Designing Maneuver Automata of Motion Primitives for Optimal Cooperative Trajectory Planning



Matheus V. A. Pedrosa, Patrick Scheffe, Bassam Alrifaee, and Kathrin Flaßkamp

Abstract Trajectory planning techniques form a central step to enable autonomous driving. The motion primitives method generates an automaton of precomputed maneuvers with structure-exploiting properties. Thereby, the trajectory planning problem can be reduced to finding an admissible/optimal sequence of motion primitives. In this chapter, we present ways to designing maneuver automata based on different system models and on either analytical or data-based approaches for automaton generation. Moreover, numerical methods for computing optimal maneuvers are listed and we discuss graph-based planning techniques. A subsequent chapter shows the evaluation of motion primitives automata in the Cyber-Physical Mobility Lab.

1 Introduction

The task of planning trajectories for multiple vehicles can be solved by many available techniques (see, for instance, [2, 3, 17]). However, there are still key challenges to be tackled for a multi-vehicle trajectories planner: (a) the admissibility of the planned trajectories, (b) the real-time capability of the optimization solvers on the respective vehicles and (c) the feasibility of the communication overhead between the vehicles.

M. V. A. Pedrosa (✉) · K. Flaßkamp
Chair of Systems Modeling and Simulation, Systems Engineering, Saarland University, Saarbrücken, Germany
e-mail: matheus.pedrosa@uni-saarland.de

K. Flaßkamp
e-mail: kathrin.flasskamp@uni-saarland.de

P. Scheffe
Chair for Embedded Software, RWTH Aachen University, Aachen, Germany
e-mail: scheffe@embedded.rwth-aachen.de

B. Alrifaee
Department of Aerospace Engineering, University of the Bundeswehr Munich, Neubiberg, Germany
e-mail: bassam.alrifaee@unibw.de

© The Author(s) 2024

C. Stiller et al. (eds.), *Cooperatively Interacting Vehicles*,
https://doi.org/10.1007/978-3-031-60494-2_8

Complemented into two chapters,[1] our work brings new methodologies for solving the cooperative trajectory planning problem for autonomous driving. We address the challenges mentioned above through graph-based optimal solutions. Before going into more detail about which methods we use and how we use them, let us first give an overview of how it is contextualized within vehicle automation.

Automated driving systems basically consist of three modules [31]:

1. Sensing or perception: capture the environment objects and conditions through sensors.
2. Planning: find a feasible trajectory.
3. Acting or control: track the trajectory by controlling the vehicle's actuators.

We place our focus on solving the second step. Motion planning aims to find a sequence of control inputs to move a vehicle from an original state to a set of possible goal states, while avoiding collisions during the trajectory [30]. At first, this task can be achieved by solving an optimal control problem (OCP). However, it could be computationally costly to get optimal control solutions when dealing with nonlinear vehicle models. Complex environments can also make it difficult to properly design all the obstacles into the optimization problem, which make the OCP unsuitable for many applications [14]. As an alternative, discrete planning techniques sample the state space, map it as a graph and perform a graph search for a minimum-cost path [22]. As disadvantages, we can cite the total neglect of the model in the case of the most famous graph search, the A*, or the numerically complex and non-time-critical solutions for the also well-known Hybrid A* search [10, 23, 26].

In order to get the best of both worlds, i.e., decreasing the motion planning problem complexity and avoiding a full discretization over the state space, we use the concept of motion primitives, originally proposed by [14]. Motion primitives are finite-time pieces of trajectories that can be concatenated. They are constructed from the dynamical system model. That is, the final path resulting from their interlocks is feasible with respect to the selected model. References [12, 14, 19] showed that, by using them, the highly complex problem of trajectory planning can be transformed into a graph search, in which solutions can be found with a suitable difficulty. However, for this to happen, it is of fundamental importance to have a library of primitives at hand that ensures appropriate routes for the desired road scenarios. At the same time, it should also have a size that makes the problem as computationally inexpensive as possible. Note that all this should also take into account the cooperative communication between agents, since it is desirable to have a trajectory planning with a sufficiently small communication effort.

The realization of motion primitives is only possible when the dynamic model has the symmetry property. To give an intuition, this property indicates that it is possible to perform rotations and translations in mechanical systems—without deformation of their path profile under the same sequence of control inputs. In the original work [14], the symmetry property of systems was exploited to develop two special kinds of

[1] Trajectory planning strategies for multiple vehicles are presented in the Chapter "Prioritized Trajectory Planning for Networked Vehicles Using Motion Primitives".

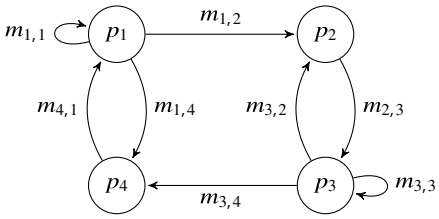


Fig. 1 Example of a maneuver automaton with four trim primitives: $\{p_1, p_2, p_3, p_4\}$ and eight maneuvers: $\{m_{1,1}, m_{1,2}, m_{1,4}, m_{2,3}, m_{3,2}, m_{3,3}, m_{3,4}, m_{4,1}\}$ (figure from [24])

motion primitives: the trim primitives and the maneuvers. Trims are steady motions, where the control inputs are kept fixed, while the maneuvers are motions transitioning between the steady motions. Their rules for concatenation can be translated into a directed graph, which we call motion primitive automaton (MPA), also referred in [14] as maneuver automaton. Figure 1 illustrates an example of an MPA with four trims and eight maneuvers. Then, solving the motion planning problem consists of using a graph search method to find a sequence of primitives, which can be concatenated according to the MPA.

In this chapter, with the first part of our studies, we present the development of methods to architect the motion primitives selection and construction, as well as the relationship between them. The second part, written in the chapter "Prioritized Trajectory Planning for Networked Vehicles Using Motion Primitives", is devoted to detailing the cooperative trajectory planning algorithms that represent maneuvers primitives. The general workflow is given in Fig. 2.

This chapter is organized as follows. In Sect. 2, we evaluate, from a list of different vehicle models, the suitable dynamics for the planning problem and determine the symmetry group for a generic class of vehicle models. In Sect. 3, based on previous works, e.g., [12–14, 21, 23], we determine a method to analytically select trim primitives from a vehicle model and, alternatively, abstract typical trim primitives from traffic data. In Sect. 4, we model the computation of maneuvers as an OCP and solve the respective OCP to obtain the optimal maneuvers. Automata of different configurations with respect to their computational complexity and solution quality are analyzed in Sect. 5. It also investigates both time-optimal and maximum comfort motion graphs via the analysis of multi-objective maneuvers. In Sect. 6, we briefly present possible algorithms to solve the graph-based planning problem. Lastly, we give concluding remarks in Sect. 7.

