Optimal Control of Sets of Solutions to Formally Guarantee Constraints of Disturbed Linear Systems

Bastian Schürmann and Matthias Althoff

Abstract— **Optimal control finds an optimal input trajectory which steers an initial state to a desired final state while satisfying given state and input constraints. However, most efficient approaches are restricted to a single initial state. In this paper, we present a new approach, which combines reachability analysis with optimal control. This enables us to solve the optimal control problem for a whole set of initial states by optimizing over the set of all possible solutions. At the same time, we are able to provide formal guarantees for the satisfaction of state and input constraints. Taking the effects of sets of disturbances into account ensures that the resulting controller is robust against them, which is a big advantage over many existing approaches. We show the applicability of our approach with a vehicle-platoon example.**

I. Introduction

Reach-avoid problems are an important yet challenging control task. Many modern control scenarios can be viewed as reach-avoid problems, in which the system must be controlled close to a given final state at a fixed final time while avoiding unsafe regions, both in the state and input spaces. One example is autonomous vehicles, which must drive to given positions while meeting input constraints. They must also avoid other traffic participants and leave no road boundaries. Another example is the collaboration between robots and humans: the robots must move or handle objects at exact positions without hitting the humans or surrounding objects.

In control theory it is common to use linear systems to model a wide variety of real-world systems. While the systems are nonlinear, in many cases linear models capture the real behavior quite well. However, if one actually wants to ensure constraint satisfaction for real systems, one has to account for the model mismatch as well as for disturbances and sensor noise. Therefore, disturbed linear systems are a very useful class of systems when considering safety-critical reach-avoid problems, in which the disturbances take all previously mentioned errors into account.

In order to solve a reach-avoid problem, many different methods exist. The fastest methods involve computing optimal open-loop trajectories for a single initial state and not considering disturbances. To solve this problem for a whole set of initial states, one would have to solve infinitely many optimal control problems, as there are infinitely many states in a continuous set, which is infeasible. However, one often has to consider a whole set of states to either account for sensor inaccuracies or to make use of offline computations when the initial state is not exactly known. Moreover, it is not enough to find an optimal solution of the nominal case, since disturbances might lead to constraint violations or final states far from the planned solution. To solve this problem, we consider a novel optimal control approach, where we directly optimize over a reachable set instead of a single trajectory. Therefore we also do not optimize the input trajectory for a single state, but for a whole initial set. This new approach is formally correct and guarantees that the final state is reached in a fixed time while satisfying input and state constraints despite disturbances.

Many existing approaches either become too complex for larger dimensional systems or do not provide optimality or guarantees. For simple systems, one can try to solve the Hamilton-Jacobi-Bellman (HJB) equation or use dynamic programming [4], [6], [20], [17] to obtain an optimal controller which takes constraints into account. However, an analytic solution of the HJB equation is only possible for relatively simple and small dimensional systems, and it becomes difficult to use for more complex and disturbed systems. For linear systems in particular, there exist several methods [7], [8] to systematically compute the optimal feedback control law for different regions in the state space depending on the goal region and the convex state and input constraints. These techniques are often used for explicit model predictive control (MPC) [7], [8]. However, since they have to divide the state space in different regions, this can easily become computationally intractable if the number of dimensions and constraints grows. It can often lead to combinatorial and computational explosions, especially if disturbance effects have to be taken into account. This curse of dimensionality is a common problem for techniques which rely on discretizing the state and input spaces, such as most abstraction-based control approaches [19], [16], [28], which are able to take complex specifications into account.

Other approaches optimize a single trajectory and control all states from an initial set around this trajectory. This can be achieved by using linear quadratic regulators (LQR) [18]; however, they cannot take state or input constraints into account. To overcome this problem, more complex methods are developed in [29], [21], where the authors use sums-of-squares techniques to find special LQR tracking controllers. These controllers are then used to compute so-called LQR trees. Other methods which stabilize a system around a trajectory include tube-based MPC [23], [24] which also uses an additional feedback controller, and the approach shown in [15], which relies on so-called trajectory robustness. While the latter technique only works for undisturbed systems,

The authors are with the Department of Informatics, Technische Universität München, Boltzmannstr. 3, 85748 Garching, Germany. bastian.schuermann@tum.de, althoff@in.tum.de

tube-based MPC loses optimality by using a fixed feedback controller, which is not optimized further, even though some approaches allow adapting the tube size during optimization [25].

In previous works [26], [27], we considered the same problem as in this paper and solved it by repeatedly interpolating optimal open-loop inputs. In [26] this was done for the extreme points using convex combinations and for better scalability for the generators of a parallelotope in [27]. While [26], [27] work even for nonlinear systems, in contrast to the approach in this paper, they do not provide a continuous feedback law.

