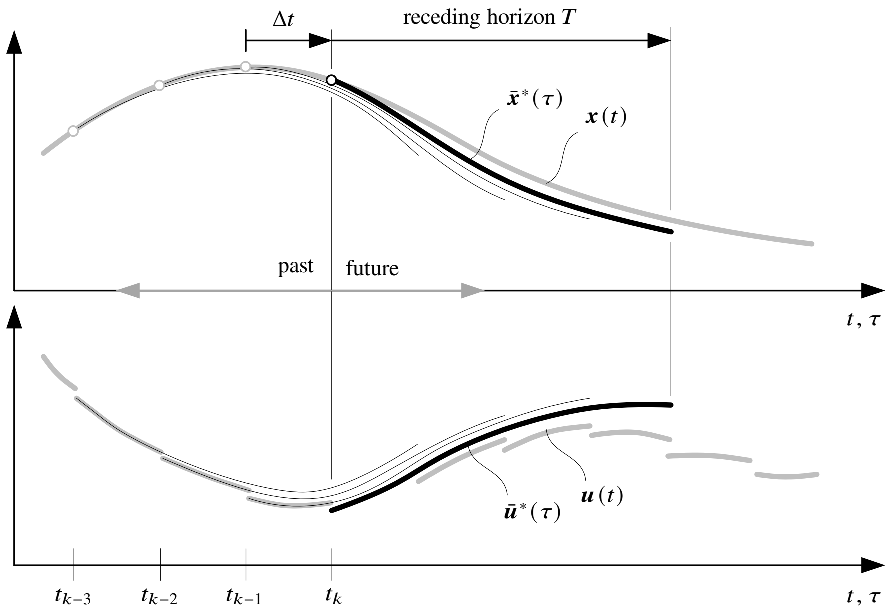

Abb. 13 Receding horizon optimization with optimization length T, cycle time Δt, current time step t_k, predicted and actual trajectory $\bar{\boldsymbol{x}}^*$, $\boldsymbol{x}$ predicted and actual system input $\bar{\boldsymbol{u}}^*$, $\boldsymbol{u}$

6 Conclusion

Owing to the increase in processing power, computationally intensive optimization algorithms can be executed in real-time in many industrial areas. As automotive electronic control units follow this trend, it is only a matter of time until receding horizon control techniques will emerge in production cars. As has been shown in this chapter, these techniques are most suitable for solving complex trajectory optimization tasks for novel driver assistance systems and automated driving. Based on well-known, elaborated principles we can combine different optimization techniques and implement fast, powerful algorithms with no need to reinvent the wheel. To surmount the complex integration of numerous safety and comfort functions to a complete driver-friendly unit, an integrated trajectory optimization module is required. Ideally, such a module covers the superset of all emerging use cases. Conventional functions such as emergency braking and lane keeping will then only be special cases within the algorithm.

Literaturverzeichnis

[1] Bellman, R. : *The theory of dynamic programming*. Rand Corporation Santa Monica CA, 1954 (RAND-P-550)

[2] Bertsekas, D. : *Dynamic programming and optimal control*. Athena Scientific Belmont, MA, 1995

[3] Blanchini, F. : Survey paper: Set invariance in control. In: *Automatica (Journal of IFAC)* 35 (1999), Nr. 11, S. 1747–1767

[4] Bock, H. ; Diehl, M. ; Leineweber, D. ; Schlöer, J. : A Direct Multiple Shooting Method for Real-Time Optimization of Nonlinear DAE Processes. In: Allgöwer, F. (Hrsg.) ; Zheng, A. (Hrsg.) ; Byrnes, C. (Hrsg.): *Nonlinear Model Predictive Control* Bd. 26. Birkhäuser Basel, 2000. – ISBN 978–3–0348–8407–5, S. 245–267

[5] Boissonnat, J.-D. ; Cerezo, A. ; Leblond, J. : A note on shortest paths in the plane subject to a constraint on the derivative of the curvature / INRIA. 1994 (2160). – Report

[6] Brandt, T. : *A predictive potential field concept for shared vehicle guidance*, Universität Paderborn, Diss., 2008

[7] Carvalho, A. ; Gao, Y. ; Gray, A. ; Tseng, H. ; Borrelli, F. : Predictive control of an autonomous ground vehicle using an iterative linearization approach. In: *Intelligent Transportation Systems-(ITSC), 2013 16th International IEEE Conference on*, 2013, S. 2335–2340

[8] Diehl, M. ; Bock, H. ; Diedam, H. ; Wieber, P. : Fast Direct Multiple Shooting Algorithms for Optimal Robot Control. In: *Fast Motions in Biomechanics and Robotics* 340 (2006), S. 65–93

[9] Dolgov, D. ; Thrun, S. ; Montemerlo, M. ; Diebel, J. : Path planning for autonomous vehicles in unknown semi-structured environments. In: *The International Journal of Robotics Research* 29 (2010), Nr. 5, S. 485–501