2 Models and Symmetry

There are several ways to represent the dynamic system of vehicles, from the simplest cases, such as the point-mass model, the Dubins curves [11] and the Reeds–Shepp curves [25], to detailed, vehicle-specific models. Both Dubins and Reeds–Shepp curves take into account a kinematic car model consisting only of the pose, i.e., the position and orientation. Halfway through, the CommonRoad benchmarks present a

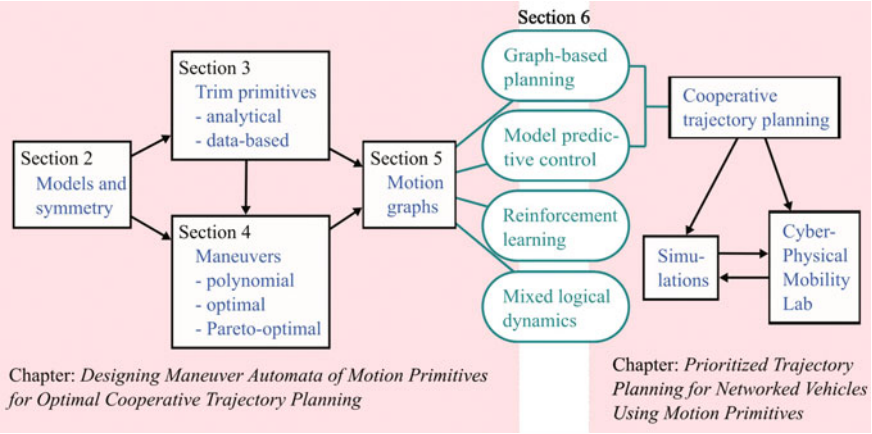


Fig. 2 Workflow of our methodology

hierarchical list of models that considers increasingly complex lateral vehicle dynamics and tire models [5]. This list includes, among others, the following models: kinematic single-track model, single-track model, and a multi-body model. It is assumed for these models the existence of controllers that can realize a commanded acceleration. The choice of the appropriate model depends on which detail you want to capture the physics of motion. In the Appendix, the reader can find the description of the equations for the kinematic single-track and the single-track models, as they will be used in this chapter.

All of the CommonRoad models have in common the following generic structure of ordinary differential equations:

$$\dot{x} = f(x,u) := \begin{bmatrix} f_1(r,u)\cos\left(f_2(r,u)+\psi\right) \\ f_1(r,u)\sin\left(f_2(r,u)+\psi\right) \\ f_\psi(r,u) \\ f_r(r,u) \end{bmatrix}, \tag{1}$$

with the vector of states $x = \begin{bmatrix} s_x & s_y & \psi & r \end{bmatrix}^{\mathrm{T}}$ belonging to a manifold $\mathcal{X}$, where s_x and s_y are the positions of the center of gravity, ψ is the vehicle orientation, r is any vector of $n-3$ states, $u \in \mathcal{U}$ is the vector of inputs and $f_1(r,u)$, $f_2(r,u)$, $f_\psi(r,u)$, and $f_r(r,u)$ are arbitrary nonlinear functions. For convenience, we omit the notation for dependence of $x(t)$ and $u(t)$ on time $t \in \mathbb{R}_{\geq 0}$. We assume the function $f(x,u)$ of Eq. (1) as being continuous and locally Lipschitz w.r.t. $x(t)$. Then, we guarantee the existence and uniqueness of solutions given by the flow

$$x(t) = \varphi_u(x(0), t) \tag{2}$$

for a given input function u on the time interval $t \in [0, T]$.

Many mechanical systems, including vehicles, exhibit the symmetry property, which acts as state transformations defined by Lie group representations [14]. They are necessary to build the primitives from the model. To describe them mathematically, we need to introduce some considerations.

Let the Lie group be denoted by $\mathcal{G}$, its identity element by e, and its left action on $\mathcal{X}$ by $\Psi : \mathcal{G} \times \mathcal{X} \to \mathcal{X}$ with Ψ smooth, $\Psi(e, x) = x$ for $x \in \mathcal{X}$, and $\Psi(g, \Psi(h, x)) = \Psi(gh, x)$ for all $g, h \in \mathcal{G}$ and $x \in \mathcal{X}$.

Definition 1 (*Symmetry*) The tuple $(\mathcal{G}, \Psi)$ is a symmetry for $\dot{x} = f(x, u)$ on $\mathcal{X}$, if for any fixed control $u \in \mathcal{L}^{\infty}_{\text{loc}}([0, \infty), \mathbb{R}^m)$,

$$\varphi_u(\Psi(g, x_0), t) = \Psi(g, \varphi_u(x_0, t)) \tag{3}$$

holds for all $g \in \mathcal{G}$, $x_0 \in \mathcal{X}$, and $t \geq 0$.

We can produce a symmetry group that fits the entire set of models described in [5]. It is given by combined rotations and translations on the pose, which we represent by $p = \begin{bmatrix} s_x & s_y & \psi \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^2 \times S^1$, in the following form [24]:

Theorem 1 *The symmetry group for Eq. (1) is given by*

$$\mathcal{G} := \left\{ g \in \mathsf{SE}(n) : g := g(\Delta x) = \begin{bmatrix} R & \Delta x \\ 0 & 1 \end{bmatrix} \right\}, \tag{4}$$

where

$$R = \begin{bmatrix} R_{\mathsf{SO}(3)} & 0 \\ 0 & I \end{bmatrix} \in \mathsf{SO}(n), \tag{5}$$

$$\Delta x = \begin{bmatrix} \Delta s_x \\ \Delta s_y \\ \Delta \psi \\ 0 \end{bmatrix} \in \mathbb{R}^2 \times S^1 \times \{0\}^{n-3}, \tag{6}$$

$$R_{\mathsf{SO}(3)} = \begin{bmatrix} \cos(\Delta\psi) & -\sin(\Delta\psi) & 0 \\ \sin(\Delta\psi) & \cos(\Delta\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \in \mathsf{SO}(3), \tag{7}$$

for I being the identity matrix with appropriate dimension, a vector Δx, and g given in homogeneous coordinates, such that the affine-linear group action can be represented by:

$$\Psi_g(x) = Rx + \Delta x. \tag{8}$$

To prove it, we show the equivariance of the system (1) w.r.t. the symmetry action (8). We will show the idea of the proof, while details can be found in [24].

Proof The vector field f is equivariant w.r.t. the symmetry action Ψ if

$$f(\Psi_g(x), u) = \frac{\mathrm{d}\Psi_g(x)}{\mathrm{d}x} \cdot f(x, u). \tag{9}$$

Let $\Delta p = \begin{bmatrix} \Delta s_x & \Delta s_y & \Delta\psi \end{bmatrix}^{\mathrm{T}}$. The group action (8) can be written as

$$\Psi_g(x) = \begin{bmatrix} R_{\mathrm{SO}(3)}\, p + \Delta p \\ r \end{bmatrix} = \begin{bmatrix} \cos(\Delta p)s_x - \sin(\Delta p)s_y + \Delta s_x \\ \sin(\Delta p)s_x + \cos(\Delta p)s_y + \Delta s_y \\ \psi + \Delta\psi \\ r \end{bmatrix}. \tag{10}$$

Then, from Eqs. (1), (10) and (5), we get that the left-hand side of Eq. (9) is:

$$f\left(\Psi_g(x), u\right) = Rf(x, u). \tag{11}$$

Considering $\Psi_g(x) = Rx + \Delta x$ as in Eq. (8),

$$\frac{\mathrm{d}\Psi_g(x)}{\mathrm{d}x} = R, \tag{12}$$

which we can replace in Eq. (11), proving the equivariance of the vector field by satisfying Eq. (9).