One application area which can benefit from our new algorithm is the design of maneuver automata [11], [14], [21], where trajectory pieces, or so-called motion primitives, are computed offline, stored in a maneuver automaton, and then combined online. Classical maneuver automata approaches [11] do not provide formal guarantees, whereas other approaches provide such guarantees either by using reachability analysis [14] or sums-of-squares programming [21]. The method of [14] is illustrated in Fig. 1: The motion primitives are computed and verified offline ① before they are combined as states in a maneuver automaton. There exists a transition from one motion primitive to another if the reachable set after a fixed time of the first motion primitive is completely contained in the initial set of the second motion primitive ②. In order to have highly connected maneuver automata, new controller design techniques are required which steer an initial set into a small final set while satisfying the constraints. The proposed optimal control strategy provides a solution to this problem. The offline computed and verified motion primitives are then combined in a planning algorithm online ③-⑥.

The remainder of this paper is organized as follows: We start with a formal problem formulation in Sec. II. This is followed by Sec. III, the main section of this paper, which contains the novel set-based optimal control method. Its applicability is shown in a numerical example in Sec. IV. The paper concludes with a summary and an outlook in Sec. V.

II. Problem Formulation

We consider a disturbed, linear, time-invariant (LTI) system of the form

$$\dot{x}(t) = Ax(t) + Bu(t) + w(t), \tag{1}$$

where $x(t) \in \mathbb{R}^n$ is the state vector, $u(t) \in \mathbb{R}^m$ is the vector of controllable inputs, and $w(t) \in \mathcal{W} \subset \mathbb{R}^n$ is the vector of uncontrollable inputs, i.e., additive disturbances. The system dynamics are defined by the matrices $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$. We do not require any stochastic properties for the disturbances nor that it can be measured, only that the set $\mathcal{W}$ is compact, i.e., closed and bounded. Before we formulate the problem statement, let us define the solution of (1) as:

Definition 1 (Solution) *A continuous curve $\xi : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}_0^+ \to \mathbb{R}^n$ is called a solution of system* (1) *for a given initial state $x(0) \in \mathbb{R}^n$, a given input function $u : \mathbb{R}_0^+ \to \mathbb{R}^m$, and disturbances $w : \mathbb{R}_0^+ \to \mathcal{W} \subset \mathbb{R}^n$, if the following two properties hold:*

1) $\xi(x(0), u(\cdot), w(\cdot), 0) = x(0)$
2) $\dot{\xi}(x(0), u(\cdot), w(\cdot), t) = A\xi(x(0), u(\cdot), w(\cdot), t) + Bu(t) + w(t), \forall t \in \mathbb{R}_0^+$.

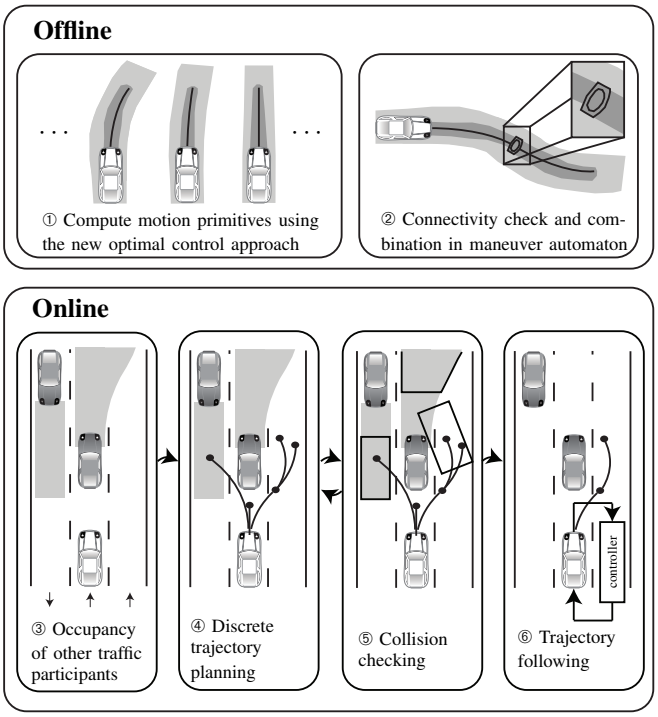


Fig. 1. Overview of robust maneuver automata design for an example in automated driving using our optimal control approach.

For a trajectory $x : \mathbb{R}_0^+ \to \mathbb{R}^n$ we refer by $x(\cdot)$ to the whole trajectory and by $x(t)$ to the value of the trajectory at time t. Moreover, we use the shorthand notation $x(\cdot) \in \mathcal{X}$ for $x(t) \in \mathcal{X}, \forall t \in [0, t_f]$, where $t_f \in \mathbb{R}_0^+$ is the final time. We use the same shorthand for input trajectories and disturbances. Sometimes, when we consider undisturbed LTI systems, we use $\xi(x(0), u(\cdot), 0, \cdot)$ to denote the solution without disturbances, i.e., $\mathcal{W} = 0$.

