2018 European Control Conference (ECC)
June 12-15, 2018. Limassol, Cyprus

Optimization-Based Maneuver Automata for Cooperative Trajectory Planning of Autonomous Vehicles

Jan Eilbrecht[1] and Olaf Stursberg[1]

Abstract— **In this paper, a major challenge in the field of trajectory planning for autonomous on-road vehicles is addressed: to calculate trajectories efficiently and reliably. The proposed method is based on a cooperative maneuver automaton, in which each maneuver is formulated as a discrete-time hybrid automaton that defines constraints on the behavior of a group of autonomous vehicles. For trajectory planning, these constraints enter an optimal control problem, which is formulated as a Mixed-Integer Quadratic Program (MIQP). Despite good practical performance, these programs have adverse worst-case run-time properties, which, however, are improved by the proposed approach. Also, calculation of controllable sets for the hybrid automata allows to determine both the feasibility of an optimal control problem for a given initial state and the required time to complete the maneuver in advance – without solving the optimization problem. The efficacy of the proposed method is demonstrated in an example scenario.**

I. INTRODUCTION

A. Motivation

Trajectory planning is one of the major challenges in the operation of autonomous vehicles such as mobile robots, unmanned aerial or underwater vehicles, and autonomous on-road cars. Existing planning procedures are usually based on optimization of a cost function in order to obtain a trajectory which is optimal with respect to a certain criterion, for example traveling time, energy consumption, or driving comfort. Depending on the application, these optimization-based procedures must meet different requirements: Autonomous robots usually operate in highly-unstructured, obstructed environments, calling for flexible methods. At the same time, these robots often move slowly and the environment mostly consists of static obstacles, such that the dynamics can be neglected in the planning and the time needed to solve the planning task is not critical. In other applications, in contrast, the vehicles move at higher speeds. This necessitates to account for their dynamics and thus increases the problem complexity and computation times, see for example [1]. If in that case, however, the number of actions available to a vehicle is rather limited, i.e., the environment is structured, this structure can be exploited by planning algorithms.

A concept that exploits this structure is the *maneuver automaton*, e.g. [2]–[5]. Typically, it consists of a finite set of trajectories which correspond to the actions available to the vehicle and which are determined offline. During online operation, an appropriate trajectory only has to be chosen. While this results in a loss of flexibility (smaller number of possible actions), it significantly speeds up the online planning. In [2], the concept was originally introduced for aerial vehicles, which do not frequently encounter static or moving obstacles, such that a loss of flexibility is not critical. In the presence of moving obstacles, as is common in on-road autonomous driving, the number of available actions must not be reduced too much in order to be able to prevent collisions. This would require a maneuver automaton with many pre-calculated maneuvers, which is difficult to design and time-consuming to explore online.

[1]Institute of Control and System Theory, Dept. of Electrical Eng. and Computer Science, University of Kassel, 34121 Kassel, Germany {jan.eilbrecht,stursberg}@uni-kassel.de

B. Contribution

Addressing this very problem, we propose a novel interpretation of the maneuver automaton concept for structured environments, which combines the flexibility of general purpose planners with the fast planning of classical maneuver automata. The automaton no longer consists of pre-calculated trajectories, but of formulations of optimization problems tailored to each maneuver. In addition, maneuvers define actions for several vehicles, i.e., planning is cooperative. By formulating the maneuver automaton as a hybrid automaton, we are able to determine both the feasibility and the duration of a maneuver prior to planning it in detail. We further describe a possible use of the maneuver automaton within a planning architecture.

C. Related Work

Owing to the high relevance and long history of the field, many algorithms exist for trajectory planning of autonomous vehicles. Early work mostly focused on planning without dynamics [6], relying heavily on graph-searching procedures. Among the procedures accounting for the dynamics of the vehicle, those based on mixed-integer programming in combination with model-predictive control (MPC) have found wide and successful application both for aerial and on-road vehicles [1], [7], [8], and will be used in this paper.

An early definition of a maneuver automaton was introduced in [2] for control of a miniature helicopter. Because no obstacles are considered, the automaton consists only of two basic elements: so-called trim trajectories as steady states, resulting from constant actuator inputs, and maneuvers for the transition between these trim trajectories. A more complex maneuver automaton is presented in [9], where maneuvers are defined as hybrid automaton, for which a supervisory controller is designed and verified using online reachability analysis. In [4], a similiar approach is chosen, where reachability analysis for a nonlinear vehicle model is used to design parameterizable motion primitives which can cover a continuous action space. However, these

primitives are determined offline, where it is attempted to connect them to form a graph in such a way that a target region is eventually reached while obstacles are avoided. The process of building the maneuver automaton, however, may be problematic if the connection attempts fail for too many primitives. This problem is addressed in [5], where optimal control is combined with reachability analysis in order to obtain maneuvers with better properties regarding connectivity. All these approaches, however, verify properties for given *initial sets*, whereas this paper focuses on given *target sets* and determines controllable initial sets.