Given the proper considerations, we can now define motion primitives as equivalence classes of trajectories.

Definition 2 (*Motion primitive*) A motion primitive is the equivalence class of a representing pair (x, u) on $[t_i, t_f]$, if for any class member $(\bar{x}, \bar{u})$ on $[\bar{t}_i, \bar{t}_f]$, it holds that $t_f - t_i = \bar{t}_f - \bar{t}_i$ and there exists a group element $g \in \mathcal{G}$ and a shift $\Delta t \in \mathbb{R}$, such that

$$(x(t), u(t)) = (\Psi(g, \bar{x}(t - \Delta t)), \bar{u}(t - \Delta t)) \quad \forall t \in [t_i, t_f]. \tag{13}$$

In the next sections, we will introduce the two types of motion primitives: trim primitives and maneuvers.

3 Trim Primitives

These primitives are characterized by fixed, i.e., trimmed, controls and are symmetry-induced motions. They were introduced in [14], and the authors add that the trims are identified with steady-state motions, also known as relative equilibria of the system. Formally, they can be defined as follows.

Definition 3 (*Trim Primitive*) Following the Definition 1, let $\mathfrak{g}$ denote the Lie algebra of $\mathcal{G}$ with the exponential map $\exp : \mathfrak{g} \to \mathcal{G}$, and $\bar{u} \in \mathcal{U}$ a fixed control input. The

tuple (x, u) on $[0, T]$ with $x(0) = x_0$ is called a trim primitive if it is a solution to the system dynamics expressed, for all $t \in [0, T]$, by

$$\begin{cases} x(t) = \Psi(\exp(\xi t), x_0), \\ u(t) \equiv \bar{u}, \end{cases} \tag{14}$$

with $\xi \in \mathfrak{g}$ being a suitable chosen Lie algebra element.

The duration of a trim primitive is, in principle, not fixed and is called "coasting time". For the kinematic single-track or the single-track models, the trims are characterized by a fixed velocity and a constant curvature[2] (see [23, 24]).

A choice for a finite number of trim primitives has to be taken. The question of representation and well-spread trims arises. A "plain vanilla" approach is to uniformly grid the Lie algebra up until borders that seem physically plausible [19]. More sophisticated approaches choose representative trim primitives based on data, either the road-geometry of interest or from driving, as detailed in the following two subsections.

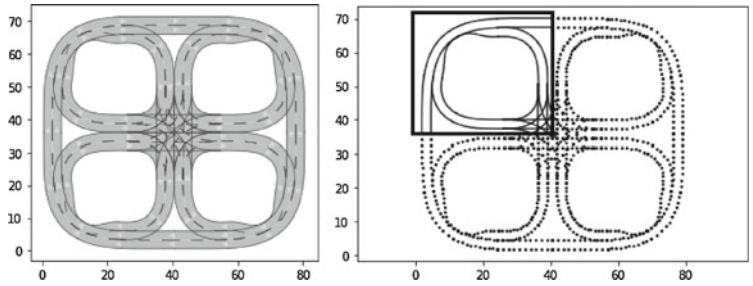
3.1 Choice Based on Road-Geometry

One way to select the trim primitives is to fit them to the geometry of the roads on which the vehicle is to drive. As an example, we take the map of the Cyber-Physical Mobility Lab (CPM Lab) [16] drawn using the CommonRoad interface [5], depicted in Fig. 3a. From information contained in the CommonRoad scenario file, we can decompose the roads into the discrete points taken from the center of each lane, as can be seen in the upper left of the Fig. 3b and generate trims by the sequence:

1. Calculate all the possible curvatures from the map.
2. Select the most frequent curvatures.
3. Choose an arbitrary set of speeds within the boundaries interval.
4. Combine curvatures and speeds in tuples that represent each trim.

The yaw angles at each point of the decomposed map could be calculated from the vectors tangent to the lane's center (see Fig. 3b). Then, we can get and store the set of different curvatures, which might be a large data set. To reduce it, we can cluster the data points, for instance via k-means, to get a smaller number of the representative curvatures [24]. However, we can directly steer the number of different curvatures by the considered rounding accuracy, as we can have many similar data points. For example, the set of curvatures $\{0.0507..., 0.0543..., 0.0539...\}$ could be reduced to the values $\{0.051, 0.054\}$ when considering three decimal places, or to just to the value of 0.05 if two decimal places are considered. See Fig. 4 for checking the number

[2] The curvature is calculated by dividing the yaw rate by the velocity. Then, the trims could, alternatively, be represented by a constant speed and constant yaw rate.



(a) CPM Lab's map

(b) Lane's centers (upper left) with their discrete decomposition

Fig. 3 Road geometry decomposition of the CPM Lab's map into 208 different points. The axes are the coordinates in meters

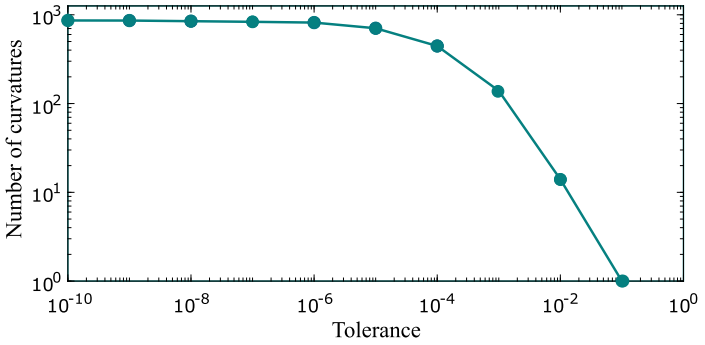


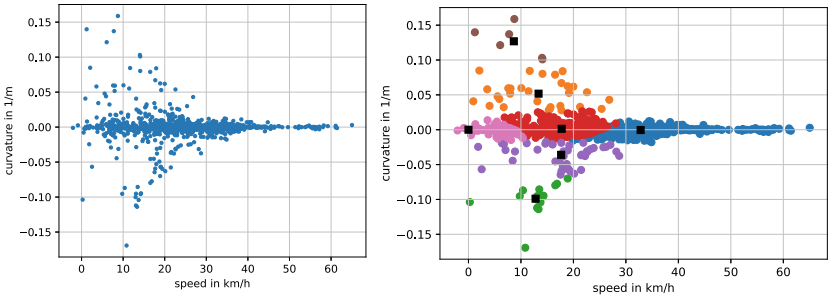
Fig. 4 The number of different curvatures computed in the example according to the tolerated decimal places

of different curvatures considered according to the accepted decimal places for this example. Having two decimal places, we get 14 classes of curvatures, that can be representative for this map. Lastly, a set of arbitrary speeds can be combined with these different classes and we get a set of trims to be used in the planning problem.