[10] Dubins, L. : On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal positions and tangents. In: *American Journal of Mathematics* 79 (1957), Nr. 3, S. 497–516

[11] Eichhorn, A. ; Werling, M. ; Zahn, P. ; Schramm, D. : Maneuver Prediction at Intersections using Cost-to-go Gradients. In: *International Conference on Intelligent Transportation Systems* IEEE, 2013

[12] Falcone, P. ; Borrelli, F. ; Asgari, J. ; Tseng, H. E. ; Hrovat, D. : Predictive Active Steering Control for Autonomous Vehicle Systems. In: *IEEE Transactions on Control Systems Technology* 15 (2007), Nr. 3, S. 566–580. – ISSN 1063–6536

[13] Ferguson, D. ; Howard, T. ; Likhachev, M. : Motion Planning in Urban Environments: Part I. In: *International Conference on Intelligent Robots and Systems* IEEE/RSJ, 2008, S. 1063–1069

[14] Ferguson, D. ; Howard, T. ; Likhachev, M. : Motion Planning in Urban Environments: Part II. In: *International Conference on Intelligent Robots and Systems* IEEE/RSJ, 2008, S. 1070–1076

[15] Fraichard, T. : A short paper about motion safety. In: *International Conference on Robotics and Automation* IEEE, 2007, S. 1140–1145

[16] Gayko, J. : Lane Keeping Support. In: Winner, H. (Hrsg.) ; Hakuli, S. (Hrsg.) ; Wolf, G. (Hrsg.): *Handbuch Fahrerassistenzsysteme*. Wiesbaden : Vieweg + Teubner Verlag, 2012, S. 554–561

[17] Gerdts, M. ; Karrenberg, S. ; Müller-Bessler, B. ; Stock, G. : Generating locally optimal trajectories for an automatically driven car. In: *Optimization and Engineering* 10 (2009), Nr. 4, S. 439–463

[18] Graichen, K. : Nichtlineare modellprädiktive Regelung basierend auf Fixpunktiterationen. In: *at - Automatisierungstechnik* 60 (2012), Nr. 8, S. 442–451

[19] Grüne, L. ; Pannek, J. : *Nonlinear Model Predictive Control: Theory and Algorithms*. Springer http://books.google.de/books?id=kBnz6OFxO8wC. – ISBN 9780857295002

[20] Gu, T. ; Dolan, J. : On-Road motion planning for autonomous vehicles. In: *Intelligent Robotics and Applications*. Springer, 2012, S. 588–597

[21] Gu, T. ; Snider, J. ; Dolan, J. M. ; Lee, J. : Focused Trajectory Planning for autonomous on-road driving. In: *Intelligent Vehicles Symposium, 2013 IEEE* IEEE, 2013, S. 547–552

[22] Gutjahr, B. ; Gröll, L. ; Werling, M. : Lateral vehicle trajectory optimization using constrained linear time-varying MPC. In: *IEEE Transactions on Intelligent Transportation Systems* 18 (2016), Nr. 6, S. 1586–1595

[23] Gutjahr, B. ; Werling, M. : Automatic collision avoidance during parking and maneuvering - An optimal control approach. In: *Intelligent Vehicles Symposium Proceedings, 2014 IEEE* IEEE, 2014, S. 636–641

[24] Hargraves, C. R. ; Paris, S. W.: Direct trajectory optimization using nonlinear programming and collocation. In: *Journal of guidance, control, and dynamics* 10 (1987), Nr. 4, S. 338–342

[25] Howard, T. ; Kelly, A. : Optimal rough terrain trajectory generation for wheeled mobile robots. In: *The International Journal of Robotics Research* 26 (2007), Nr. 2, S. 141–166

[26] Kang, K. : *Online optimal obstacle avoidance for rotary-wing autonomous unmanned aerial vehicles*, Diss., 2012

[27] Kelly, A. ; Nagy, B. : Reactive nonholonomic trajectory generation via parametric optimal control. In: *The International Journal of Robotics Research* 22 (2003), Nr. 7-8, S. 583–601

[28] Koren, Y. ; Borenstein, J. : Potential field methods and their inherent limitations for mobile robot navigation. In: *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on* IEEE, 1991, S. 1398–1404

[29] Krogh, B. : A generalized potential field approach to obstacle avoidance control. In: *International Robotics Research Conference*. Bethlehem, Pennsylvania, 1984

[30] Latombe, J. : *Robot Motion Planning*. Springer Verlag, 1990. – ISBN 0792391292

[31] LaValle, S. : *Planning Algorithms*. Cambridge University Press, 2006

[32] Lawitzky, A. ; Nicklas, A. ; Wollherr, D. ; M., B. : Determining States of Inevitable Collision using Reachability Analysis. In: *International Conference on Intelligent Robots and Systems* IEEE, 2014, S. 4142–4147