Controllable sets have been applied successfully, primarily in the field of flight control using level-set methods. In [10], controllable sets are obtained for continuous-time dynamics in order to determine whether or not an aircraft can land safely. In [11], the same question is considered, though the process is modeled as a hybrid automaton this time. For each location of the automaton, controllable sets are determined under which the aircraft will remain safe or transition to other nodes. Computations are carried out for nonlinear, continuous-time dynamics, solving Hamilton-Jacobi partial differential equations. In [12], the problem of automated aerial refueling is considered. The process is modeled as a hybrid automaton, in which states represent waypoints. A transition between waypoints is initiated by a human operator, while controllable sets indicate if a waypoint can be reached. In [13], complex acrobatic flights are decomposed into a sequence of discrete modes, each of which is governed by a different controller. Safe transitions between these modes are ensured using controllable sets, also accounting for disturbances in a dynamic game formulation, but not for obstacles in the environment. Set calculations are again based on the method employed in [11], which is limited to very low-dimensional systems (2–3 states). In [14], an approach based on optimal control methods is given which is also concerned with assessing the feasibility of certain maneuvers, however focusing on emergency situations.

The proposed method contrasts these approaches as follows: The maneuver automaton does not contain trajectories or closed-loop dynamics, but maneuver-dependent formulations of optimization problems which can be solved efficiently online, increasing the flexibility of the trajectory planning. Since optimization requires linear, discrete-time systems for tractability and on-line implementation, the controllable sets are calculated for this system class. Because the level-set methods employed in the references above are limited to lower dimensions than required for this application, a different method is used in this work.

This paper is structured as follows: Sec. II details the maneuver automaton and the architecture in which it is embedded, along with the optimal control problem used for trajectory generation. Sec. III provides details on controllable sets and their computation, which is exemplified in Sec. IV.

II. Planning Framework

A. Architecture

The methods proposed in this paper are envisioned to be implemented in the planning architecture shown in Fig. 1. For now, non-autonomous traffic participants are not considered, which will be subject of future work. Each autonomous vehicle contains the following modules as illustrated exemplary for vehicle i: The *route planner* functions as a conventional navigation unit and only determines on which roads to drive. The *trajectory planner*, which is detailed in Sec. II-C, determines position references for a given set of vehicles (the "*coalition*") which are to perform a certain maneuver on the specified road within a specified duration. A coalition is formed and a maneuver of a certain duration is assigned to it according to the decision of the *scheduler*-module. This maneuver then has to be planned cooperatively by the vehicles, i.e., we assume that they communicate with each other and optimize a common cost function. The obtained position set points then have to be tracked by a *low-level control* that actuates the single vehicles.

Since the scheduler-module is not in the focus of this paper, we assume that it is given, for example by a road side unit or, more preferably, implemented in a distributed fashion in the single vehicles, which communicate with each other. Regardless of the implementation, we assume that the scheduler decides based on information about: which vehicles are available for forming a coalition, what their preferences are (quantified by some cost function), and what their options are (depending on their current positions and velocities). These options are encoded in the *maneuver automaton*, which is described in detail in the following section.

B. Maneuver Automaton

In this work, the maneuver automaton is a hierarchy of two layers, cf. Fig. 2: On the upper layer, the states of a finite state machine represent possible maneuvers. The default-maneuver "Lane Keeping" requires all vehicles to stay in their current lane, only adapting longitudinal speed according to that of the vehicles in front. If all vehicles in a certain radius are in this state, the scheduler module can form new coalitions or initiate other maneuvers. Upon completion, the

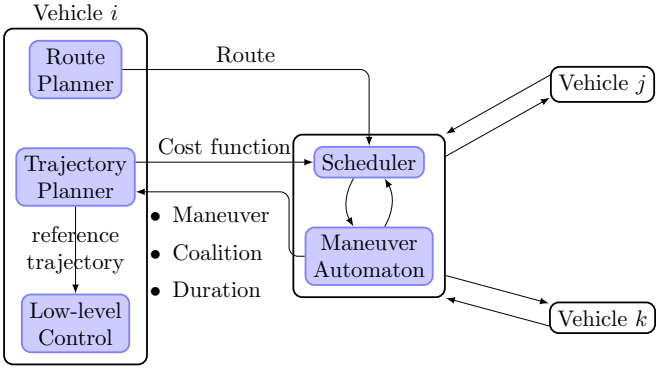

Fig. 1. Envisioned planning architecture.

lane-keeping-state then becomes active again. Note that if the depicted selection of maneuvers should not suffice in practice, more can be added easily.

On the lower layer, a hybrid automaton translates a certain maneuver into constraints on the actions of the involved vehicles. These constraints are accounted for during the planning of the maneuver, which is carried out by the trajectory planner as described in Sec. II-C. Extending on [13], the basic idea is to divide each maneuver in several phases, which are represented by the locations of the hybrid automaton. The transitions between the phases depend on certain conditions and in each phase, different constraints, e.g. on the position of the vehicles in order to prevent collisions, are enforced. This can be formalized as a discrete-time hybrid automaton. According to the syntax given by [15], a hybrid automaton consists of:

1) a finite set of locations $\mathcal{Q}$ with initial location $q_0 \in \mathcal{Q}$,
2) a continuous state space $\mathbb{X} \subseteq \mathbb{R}^{n_x}$ with the set of initial states $\mathcal{X}_0 \subseteq \mathbb{X}$,
3) a continuous input space $\mathbb{U} \subseteq \mathbb{R}^{n_u}$,
4) a function $\text{inv} : \mathcal{Q} \to 2^{\mathbb{X}}$, assigning to each location $q \in \mathcal{Q}$ a set $\text{inv}(q) \subseteq \mathbb{X}$ in which the continuous state $x \in \mathbb{X}$ may evolve (assume that $\mathcal{X}_0 \subseteq \text{inv}(q_0)$),
5) a set of discrete transitions $\Theta \subseteq \mathcal{Q} \times \mathcal{Q}$,
6) a guard function $g : \Theta \to 2^{\mathbb{X}}$ which assigns a guard set $g(\theta) \subseteq \mathbb{X}$ to each transition $\theta = (q_i, q_j) \in \Theta$,
7) a jump function $j : \Theta \times \Theta \times \mathbb{X} \to \mathbb{X}$ that assigns a state $j(\theta, x) \in \mathbb{X}$ to each pair $\theta \in \Theta$ and $x \in g(\theta)$,
8) and a flow function $f : \mathcal{Q} \to (\mathbb{X} \times \mathbb{U} \to \mathbb{R}^{n_x})$.