3.2 Choice Based on Driving Data

An automatic generation of data-based automata was proposed in [24]. The authors assumed that the data represents a dynamical model with symmetries. Also, this model is observable such that the full system state could be reconstructed from the available states on the data. Then, making an assumption on the model, the following sequence of steps was carried out:

1. Find invariances of trims in data.
2. Cluster trim primitives.



(a) Extracted trims (b) Trims clustered

Fig. 5 Trims clustered using k-means in seven representative points, where the black squares are the centers of each cluster (figure from [24])

3. Evaluate a transition matrix.

The selected data in [24] was taken from the nuScenes data set [8], having multiple information about the vehicle's states, including the pose, the velocity, the acceleration, and the rotation rate recorded using an inertial measurement unit during urban driving in Singapore (Singapore) and Boston (United States). It is worth mentioning that this data set represents the interaction of the car in real traffic with other vehicles, including overtaking, braking, waiting on corners, etc. That is, several real interactions between vehicles are embedded in the selected primitives, ideal for a cooperative planning scenario.

Consider the data points being represented by the triples (t_i, x_i, u_i) for $i = 0, 1, 2, \ldots, D$, where $D \in \mathbb{R}$ is the number of elements in the data set. Consider that, for a suitable chosen symmetry $(\mathcal{G}, \Psi)$, there exist solutions (x, u) satisfying Definition 3 for a model $\dot{x} = f(x, u)$. Then, subsequent data points belong to the same trim if

$$\begin{cases} ||u_{i+1} - u_i|| < \epsilon_u, \text{ and} \\ ||\Psi(\exp(\xi(t_{i+1} - t_i)), x_i) - x_{i+1}|| < \epsilon_x \end{cases} \tag{15}$$

for a sufficient small positive error margins ϵ_x and ϵ_u.

We can determine a minimum time length τ for the duration of a trim primitive, i.e., a minimal coasting time. Then, a trim will be considered only if, for a sequence of $N > 1$ points, the conditions (15) hold from i to $i + N - 1$ such that $t_{i+N} - t_i \geq \tau$.

However, the number of extracted trims from the data can be huge. Then, we need to look for a finite amount of clusters that define the most representative trims during a route in real traffic. In [24], they worked with the k-means algorithm, an unsupervised learning technique that finds clusters in a set of data points, where the amount of clusters is given [20]. The representative trims will be selected as the center points of each cluster. Figure 5 shows an example of trims being clustered for the kinematic single-track model (28) from [24]. The trims are represented by a constant speed (x-axis) and a constant curvature (y-axis).

At first, we could consider that a vehicle would be able to transit from any relative equilibrium to any other. For instance, in a kinematic robot model under nonholonomic constraints, given by

$$\begin{bmatrix} \dot{s}_x(t) \\ \dot{s}_y(t) \\ \dot{\psi}(t) \end{bmatrix} = \begin{bmatrix} \cos(\psi(t)) \\ \sin(\psi(t)) \\ 0 \end{bmatrix} u_1(t) + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u_2(t), \tag{16}$$

where the states are the pose and the controller manages the linear velocity $u_1(t) = v(t)$ and the angular velocity $u_2(t) = \dot{\psi}(t)$, trajectories can switch directly from one trim to another [13]. This is due to this model directly controlling the velocities and, thus, allowing discontinuities thereof. In this case, as well as in the (kinematic) single-track model, every constant control input defines a trim, either going straight with constant velocity or going in a circle with constant rotational velocity. However, in the (kinematic) single-track model, the control inputs correspond to the longitudinal acceleration and the steering angle velocity (see the Appendix for these models' equations). Thus, trims necessarily correspond to uncontrolled, i.e., constant-velocity motion. Smooth transitions between trims are then needed for, e.g., accelerating and decelerating to a new cruising speed, or for transitioning between straight and circular motions.

We can search and select these transitions according to their occurrence in the data. That is, only transitions with a high probability of occurrence will be considered. The probabilities are organized in a transition matrix, in which, for each trim cluster, the transitions from all points of this cluster to other clusters are counted in the data.

These transitions are another kind of primitive, called "maneuvers". The last step for the automatic generation of an automaton is the computation of the maneuvers. Their formal definition, as well as techniques to compute them, will be given in the next section.

4 Maneuvers

The second type of motion primitives is the maneuvers. They are responsible for smooth transitions in the system from one trim primitive to another. Formally, we can define them as follows.

Definition 4 (*Maneuver*) A maneuver is a finite-time trajectory that connects two trim primitives and is identified by:

- a time duration T;
- a sequence of control actions $u : [0, T] \to \mathbb{R}^m$;
- and an evolution in the form of (2) such that $(x(0), u(0))$ and $(x(T), u(T))$ belong to trim primitives characterized by (14).

In the class of vehicle models, the physics of maneuvers depends on the specific choice of the dynamical system model.

To derive maneuvers for the considered family of vehicle models, we present a geometric approach, in which polynomial equations define the transitions from the predecessor trim to the successor one. Alternatively, maneuvers can be computed as solutions of an OCP. In this case, we can also explore Pareto fronts in a multiobjective optimization problem.

4.1 Polynomial Approach

The paper [23] exemplifies a concrete case of formulating the geometric method using the single-track model (30) from [5]. In this case, a smooth transition needs to be made between the velocities v and steering angles δ from the predecessor trim to the successor one, both having these parameters fixed. Then, for a maneuver with the duration $T > 0$, we have the constraints $v(0) = v_0$ and $v(T) = v_T$. A jump in acceleration at the beginning or the end of the maneuver would theoretically result in infinite jerk, which can be avoided by setting $u(0) = u(T) = \begin{bmatrix} 0 & 0 \end{bmatrix}^{\mathrm{T}}$. Then, the control inputs are continuous, but we have additional constraints on the velocity $\dot{v}(0) = \dot{v}(T) = 0$. These constraints are met by the following cubic polynomial transitions for $0 \leq t \leq T$:

$$\begin{cases} v(t) = (v_T - v_0)\left(3 - 2\dfrac{t}{T}\right)\left(\dfrac{t}{T}\right)^2 + v_0, \\ \delta(t) = (\delta_T - \delta_0)\left(3 - 2\dfrac{t}{T}\right)\left(\dfrac{t}{T}\right)^2 + \delta_0. \end{cases} \tag{17}$$