[33] Lewis, F. ; Syrmos, V. : *Optimal Control*. John Wiley & Sons, Inc. http://de.scribd.com/doc/49575744/Opti-Control-Lewis

[34] Likhachev, M. ; Ferguson, D. : Planning long dynamically feasible maneuvers for autonomous vehicles. In: *The International Journal of Robotics Research* 28 (2009), Nr. 8, S. 933–945

[35] McNaughton, M. : *Parallel Algorithms for Real-time Motion Planning*, Carnegie Mellon University, Diss., 2011

[36] McNaughton, M. ; Urmson, C. ; Dolan, J. ; Lee, J. : Motion Planning for Autonomous Driving with a Conformal Spatiotemporal Lattice. In: *International Conference on Robotics and Automation* IEEE, 2011, S. 4889–4895

[37] Montemerlo, M. ; Becker, J. ; Bhat, S. ; Dahlkamp, H. ; Dolgov, D. ; Ettinger, S. ; Haehnel, D. ; Hilden, T. ; Hoffmann, G. ; Huhnke, B. u. a.: Junior: The Stanford entry in the Urban Challenge. In: *Journal of Field Robotics* 25 (2008), Nr. 9

[38] Nocedal, J. ; Wright, S. : *Numerical Optimization*. Springer Science + Business Media, 2006

[39] Papageorgiou, M. : *Optimierung: Statische, dynamische, stochastische Verfahren*. Springer, 2012

[40] Park, J. ; Kim, D. ; Yoon, Y. ; Kim, H. ; Yi, K. : Obstacle avoidance of autonomous vehicles based on model predictive control. In: *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering* 223 (2009), Nr. 12, S. 1499–1516

[41] Pivtoraiko, M. ; Knepper, R. ; Kelly, A. : Differentially constrained mobile robot motion planning in state lattices. In: *Journal of Field Robotics* 26 (2009), Nr. 3, S. 308–333

[42] Rawlings, J. : Tutorial overview of model predictive control. In: *Control Systems, IEEE* 20 (2000), Nr. 3, S. 38–52

[43] Reeds, J. ; Shepp, L. : Optimal paths for a car that goes both forwards and backwards. In: *Pacific Journal of Mathematics* 145 (1990), Nr. 2, S. 367–393

[44] Rouchon, P. ; Fliess, M. ; Lévine, J. ; Martin, P. : Flatness, motion planning and trailer systems. In: *Conference on Decision and Control* IEEE, 1993, S. 2700–2705

[45] Turri, V. ; Carvalho, A. ; Tseng, H. ; Johansson, K. H. ; Borrelli, F. : Linear Model Predictive Control for Lane Keeping and Obstacle Avoidance on Low Curvature Roads. In: *2013 16th International IEEE Conference on Intelligent Transportation Systems: Intelligent Transportation Systems for All Modes, ITSC 2013; The Hague; Netherlands; 6 October 2013 through 9 October 2013* IEEE, 2013, S. 378–383

[46] Werling, M. ; Liccardo, D. : Automatic collision avoidance using model-predictive online optimization. In: *Conference on Decision and Control* IEEE, 2012, S. 6309–6314

[47] Werling, M. ; Ziegler, J. ; Kammel, S. ; Thrun, S. : Optimal Trajectory Generation for Dynamic Street Scenarios in a Frenet Frame. In: *International Conference on Robotics and Automation* IEEE, 2010, S. 987–993

[48] Werling, M. : *Optimale aktive Fahreingriffe: für Sicherheits-und Komfortsysteme in Fahrzeugen*. Walter de Gruyter GmbH & Co KG, 2017

[49] Yoon, Y. ; Shin, J. ; Kim, H. ; Park, Y. ; Sastry, S. : Model-predictive active steering and obstacle avoidance for autonomous ground vehicles. In: *Control Engineering Practice* 17 (2009), Nr. 7, S. 741–750

[50] Ziegler, J. ; Stiller, C. : Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios. In: *International Conference on Intelligent Robots and Systems* IEEE/RSJ, 2009, S. 1879–1884

[51] Ziegler, J. ; Stiller, C. : Fast Collision Checking for Intelligent Vehicle Motion Planning. In: *Intelligent Vehicles Symposium* IEEE, 2010, S. 518–522

[52] Ziegler, J. ; Werling, M. ; Schröder, J. : Navigating car-like robots in unstructured environments using an obstacle sensitive cost function. In: *Intelligent Vehicles Symposium* IEEE, 2008, S. 787–791

[53] Ziegler, J. ; Bender, P. ; Dang, T. ; Stiller, C. : Trajectory planning for Bertha - A local, continuous method. In: *Intelligent Vehicles Symposium Proceedings, 2014 IEEE* IEEE, 2014, S. 450–457