We consider discrete-time dynamics on the time domain:

$$\mathbb{T} = \{t_k | t_k = kT_\text{s},\ T_\text{s} \in \mathbb{R}^+,\ k \in \mathbb{N}\}.$$

An *admissible run* of the automaton for a sequence of inputs $(u_0, \ldots, u_k, \ldots, u_{k+N})$ at time instants $(t_0, \ldots, t_k \ldots, t_{k+N})$ starts from an admissible initial location and continuous state, i.e., $(q_0, x_0) \in \mathcal{Q} \times \mathcal{X}_0$. Successor states (q_{k+1}, x_{k+1}) at time t_{k+1} are obtained in two steps:

1) At first, a candidate state $\hat{x}$ is obtained through continuous evolution based on the flow function, $\hat{x}_{k+1} = f(q_k, x_k, u_k)$. If $\hat{x}_{k+1} \in \text{inv}(q_k)$ and $\hat{x}_{k+1} \notin g(\theta)$ for any $\theta \in \Theta$, the candidate is accepted without changing the location: $x_{k+1} = \hat{x}_{k+1}$, $q_{k+1} = q_k$.
2) If $\hat{x}_{k+1} \in g(\theta)$ for some $\theta = (q_k, q_{k+1}) \in \Theta$, an immediate transition is enforced and the jump function is applied. The resulting state is only admissible if it lies inside the invariant of the successor location: $x_{k+1} = j(\theta, \hat{x}_{k+1}) \in \text{inv}(q_k + 1)$.

States outside of the invariant but not inside of a guard set are not admissible.

The bottom part of Fig. 2 delineates the general structure of a hybrid automaton, while an example for the formulation of a specific maneuver is given in Sec. IV-A. For simplicity, we assume that a maneuver is defined through appropriate state constraints in such a way that its locations form a directed tree in which each node has at most one predecessor and at most one successor. While this topology constrains the flexibility of the trajectory planner, it also reduces the complexity of the computations required in Sec. II-C. If higher flexibility is required, the assumption can be dropped at the price of higher computational efforts.

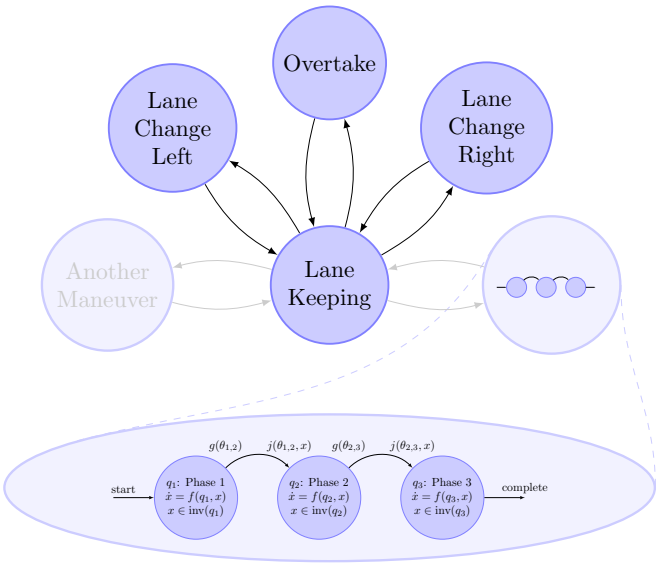

Fig. 2. Hierarchical maneuver automaton: Finite state machine (top) and hybrid automaton (close-up at bottom).

C. Trajectory Planner

While the choice of a certain maneuver made by the scheduler determines a specific qualitative behavior of the involved vehicles in the sense of a *set* of admissible trajectories, the actually resulting reference trajectories for the single vehicles still need to be determined. Given a coalition, a maneuver, and its scheduled duration, such trajectories are obtained as the result of an optimal control problem. This aims at minimizing the value of a performance criterion predicted for a sequence of inputs, while accounting for the constraints imposed by the dynamics of the vehicles as well as by the maneuver. For computational tractability, we consider discrete-time, linear dynamics as flow of the automaton:

$$x(t_{k+1}) = A_q x(t_k) + B_q u(t_k), \tag{1}$$

with possibly location-dependent matrices $A_q \in \mathbb{R}^{n_x \times n_x}$ and $B_q \in \mathbb{R}^{n_x \times n_u}$. Note that the flow function aggregates the dynamics of all vehicles participating in a maneuver. In this paper, the dynamics is assumed to be known exactly, while future work will aim at accounting for uncertainties. If time indices are omitted in the following, a quantity refers to time step t_k. Quantities predicted at time t_k for j future time steps ahead are denoted by double brackets, e.g. $x[\![j]\!] := x(t_{k+j}|t_k)$. The objective is to minimize:

$$J = \sum_{j=1}^{N} \left(x[\![j]\!] - x_\text{ref}\right)^\mathsf{T} Q \left(x[\![j]\!] - x_\text{ref}\right) + u[\![j-1]\!]^\mathsf{T} R u[\![j-1]\!] \tag{2}$$

over the prediction horizon $N \in \mathbb{N}^+$, in which $x[\![0]\!] = x(t_k)$, $x_\text{ref} \in \mathbb{R}^{n_x}$ is a known state reference value, and $Q \geq 0$ and