Then, the corresponding control signals $u_{\dot{v}}$ and $u_{\dot{\delta}}(t)$ are

$$\begin{cases} u_{\dot{v}}(t) = 6(v_T - v_0)\left(1 - \dfrac{t}{T}\right)\dfrac{t}{T^2}, \\ u_{\dot{\delta}}(t) = 6(\delta_T - \delta_0)\left(1 - \dfrac{t}{T}\right)\dfrac{t}{T^2}. \end{cases} \tag{18}$$

In addition, to ensure the feasibility of the maneuver, constraints on the longitudinal acceleration and the derivative of the steering angle need to be considered. For the selected model, there exist the constraints

$$|\dot{v}| \leq \left|\frac{3}{2}\frac{v_T - v_0}{T}\right| \text{ and } |\dot{\delta}| \leq \left|\frac{3}{2}\frac{\delta_T - \delta_0}{T}\right|. \tag{19}$$

When the maneuvers have positive acceleration (i.e., $v_0 < v_T$), another constraint needs to be considered:

$$u_{\dot{v}} \leq a_{\max} \frac{v_s}{v}, \tag{20}$$

with the switching velocity v_s, representing limited engine power, and a maximal longitudinal acceleration $a_{\max} > 0$. Then, the duration of the maneuver can be chosen according to

$$\begin{cases} T = \max\left(\frac{3}{2}\frac{|v_T - v_0|}{a_{\max}}, \frac{3}{2}\frac{|\delta_T - \delta_0|}{\dot{\delta}_{\max}}, \frac{3}{2}\frac{(v_T - v_0)v_T}{a_{\max} v_s}, T_{\min}\right), \\ \quad \text{for } v_0 < v_T, \\ T = \max\left(\frac{3}{2}\frac{|v_T - v_0|}{a_{\max}}, \frac{3}{2}\frac{|\delta_T - \delta_0|}{\dot{\delta}_{\max}}, T_{\min}\right), \text{ otherwise.} \end{cases} \tag{21}$$

where $T_{\min}$ is a defined shortest duration, set as a design choice.

4.2 Optimal and Pareto-Optimal Maneuvers

Alternatively, the maneuvers can be computed optimally with respect to a cost functional $J(T, x, u)$, for a duration T. Then, each maneuver is obtained by solving the following OCP:

$$\begin{aligned} \underset{T,x,u}{\text{minimize}} \quad & J(T, x, u) && \text{(22a)} \\ \text{subject to} \quad & \dot{x}(t) = f(x(t), u(t)), \ 0 < t \leq T && \text{(22b)} \\ & 0 \geq g(x(t), u(t)), \ 0 < t \leq T && \text{(22c)} \\ & x(0) = x_0 && \text{(22d)} \\ & x(T) = x_T, && \text{(22e)} \end{aligned}$$

with x_0 and x_T as fixed states[3] evaluated at the predecessor and successor trims, respectively, and $g(\cdot)$ as the constraints for the states and inputs.

In the case of multiple cost functionals to be considered, the problem (22a) becomes a multiobjective optimal control problem. Then, we can select a Pareto-optimal maneuver by computing the so-called Pareto set of optimal compromises between the concurrent objectives [9] and choosing one of its points (see Fig. 6).

For instance, consider the kinematic single-track model (28), the costs $J_1 = T$ and $J_2 = \int_0^T ||u_{\dot{v}}||_2^2$, for a trade-off between fast and comfortable trajectories. The maneuver goes from a trim described by $(v, \delta) = (0\,\mathrm{km\,h^{-1}}, 0°)$ to a trim $(20\,\mathrm{km\,h^{-1}}, 15°)$ and it is limited by 5 s. The Pareto front with 25 points is given in Fig. 6 together with their respective pose and inputs. Optimal control problems can be solved using numerical software tools, for instance CasADi [6] or TransWORHP [18]. We can

[3] Depending on the dynamical system, only part, not all, of the states x could be considered as fixed at the initial and final times of the maneuver.

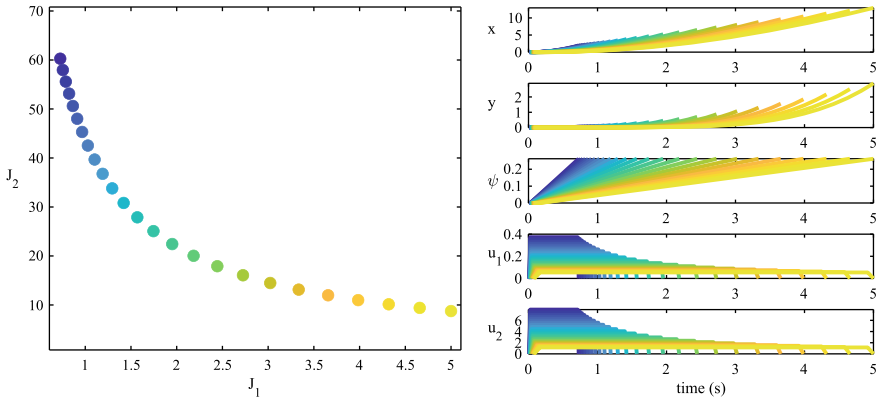


Fig. 6 Example of a Pareto front for a maneuver with $J_1 = T$ and $J_2 = \int_0^T ||u_{\dot{v}}||_2^2$

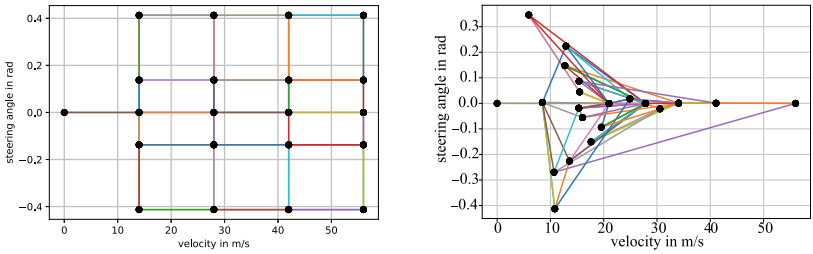
select a Pareto-optimal point based on a decision-making, to get the maneuver to be considered in the MPA.

5 Maneuver Automaton Selection

In [14], motion graphs are introduced as "maneuver automata", in which trims form the vertices and maneuvers the edges of the graph. This defines the concatenation rules, i.e., any path in the automaton defines a sequence of primitives. Together with a choice of coasting times, this sequence can be transformed into an admissible, controlled trajectory of the underlying dynamical system.