$R \geq 0$ are weighting matrices of suitable dimensions. In order to enforce completion of the maneuver, the terminal constraint:

$$x[\![N]\!] \in \mathcal{X}_\mathrm{t}$$

is imposed, where $\mathcal{X}_\mathrm{t} \subset \mathbb{X}$ is a target set, cf. Sec. III. Also, input constraints:

$$u(t_k) \in \mathbb{U}$$

must hold. Maneuvers impose constraints of the type:

$$E_1 x(t_k) + E_2 \delta(t_k) + E_3 \leq 0,$$

combining continuous state variables x with binary variables δ using matrices E_1, E_2, and E_3 of suitable dimensions. The binary variables are required to model phase transitions. Their presence, together with the quadratic cost function and linear constraints, leads to a Mixed-Integer Quadratic Program (MIQP). Despite their adverse theoretic run-time properties, practical performance is good if sophisticated solvers like CPLEX [16] or GUROBI [17] are used.

III. j-Step Controllable Sets

The architecture described in Sec. II-A relies on the separability of the choice of a maneuver and its duration on the one hand, and the decision on the exact realization of this maneuver in terms of a trajectory on the other hand. While this property does not hold in general, it can be established using the concept of controllable sets (also known as backwards reachable sets). If a maneuver is formulated as a hybrid automaton, it is completed if the automaton reaches a target location $q_\mathrm{t} \in \mathcal{Q}$ and a set $\mathcal{X}_\mathrm{t} \subseteq \mathrm{inv}(q_\mathrm{t})$ of target states. States which can reach the target set in j time steps are contained in the j-step controllable set $\mathcal{K}_j(q_\mathrm{t}, \mathcal{X}_\mathrm{t}) \subseteq \mathcal{Q} \times \mathbb{X}$, with $j \in \mathbb{N}_0$. Given controllable sets for different j, it becomes possible for the scheduling unit to determine if for a given initial state x_0 it is possible to reach the target set and how many time steps j are required at least. This information can be obtained by testing if the initial state is included in some of the known controllable sets. It is not required to actually calculate the trajectory, which is highly beneficial, especially if the scheduling module has to assess several options. Since the controllable sets can be determined offline prior to deployment of the planning software, the only extra cost in online operation comes from testing set inclusion – the sets themselves do not enter the optimization problem. If x_0 is included in some set $\mathcal{K}_j(q_\mathrm{t}, \mathcal{X}_\mathrm{t})$, the index j of such a set can then be used as prediction horizon N for the cost function (2) of the optimal control problem.

Similarly as in [18] for purely continuous dynamics, the j-step controllable set $\mathcal{K}_j(q_\mathrm{t}, \mathcal{X}_\mathrm{t})$ is defined recursively by:

$$\mathcal{K}_j(q_\mathrm{t}, \mathcal{X}_\mathrm{t}) = \mathtt{Pre}(\mathcal{K}_{j-1}(q_\mathrm{t}, \mathcal{X}_\mathrm{t})), \tag{3}$$

with $\mathcal{K}_0(q_\mathrm{t}, \mathcal{X}_\mathrm{t}) = \{(q_\mathrm{t}, \mathcal{X}_\mathrm{t})\}$. The operator $\mathtt{Pre} : \mathcal{Q} \times \mathbb{X} \to \mathcal{Q} \times 2^{\mathbb{X}}$ gives the set from which the tuple $(q, \mathcal{X})$, $\mathcal{X} \subseteq \mathrm{inv}(q)$, can be reached in one time step. Following [19, p. 42], it is defined to consist of two parts. The first is to compute the set of predecessor states resulting from continuous evolution of the dynamics in the same location q, $\mathtt{Pre}_\mathrm{c}(q, \mathcal{X})$:

$$\begin{aligned}\mathtt{Pre}_\mathrm{c}(q, \mathcal{X}) = \{(q, x) | x \in \mathrm{inv}(q),\ \exists u \in \mathbb{U} :\\ A_q x + B_q u \in \mathcal{X}\}.\end{aligned} \tag{4}$$

This set is obtained by intersecting the set of all continuous predecessors of the set $\mathcal{X}$ with the invariant of the current location.

The second part consists of sets of states resulting after discrete transitions of states from different locations to q, $\mathtt{Pre}_\mathrm{d}(\theta, \mathcal{X})$. The set $\Theta^- \subseteq \Theta$ of all discrete transitions leading into q is:

$$\Theta^-(q) = \{\theta | \theta = (q^-, q) \subseteq \Theta\}.$$

For each of these transitions $\theta \in \Theta^-$ (if there are any), the set of states that can reach $\mathcal{X}$ after one transition is:

$$\begin{aligned}\mathtt{Pre}_\mathrm{d}(\theta, \mathcal{X}) = \{(q^-, x) |\ x \in \mathrm{inv}(q^-),\ \exists u \in \mathbb{U} :\\ A_{q^-} x + B_{q^-} u \in g(\theta),\ j(\theta, A_{q^-} x + B_{q^-} u) \in \mathcal{X}\}.\end{aligned} \tag{5}$$

Such a set is obtained in several steps: at first, the jump function, which must be invertible, is inverted on the set $\mathcal{X}$. The result is intersected with the guard set of θ. Then, the continuous predecessor set of this set is determined and intersected with the invariant of the predecessor location q^-. Combining (4) and (5), $\mathtt{Pre}$ in (3) is then defined as:

$$\mathtt{Pre}(q, \mathcal{X}) = \mathtt{Pre}_\mathrm{c}(q, \mathcal{X}) \cup \bigcup_{\theta \in \Theta^-(q)} \mathtt{Pre}_\mathrm{d}(\theta, \mathtt{Pre}_\mathrm{c}(q, \mathcal{X})).$$

Given this operator, the controllable sets can be determined iteratively as summarized in the algorithm shown in Fig. 3. Note that $\mathtt{Pre}$ may generate several predecessor sets, depending on the number of discrete predecessors. In that case, the kth j-step controllable set is denoted by $\mathcal{K}_j^{(k)}$. While the number of sets can grow exponentially from step to step, this must not be an issue in practice for three reasons: at first, set computations are carried out off-line. Second, it is possible to drop some sets at the cost of covering less initial states. This is different from forward reachability analysis, where it is important to track all sets in order to guarantee avoidance of forbidden states. Third, for real-life maneuvers, the number of time steps required in total will not be prohibitively large.

The procedure described so far has been generic; for implementational purposes, however, a representation of sets is required. Common set representations are polytopes, ellipsoids, or zonotopes, where each has its own advantages and disadvantages. Due to their ability to represent sets with high accuracy, in this paper, all sets are represented by polytopes, based on the methods implemented in [20].

Require: q_t, $\mathcal{X}_\mathrm{t}$, N
1: $\mathcal{K}_0 \leftarrow \{(q_\mathrm{t}, \mathcal{X}_\mathrm{t})\}$
2: **for** $j = 1$ to N **do**
3: **for** $k = 1$ to $|\mathcal{K}_{j-1}|$ **do**
$\mathcal{K}_j \leftarrow \mathcal{K}_j \cup \mathtt{Pre}\left(\mathcal{K}_{j-1}^{(k)}(q_\mathrm{t}, \mathcal{X}_\mathrm{t})\right)$
4: **end for**
5: **end for**

Fig. 3. High-level algorithm for controllable set computation.

IV. Example Maneuver

A. Maneuver Formulation

The efficacy of the proposed method is demonstrated by means of a simple overtaking maneuver as illustrated in Fig. 4, where Vehicle 1 is scheduled to overtake Vehicle 2. This must be enabled by Vehicle 2 and the oncoming Vehicle 3. Choosing the same simple vehicle model for all three vehicles, the dynamical state of the ith vehicle, $i \in \{1,2,3\}$, is described by its longitudinal position and velocity, $p_x^{(i)}$ and $v_x^{(i)}$, respectively, and its lateral position and velocity, $p_y^{(i)}$ and $v_y^{(i)}$. Because only positions of the vehicles relative to each other are relevant, the relative positions:

$$p_{\text{rel}}^{(2)} := p_x^{(2)} - p_x^{(1)},\ p_{\text{rel}}^{(3)} := p_x^{(3)} - p_x^{(1)}$$

are introduced. In addition, vehicles 2 and 3 are constrained to keep their lanes during the maneuver, such that the lateral velocities $v_y^{(2)}$ and $v_y^{(3)}$ are set to zero, leading to constant lateral positions $p_y^{(2)}$ and $p_y^{(3)}$. These simplifications reduce the dimensionality of the continuous state space $\mathbb{X}$ of the hybrid automaton to be defined. The composed state vector reads:

$$x = \begin{bmatrix} p_{\text{rel}}^{(2)} & p_{\text{rel}}^{(3)} & p_y^{(1)} & v_x^{(1)} & v_x^{(2)} & v_x^{(3)} & v_y^{(1)} \end{bmatrix}^{\mathsf{T}}.$$

As inputs, the accelerations in the different directions are chosen, i.e., $u_x^{(i)} := \dot{v}_x^{(i)}$ and $u_y^{(i)} := \dot{v}_y^{(i)}$, such that:

$$u = \begin{bmatrix} u_x^{(1)} & u_x^{(2)} & u_x^{(3)} & u_y^{(1)} \end{bmatrix}^{\mathsf{T}}.$$

In continuous time, the dynamics then becomes:

$$\dot{x} = A_{\text{c}} x + B_{\text{c}} u,$$

with

$$A_{\text{c}} = \begin{bmatrix} & -1 & 1 & 0 & 0 \\ \mathbf{0}_{3\times 3} & -1 & 0 & 1 & 0 \\ & 0 & 0 & 0 & 1 \\ & \multicolumn{4}{c}{\mathbf{0}_{4\times 7}} \end{bmatrix},\quad B_{\text{c}} = \begin{bmatrix} \mathbf{0}_{3\times 4} \\ I_{4\times 4} \end{bmatrix}.$$

While different matrices A_{c}, B_{c} could be chosen for the different locations q_1, q_2, and q_3, we here choose the same matrices for simplicity. Zero-order hold discretization using a sampling time $T_{\text{s}} = 1\,\text{s}$ eventually gives the flow function (1). The inputs are subject to the constraints:

$$u_{x,\min} \le u_x^{(i)} \le u_{x,\max},\ u_{y,\min} \le u_y^{(1)} \le u_{y,\max},$$