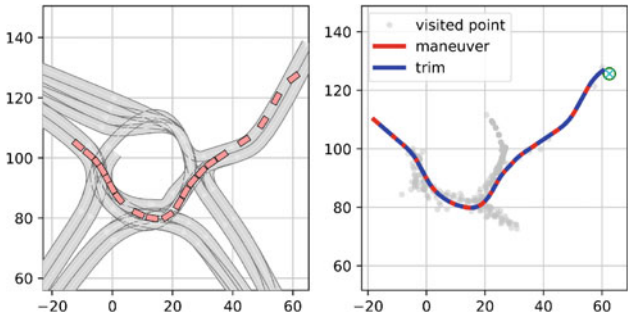
As presented in Sect. 3, maneuver automata can be constructed in an automatic way by extracting representative primitives from a data set. In [24], numerical examples were solved to compare handcrafted and extracted automata for the kinematic single-track model (Eq. (28)). The handcrafted automata consider a usual pragmatic way of designing it: a grid covering the entire space of allowed velocities and steering angles for the model [23]. For comparison, the handcrafted and extracted automata had the same quantity of trims and a similar number of maneuvers. A visual comparison of these two different ways of constructing an automaton is given in Fig. 7, considering the selection of 21 trim primitives. The difference in the trajectory planning when using each of these automata is replicated in Fig. 8. Note that the extracted primitives fit better to the road shape and the final goal position.

For the planning problem, a starting trim is assumed and an initial condition $x(0)$, i.e., a starting node in the MPA, is given. For a guarantee of the existence of a solution from an initial trim to a final trim (or node), it is shown in [14] that one of the requirements is the strong connectivity of the MPA. However, a priori, an MPA

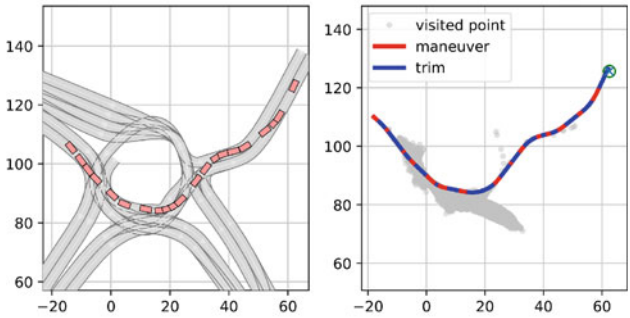


(**a**) Handcrafted automaton. (**b**) Extracted automaton.

Fig. 7 Automata with 21 trim primitives. The dots correspond to trim primitives (axes: velocities versus steering angle) and the colored lines represent maneuvers connecting the trims (figures from [24])

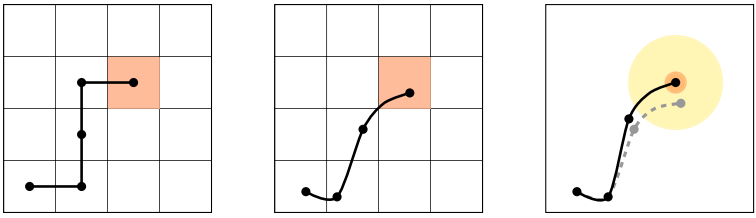


(**a**) Handcrafted automaton.



(**b**) Extracted automaton.

Fig. 8 Trajectories for the two different automata with 21 trim primitives (figures from [24])



(a) A* search (b) Hybrid A* search (c) Π* search

Fig. 9 Comparison between different graph search methods: the goal regions are denoted in red and the yellow area is an optimization region, where the Π* will try to optimize the trim's coasting times to lead the vehicle to a goal point inside the goal region

does not need to be strongly connected. For the cases where there exists more than one admissible solution, an optimization problem can be posed.

6 Planning Algorithms

With a library of primitives condensed into a graph, path planning can be done using different techniques. In this section, we will mention some of the ideas developed. The complementary chapter will, however, delve into planning in a cooperative trajectory planning scenario.

6.1 Optimized Primitives (Π) Search*

The Π* search was developed in [23] and it is inspired by the Hybrid A* algorithm [10], an A*-based search. In the Hybrid A*, continuous states are associated with grid cells and the costs of the states, therefore, are the cost of their respective cell. However, in Π* search, each state is fully continuous, instead of being associated with discrete grid cells. The trims' coasting times can be adjusted by an online optimization problem of reduced complexity. The algorithm encapsulates the method of anytime search to deal with time deadlines [32]. The search, then, can lead the vehicle to an exact goal point in the state space while respecting computation time constraints. Figure 9 compares the different graph search methods.

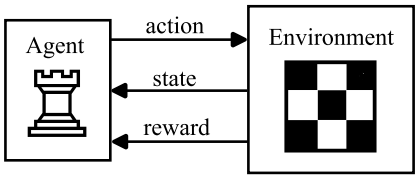


Fig. 10 The interaction of agent and environment in reinforcement learning

6.2 Reinforcement Learning

Reinforcement learning as a Markov decision process, as described in [29], is the task of learning from the interaction between an agent and an environment to achieve a goal. The agent is the decision-maker and learns which is the best action given the current state. A numerical value evaluates an action and it is called "reward". Thus, the action is selected to maximize the rewards. The environment, in turn, responds to the agent with a new state and the reward for a given action. A schematic depiction of this iterative process can be seen in Fig. 10.

It is possible to use primitives as the actions of a reinforcement learning agent, as opposed to using a discrete or continuous set of control inputs as the action space [15]. A work in this regard was developed in the Bachelor's thesis [28].

6.3 Graph-Based Receding Horizon Control

Introduced in [27], this method aims to transfer the receding horizon control approach into graph-search problems, specially made for maneuver automata. Thus, nonlinear, nonconvex optimization problems are solved in real-time, in opposite to traditional graph-search approaches that keep the search until the goal vertex is found. This approach was applied to cooperative planning of multiple networked and autonomous vehicles on the CPM Lab [16]. Also, it was shown that the solutions are recursively feasible by design of the finite state automaton. This method is explained in detail in the chapter "Prioritized Trajectory Planning for Networked Vehicles Using Motion Primitives".

6.4 Motion Graphs as Mixed Logical Dynamical System

We can model the motion graphs as a mixed logical dynamical (MLD) system to transform the graph search into an OCP. MLD systems were introduced by [7] and describe systems by a combination of continuous variables with Boolean ones. As an example of application, an MLD system was modeled to solve collision avoidance of collaborative vehicles in [4]. The authors did not use primitives, but linearized

the vehicle model over the operation points and solved mixed-integer linear and quadratic programming problems.

In short, the idea of our proposed MLD system is to formulate the execution of a primitive at a discrete-time k by "enabling" one primitive over all others available, given which node of the MPA is active for the vehicle. For that, we can define the Boolean variables, for $i = 1, 2, \ldots$, as

$$m_i(k) = \begin{cases} 1, & \text{if the primitive } \mathrm{p}_i \text{ is executed at } k, \\ 0, & \text{otherwise.} \end{cases} \tag{23}$$

where the set of available primitives at time k is $\{\mathrm{p}_i, i \in \mathbb{N}\}$. Then, given the current automaton state in the MPA and $x(k)$, the system dynamics can be written as:

$$\begin{cases} x(k+1) = \sum_i \Psi_{g_i}(x(k)) \cdot m_i(k), & \text{(8.24a)} \\ \sum_i m_i(k) = 1, & \text{(8.24b)} \end{cases}$$

for the continuous times given by

$$t_{k+1} = t_k + \sum_i \tau_i m_i \tag{25}$$

with τ_i representing the duration of the primitive p_i. This modeling approach leads to a mixed-integer nonlinear programming problem when searching for the optimal sequence for a given planning problem within an MPA.