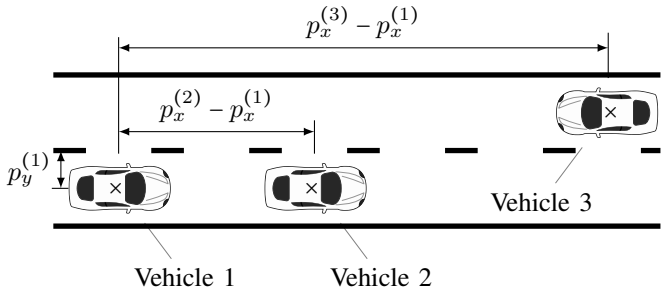

Fig. 4. Example Maneuver: overtaking with oncoming traffic.

and the velocities are chosen such that the vehicles cannot reverse their driving direction:

$$0 \le v_x^{(1)} \le v_{x,\max},\ 0 \le v_x^{(2)} \le v_{x,\max},\ -v_{x,\max} \le v_x^{(3)} \le 0.$$

All vehicles have a longitudinal reference speed of $20\,\text{m}\,\text{s}^{-1}$ and a lateral reference speed of $0\,\text{m}\,\text{s}^{-1}$. For the relative positions, no reference is prescribed, which is implemented by zeroing the corresponding entries in the matrices of the cost function (2), chosen as diagonal matrices with $Q = \text{diag}\,(0,0,1,1,1,1,1)$ and $R = \text{diag}\,(1,1,1,1)$.

The overtaking maneuver is divided into three phases: in the first, Vehicle 1 is behind Vehicle 2, in the second, Vehicle 1 overtakes and is somewhere next to Vehicle 2, and in the third phase, Vehicle 1 drives in between Vehicle 2 and the oncoming Vehicle 3, always maintaining safety distances in longitudinal and lateral direction, $l_{x,\text{safe}}$ and $l_{y,\text{safe}}$, respectively. The relative positions are bounded above and below by $p_{x,\text{rel,max}}$ and $p_{x,\text{rel,min}}$ viz. $p_{y,\text{rel,max}}$ and $p_{y,\text{rel,min}}$. The phases correspond to the set of locations $\mathcal{Q} = \{q_1, q_2, q_3\}$, where transitions are only allowed according to: $\Theta = \{\theta_{1,2} = (q_1, q_2), \theta_{2,3} = (q_2, q_3)\}$. Note that this does not allow to fall back to a prior phase. This may limit flexibility of the maneuver, but drastically decreases computational complexity, as will be discussed later. The jump function is chosen as identity matrix:

$$j(\theta, x) = x.$$

For each location, a definition of different invariant and guard sets is required:

1) Location q_1: The invariant is defined by the constraints:

$$\begin{aligned} p_x^{(2)} - p_x^{(1)} &\le p_x^{(3)} - p_x^{(1)} - l_{x,\text{safe}}, \\ p_x^{(3)} - p_x^{(1)} &\le p_{x,\text{rel,max}}, \\ p_{y,\text{rel,min}} \le p_y^{(1)} &\le p_{y,\text{rel,max}}, \\ 0 \le v_y^{(1)} &\le v_{y,\max}. \end{aligned}$$

The transition $\theta_{1,2}$ occurs in the guard set:

$$g(\theta_{1,2}) = \{x | p_x^{(2)} - p_x^{(1)} \le l_{x,\text{safe}}\}.$$

2) Location q_2: The invariant is defined by:

$$\begin{aligned} p_x^{(2)} - p_x^{(1)} &\le l_{x,\text{safe}}, \\ l_{x,\text{safe}} \le p_x^{(3)} - p_x^{(1)} &\le p_{x,\text{rel,max}}, \\ p_y^{(2)} + l_{y,\text{safe}} \le p_y^{(1)} &\le p_{y,\text{rel,max}}, \\ v_{y,\min} \le v_y^{(1)} &\le v_{y,\max}. \end{aligned}$$

The transition $\theta_{2,3}$ occurs in the guard set:

$$g(\theta_{2,3}) = \{x | p_x^{(2)} - p_x^{(1)} \le -l_{x,\text{safe}}\}.$$

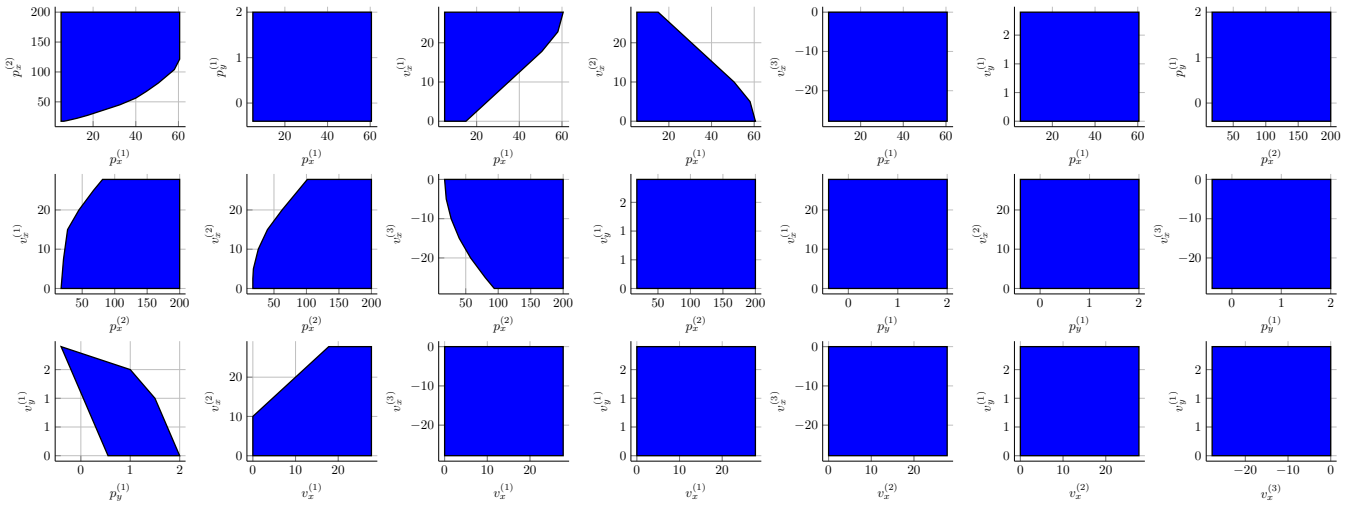

Fig. 5. Projections of an exemplary set controllable in $N = 5$ steps.

3) Location q_3: The invariant is defined by:

$$\begin{aligned} p_{x,\text{rel,min}} \le p_x^{(2)} - p_x^{(1)} \le -l_{x,\text{safe}}, \\ l_{x,\text{safe}} \le p_x^{(3)} - p_x^{(1)} \le p_{x,\text{rel,max}}, \\ p_{y,\text{min}} \le p_y^{(1)} \le p_{y,\text{max}}, \\ v_{y,\text{min}} \le v_y^{(1)} \le 0. \end{aligned}$$

Since location q_3 is the target location of the automaton, no transitions exist from here. Once the state of the hybrid automaton has reached the target set:

$$\mathcal{X}_\text{t} = \left\{x | x \in \text{inv}(q_3),\ 1.1 p_y^{(2)} \le p_y^{(1)} \le 0.9 p_y^{(2)},\ |v_y^{(1)}| \le 0.1\right\}$$

in location q_3, the maneuver is completed.

B. Resulting Sets and Trajectories

Given this definition of the hybrid automaton for the overtaking maneuver, it becomes possible to determine controllable sets for the target set $\mathcal{X}_\text{t}$ for different values of j. An exemplary projection of one of these originally 7-dimensional sets is shown in Fig. 5. Based on these sets, it can be guaranteed that the optimal control problem stated in Sec. II-C can be solved using a horizon length of $N = j$ steps for any given initial state x_0 included in a set $\mathcal{K}_j(q(x_0), x_0)$ (note that q_0 can be uniquely determined given x_0). In on-line operation, it only has to be checked in which set the current measured state x_0 is contained – the sets do not enter the optimal control problem.

In order to test our implementation, we randomly picked a large number of initial states, uniformly covering the controllable sets, and checked the feasibility of the corresponding optimal control problem. Exemplary position and velocity trajectories for the three vehicles resulting from one such initialization are given in Fig. 6 and Fig. 7. Clearly, Vehicle 1 overtakes Vehicle 2 and avoids collision with Vehicle 3. Even though the vehicles try to maintain their reference speed, vehicles 2 and 3 brake cooperatively in order to facilitate overtaking. While in this example, a rather tight longitudinal safety distance of $l_{x,\text{safe}} = 5\,\text{m}$ was prescribed between the vehicle center points, larger values can be chosen easily.

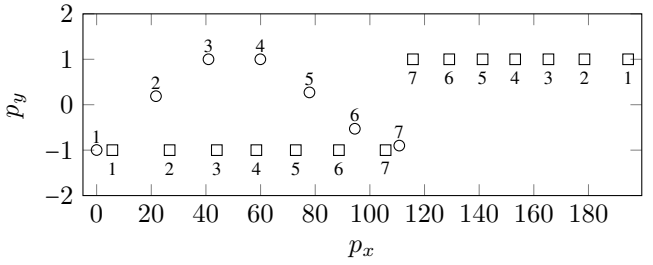

Fig. 6. Exemplary position trajectories (numbers indicate time steps j).

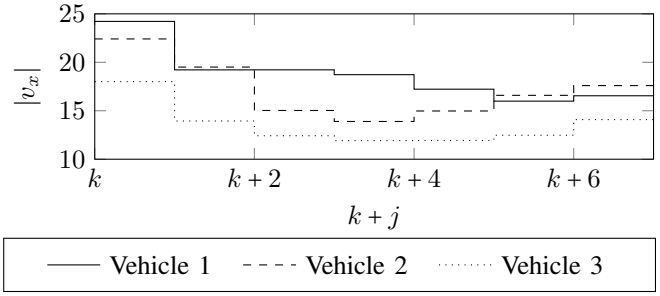

Fig. 7. Exemplary velocity magnitude trajectories.

C. Worst-case run-time

Having demonstrated the general operability of the proposed method, it remains to establish the reduction of the worst-case run-time of the optimization algorithm. We argue informally, based on the number of different combinations the values of the binary variables can assume in a specific problem formulation. The approach used in [1], [7], [8] is taken for comparison. It also relies on mixed-integer programming, but does not implement a hybrid automaton in order to avoid collisions of the vehicles. Instead, each vehicle is modeled as a bounding box which must not be

entered by other vehicles. This can be implemented using one binary variable per side of the bounding box. For the considered example scenario, a number of five binary variables (three for bounding vehicle 2, two for vehicle 3) is reasonable. Since no temporal constraints exist among these variables (arbitrary switches between two consecutive time instants are allowed), a total of $2^{(5 \cdot N)}$ different combinations exists. The proposed method, in contrast, only requires two binary variables to implement the optimal control problem, which are in addition connected by temporal constraints, namely, the switching order imposed by the set of allowed transitions Θ. Since the sequence in which the locations are traversed is known, only the switching times must be determined. For at most two switching times, there exist $\begin{pmatrix} N \\ 2 \end{pmatrix}$ possibilities, which is significantly less than the above mentioned $2^{(5 \cdot N)}$ combinations.