Thus, it is possible to extend this modeling into a model predictive control (MPC) formulation and thus exploit the tools available for MPC, for example, stability, robustness, and inclusion of constraints, in the computation of trajectories with motion primitives.

7 Conclusion

We presented in this chapter a description of methods to design an automaton of motion primitives by properly selecting and constructing them. This automaton of primitives is implemented in trajectory planning for cooperative vehicles and its architecture is essential for efficient paths. We presented a list of vehicle models abstracted in a general formulation. Then, we showed how to abstract typical trim primitives from traffic data and derived maneuvers by the polynomial method and by an OCP. This last one is useful for finding Pareto-optimal maneuvers. We also compared different automata and presented possible algorithms to solve the graph-based planning problem.

Appendices

Here, we present two vehicle models from [5], the kinematic single-track and the single-track model.

A. The Kinematic Single-Track Model

The kinematic bicycle model has the state vector

$$x = \left[s_x \; s_y \; \psi \; v \; \delta\right]^{\mathrm{T}} \in \mathbb{R}^5, \tag{26}$$

and the input vector:

$$u = \left[u_v \; u_{\dot{\delta}}\right]^{\mathrm{T}} \in \mathbb{R}^2, \tag{27}$$

where s_x and s_y are the positions of the rear axis, ψ is the vehicle orientation, v is the velocity, δ is the steering angle, u_v is the longitudinal acceleration, and $u_{\dot{\delta}}$ is the velocity of the steering angle. The state space equations are given by:

$$\begin{cases} \dot{s}_x(t) = v(t) \cdot \cos(\psi(t)), \\ \dot{s}_y(t) = v(t) \cdot \sin(\psi(t)), \\ \dot{\psi}(t) = \dfrac{v(t)}{L} \cdot \tan(\delta(t)), \\ \dot{v}(t) = u_v(t), \\ \dot{\delta}(t) = u_{\dot{\delta}}(t), \end{cases} \tag{28}$$

for L being the wheelbase of the vehicle. In [24], it was used the wheelbase of the Renault Zoe, used in obtaining the nuScenes data, with value 2.588 m [1].

B. The Single-Track Model

The state vector

$$x = \left[s_x \; s_y \; \psi \; \dot{\psi} \; v \; \delta \; \beta\right]^{\mathrm{T}} \in \mathbb{R}^7, \tag{29}$$

has the same variables described for Eq. (26) together with the slip angle at the center of gravity β (see Fig. 11). The inputs are the same as in Eq. (27). The state space equations are:

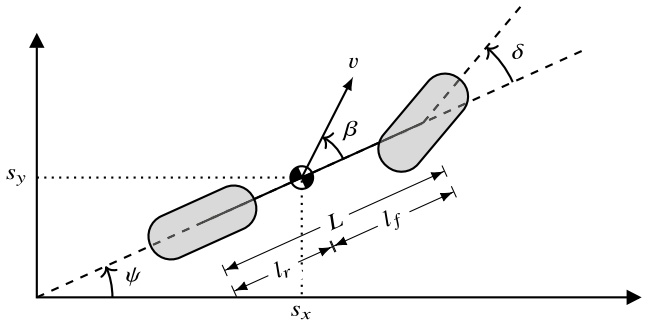


Fig. 11 Single-track model

$$
\begin{cases}
\dot{s}_x(t) = v(t) \cdot \cos(\psi(t) + \beta(t)), \\
\dot{s}_y(t) = v(t) \cdot \sin(\psi(t) + \beta(t)), \\
\dot{\psi}(t) = \dfrac{d}{dt}\psi(t), \\
\ddot{\psi}(t) = \dfrac{\mu M}{I_z L}\Big(l_{\mathrm{f}} \cdot \alpha_{\mathrm{f,r}} \cdot \delta(t) + (l_{\mathrm{r}} \cdot \alpha_{\mathrm{r,f}} - l_{\mathrm{f}} \cdot \alpha_{\mathrm{f,r}})\beta(t) \\
\qquad - (l_{\mathrm{f}}^2 \cdot \alpha_{\mathrm{f,r}} + l_{\mathrm{r}}^2 \cdot \alpha_{\mathrm{r,f}}) \dfrac{\dot{\psi}(t)}{v(t)} \Big), \\
\dot{v}(t) = u_{\dot{v}}(t), \\
\dot{\delta}(t) = u_{\dot{\delta}}(t), \\
\dot{\beta}(t) = \dfrac{\mu}{L \cdot v(t)}\Big(\alpha_{\mathrm{f,r}} \cdot \delta(t) - (\alpha_{\mathrm{r,f}} + \alpha_{\mathrm{f,r}})\beta(t) \\
\qquad + (l_{\mathrm{r}} \cdot \alpha_{\mathrm{r,f}} - l_{\mathrm{f}} \cdot \alpha_{\mathrm{f,r}}) \dfrac{\dot{\psi}(t)}{v(t)} \Big) - \dot{\psi}(t),
\end{cases}
\tag{30}
$$

where $\alpha_{i,j} := \alpha_{i,j}(u_{\dot{\delta}}(t))$ is a function of the input $u_{\dot{\delta}}(t)$ defined as

$$
\alpha_{i,j} = C_i(g \cdot l_j - h \cdot u_{\dot{\delta}}(t)) \tag{31}
$$

for $i, j \in \{\mathrm{f}, \mathrm{r}\}$, L given by $L = l_{\mathrm{f}} + l_{\mathrm{r}}$ and the parameters described in Table 1 with the values used in [23].

Table 1 Single-track model's parameters

Parameter	Symbol	Unit	Value
Distance from the center of gravity to front axle	l_f	$[m]$	0.883
Distance from the center of gravity to rear axle	l_r	$[m]$	1.508
Total vehicle mass	M	$[kg]$	1.225
Moment of inertia about z axis	I_z	$[kg \cdot m^2]$	1.538
Center of gravity height of M	h	$[m]$	0.557
Cornering stiffness coeff. (front, rear)	C_f, C_r	$[1/rad]$	20.89
Friction coefficient	μ	$[-]$	1.048

Acknowledgements We thank Tristan Schneider and Matthias K. Hoffmann for their contributions on this chapter. This research is supported by the Deutsche Forschungsgemeinschaft (German Research Foundation) within the Priority Program SPP 1835 "Cooperative Interacting Automobiles" (grant number: KO 1430/17-1).