V. CONCLUSIONS

In this paper, it was shown that a novel type of maneuver automaton for trajectory planning of autonomous on-road vehicles provides several benefits when compared to existing approaches: on the one hand, using controllable sets, it allows to assess the feasibility of the planning problem in advance, together with the required time to complete the maneuver. On the other hand, the maneuver formulation was shown to reduce the worst-case complexity of the optimal control problem, further contributing to reliable operation.

In the future, we will focus on the following topics: regarding the implementation, the calculation of the controllable sets must be made more efficient, while conceptually, strategies for incorporating non-automated, non-communicating traffic participants into the framework are developed. Strategies to make the framework robust to uncertainties resulting from measurement noise and inter-vehicle communication are also of importance. The question of how many and which maneuvers are required to ensure sufficient coverage of the vehicles' action space will also be addressed.

ACKNOWLEDGMENT

We thank Jonas Bubenhagen for his help implementing the maneuvers. Support by the German Research Foundation (DFG) within the priority program (SPP) 1835 under grant no. STU 262/6-1 is gratefully acknowledged.

REFERENCES

[1] T. Schouwenaars, B. De Moor, E. Feron, and J. How, "Mixed integer programming for multi-vehicle path planning," in *European Control Conf.*, IEEE, 2001, pp. 2603–2608.

[2] E. Frazzoli, M. A. Dahleh, and E. Feron, "A hybrid control architecture for aggressive maneuvering of autonomous helicopters," in *38th IEEE Conf. on Decision and Control*, IEEE, vol. 3, 1999, pp. 2471–2476.

[3] ——, "Maneuver-based motion planning for nonlinear systems with symmetries," *IEEE Trans. on Robotics*, vol. 21, no. 6, pp. 1077–1091, 2005.

[4] D. Heß, M. Althoff, and T. Sattel, "Formal verification of maneuver automata for parameterized motion primitives," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, IEEE, 2014, pp. 1474–1481.

[5] B. Schürmann and M. Althoff, "Convex interpolation control with formal guarantees for disturbed and constrained nonlinear systems," in *Proc. Hybrid Systems: Computation and Control*, 2017, pp. 121–130.

[6] S. M. LaValle, *Planning algorithms*. Cambridge University Press, 2006.

[7] X. Qian, F. Altché, P. Bender, C. Stiller, and A. de La Fortelle, "Optimal trajectory planning for autonomous driving integrating logical constraints: An miqp perspective," in *19th Int. Conf. on Intelligent Transportation Systems*, IEEE, 2016, pp. 205–210.

[8] J. Eilbrecht and O. Stursberg, "Cooperative driving using a hierarchy of mixed-integer programming and tracking control," in *IEEE Intelligent Vehicles Symposium*, IEEE, 2017, pp. 673–678.

[9] O. Stursberg and S. Lohmann, "Synthesizing safe supervisory controllers for hybrid nonlinear systems," in *17th IMACS World Congress*, 2005.

[10] J. Sprinkle, J. M. Eklund, and S. S. Sastry, "Deciding to land a uav safely in real time," in *American Control Conference*, IEEE, 2005, pp. 3506–3511.

[11] A. Bayen, I. Mitchell, M. Oishi, and C. Tomlin, "Aircraft autolander safety analysis through optimal control-based reach set computation," *J. of Guidance, Control, and Dynamics*, vol. 30, no. 1, p. 68, 2007.

[12] J. Ding, J. Sprinkle, S. S. Sastry, and C. J. Tomlin, "Reachability calculations for automated aerial refueling," in *47th IEEE Conf. on Decision and Control*, IEEE, 2008, pp. 3706–3712.

[13] J. H. Gillula, H. Huang, M. P. Vitus, and C. J. Tomlin, "Design of guaranteed safe maneuvers using reachable sets: Autonomous quadrotor aerobatics in theory and practice," in *Int. Conf. on Robotics and Automation*, IEEE, 2010, pp. 1649–1654.

[14] S. Manzinger, M. Leibold, and M. Althoff, "Driving strategy selection for cooperative vehicles using maneuver templates," in *IEEE Intelligent Vehicles Symposium*, IEEE, 2017, pp. 647–654.

[15] O. Stursberg and B. H. Krogh, "Efficient representation and computation of reachable sets for hybrid systems," in *Int. Workshop on Hybrid Systems: Computation and Control*, Springer, 2003, pp. 482–497.

[16] IBM, *IBM ILOG CPLEX optimization studio*.

[17] Gurobi Optimization, Inc., *Gurobi optimizer*.

[18] F. Borrelli, A. Bemporad, and M. Morari, *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[19] T. X. T. Dang, "Verification and synthesis of hybrid systems," PhD thesis, Institut National Polytechnique de Grenoble, 2000.

[20] M. Herceg, M. Kvasnica, C. Jones, and M. Morari, "Multi-Parametric Toolbox 3.0," in *European Control Conf.*, 2013, pp. 502–510.