References

1. Renault Zoe dimensions & specifications. https://www.renault.co.uk/electric-vehicles/zoe/specifications.html. Accessed 25 Feb 2021
2. Coordinated non-cooperative distributed model predictive control for decoupled systems using graphs. IFAC-PapersOnLine **49**(22), 216–221 (2016). https://doi.org/10.1016/j.ifacol.2016.10.399. 6th IFAC Workshop on Distributed Estimation and Control in Networked Systems NECSYS 2016
3. Alrifaee, B.: Networked model predictive control for vehicle collision avoidance. Ph.D. thesis (2017). https://doi.org/10.18154/RWTH-2017-04199
4. Alrifaee, B., Mamaghani, M.G., Abel, D.: Centralized non-convex model predictive control for cooperative collision avoidance of networked vehicles. In: 2014 IEEE International Symposium on Intelligent Control (ISIC), pp. 1583–1588 (2014). https://doi.org/10.1109/ISIC.2014.6967623
5. Althoff, M., Koschi, M., Manzinger, S.: Commonroad: Composable benchmarks for motion planning on roads. In: Proceedings of the IEEE Intelligent Vehicles Symposium (2017). https://doi.org/10.1109/ivs.2017.7995802
6. Andersson, J.A.E., Gillis, J., Horn, G., Rawlings, J.B., Diehl, M.: CasADi—a software framework for nonlinear optimization and optimal control. Math. Program. Comput. **11**(1), 1–36 (2019). https://doi.org/10.1007/s12532-018-0139-4
7. Bemporad, A., Morari, M.: Control of systems integrating logic, dynamics, and constraints. Automatica **35**(3), 407–427 (1999). https://doi.org/10.1016/S0005-1098(98)00178-2
8. Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: Nuscenes: a multimodal dataset for autonomous driving. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 11621–11631 (2020)
9. Dellnitz, M., Eckstein, J., Flaßkamp, K., Friedel, P., Horenkamp, C., Köhler, U., Ober-Blöbaum, S., Peitz, S., Tiemeyer, S.: Multiobjective optimal control methods for the development of an intelligent cruise control. In: Russo, G., Capasso, V., Nicosia, G., Romano, V. (eds.) Progress in

Industrial Mathematics at ECMI 2014, pp. 633–641. Springer International Publishing, Cham (2016)
10. Dolgov, D., Thrun, S., Montemerlo, M., Diebel, J.: Practical search techniques in path planning for autonomous driving. Ann. Arbor. **1001**(48105), 18–80 (2008)
11. Dubins, L.E.: On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. Am. J. Math. **79**(3), 497–516 (1957)
12. Flaßkamp, K., Ober-Blöbaum, S., Kobilarov, M.: Solving optimal control problems by exploiting inherent dynamical systems structures. J. Nonlinear Sci. **22**, 599–629 (2012)
13. Flaßkamp, K., Ober-Blöbaum, S., Worthmann, K.: Symmetry and motion primitives in model predictive control. Math. Control Signals Syst. **31**, 455–485 (2019)
14. Frazzoli, E., Dahleh, M., Feron, E.: Maneuver-based motion planning for nonlinear systems with symmetries. IEEE Trans. Rob. **21**(6), 1077–1091 (2005). https://doi.org/10.1109/TRO.2005.852260
15. Kiran, B.R., Sobh, I., Talpaert, V., Mannion, P., Al Sallab, A.A., Yogamani, S., Pérez, P.: Deep reinforcement learning for autonomous driving: a survey. IEEE Trans. Intell. Transp. Syst. (2021)
16. Kloock, M., Scheffe, P., Maczijewski, J., Kampmann, A., Mokhtarian, A., Kowalewski, S., Alrifaee, B.: Cyber-physical mobility lab: an open-source platform for networked and autonomous vehicles. In: 2021 European Control Conference (ECC), pp. 1937–1944 (2021). https://doi.org/10.23919/ECC54610.2021.9654986
17. Kloock, M., Scheffe, P., Marquardt, S., Maczijewski, J., Alrifaee, B., Kowalewski, S.: Distributed model predictive intersection control of multiple vehicles. In: 2019 IEEE Intelligent Transportation Systems Conference (ITSC), pp. 1735–1740 (2019). https://doi.org/10.1109/ITSC.2019.8917117
18. Knauer, M., Büskens, C.: Real-time optimal control using TransWORHP and WORHP Zen, pp. 211–232. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-030-10501-3_9
19. Kobilarov, M.: Discrete geometric motion control of autonomous vehicles. University of Southern California (2008)
20. Lloyd, S.P.: Least squares quantization in PCM. IEEE Trans. Inf. Theory **28**, 129–137 (1982)
21. Lüttgens, L., Jurgelucks, B., Wernsing, H., Roy, S., Büskens, C., Flaßkamp, K.: Autonomous navigation of ships by combining optimal trajectory planning with informed graph search. Math. Comput. Model. Dyn. Syst. **28**(1), 1–27 (2022). https://doi.org/10.1080/13873954.2021.2007138
22. Paden, B., Čáp, M., Yong, S.Z., Yershov, D., Frazzoli, E.: A survey of motion planning and control techniques for self-driving urban vehicles. IEEE Trans. Intell. Veh. **1**(1), 33–55 (2016). https://doi.org/10.1109/TIV.2016.2578706
23. Pedrosa, M.V.A., Schneider, T., Flaßkamp, K.: Graph-based motion planning with primitives in a continuous state space search. In: 2021 6th International Conference on Mechanical Engineering and Robotics Research (ICMERR), pp. 30–39 (2021). https://doi.org/10.1109/ICMERR54363.2021.9680825
24. Pedrosa, M.V.A., Schneider, T., Flaßkamp, K.: Learning motion primitives automata for autonomous driving applications. Math. Comput. Appl. **27**(4) (2022). https://doi.org/10.3390/mca27040054. https://www.mdpi.com/2297-8747/27/4/54
25. Reeds, J., Shepp, L.: Optimal paths for a car that goes both forwards and backwards. Pac. J. Math. **145**(2), 367–393 (1990)
26. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach, 3rd edn. Prentice Hall (2010)
27. Scheffe, P., Pedrosa, M.V.A., Flaßkamp, K., Alrifaee, B.: Receding horizon control using graph search for multi-agent trajectory planning. IEEE Trans. Control Syst. Technol. 1–14 (2022). https://doi.org/10.1109/TCST.2022.3214718
28. Schneider, T.: A comparison between reinforcement learning and graph search for motion planning in autonomous driving applications. Bachelor's thesis, Saarland University, Germany (2022)

29. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press (2018)
30. Yang, Y., Pan, J., Wan, W.: Survey of optimal motion planning. IET Cyber-Syst. Robot. **1**, 13–19(6) (2019). https://doi.org/10.1049/iet-csr.2018.0003
31. Yurtsever, E., Lambert, J., Carballo, A., Takeda, K.: A survey of autonomous driving: common practices and emerging technologies. IEEE Access **8**, 58443–58469 (2020). https://doi.org/10.1109/ACCESS.2020.2983149
32. Zilberstein, S.: Using anytime algorithms in intelligent systems. AI Mag. **17**(3), 73 (1996). https://doi.org/10.1609/aimag.v17i3.1232

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