Problem 1 *The task is to find a control algorithm $u_{control}(x, t)$ for system* (1) *which guarantees that all states in an initial set $\mathcal{S}_{init} \subset \mathbb{R}^n$ are steered into a final set $\mathcal{S}_f \subset \mathbb{R}^n$ as close as possible to a given end state $x^{(f)}$ after time t_f, despite the disturbance set $\mathcal{W}$. With "as close as possible", we refer to the solution of*

$$\min_{u_{control}} \rho(\mathcal{S}_f, x^{(f)}),$$

where $\rho(\mathcal{S}_f, x^{(f)}) \to \mathbb{R}_0^+$ is a cost function measuring the distance of the states in $\mathcal{S}_f$ to $x^{(f)}$. Furthermore, we consider convex constraints on the states and inputs, i.e.,

$$\xi(x(0), u(\cdot), w(\cdot), \cdot) \in \mathcal{X} \subseteq \mathbb{R}^n,$$
$$u(\cdot) \in \mathcal{U} \subseteq \mathbb{R}^m,$$

where $\mathcal{X}$ and $\mathcal{U}$ are convex sets.

The proposed method also works with minor adaptations for other cost functions. Moreover, if it is necessary to end in a given set, i.e., if the reach-avoid problem contains ending in a pre-specified set $\mathcal{S}_{f,desired}$ rather than in a final set which is as small as possible, we can easily take this into account by adding the constraint

$$\mathcal{S}_f \subseteq \mathcal{S}_{f,desired}. \tag{2}$$

Clearly, a bad choice of constraints and final time may make it impossible to satisfy all constraints at the same time.

We consider convex state constraints, since most static constraints, such as speed limits and street boundaries (in Frenet coordinates) in the previous car example, can be expressed as convex state constraints. Nonconvex constraints mostly appear as dynamical constraints, such as other traffic participants in automatic driving, which are only known during runtime. Therefore, we cannot take them into account during the offline computation of motion primitives, which is considered in this paper, but rather during online planning. The proposed methods also work for nonconvex constraints; however, checking the constraints becomes much harder to compute.

III. Optimal, Robust Controller Synthesis Using Reachability Analysis

In practice, reach-avoid problems are often solved by combining optimal open-loop control with tracking control. Thereby, a reference trajectory is computed for a single initial point using efficient numerical tools, e.g., linear programming or quadratic programming algorithms [9]. To take advantage of these efficient algorithms, one has to consider discrete-time systems, i.e., one has to restrict the inputs to piecewise-constant input trajectories. While these algorithms are able to take constraints and final times into account, they are only able to compute the solution for a single initial state. Therefore, if we want to control a whole set of initial states, we would have to solve infinitely many optimal control problems, which is not possible. At the same time, open-loop optimal control is not robust against disturbances.

Therefore one often combines the optimized reference trajectory with a tracking controller, which should drive the error between the actual state and the desired state along the reference trajectory to zero. An often used controller type is LQR [18], which computes for an undisturbed linear system a feedback matrix K, which minimizes a quadratic cost function on inputs and states. While the feedback controllers are able to control all states starting in an initial set and can also react to disturbances, they are not able to take input or state constraints into account.

In this section, we present a way to overcome the previously described problems by including reachability analysis in the controller synthesis. Our method optimizes the reference trajectory and feedback matrices such that they take state and input constraints into account, despite disturbances, and directly minimize the reachable set at the final time.

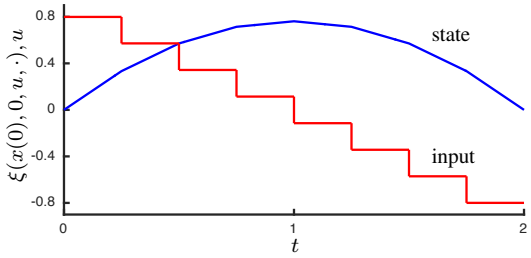
Let us first illustrate the main idea of our new concept. In Fig. 2(a), we show a classic optimal control approach, in which an optimal input trajectory $u(\cdot)$ for a single initial state $x(0)$ is computed. This input trajectory minimizes the difference between the final state of the center trajectory $\xi(x(0,u(\cdot),0,t_f))$ and the desired final state $x^{(f)}$. Our new approach is shown in Fig. 2(b), where we consider a whole set of initial states for which we compute the tracking controller

$$u_{track}(x(t),t) = u_{ref}(t) + K(t)(x(t) - x_{ref}(t)), \tag{3}$$

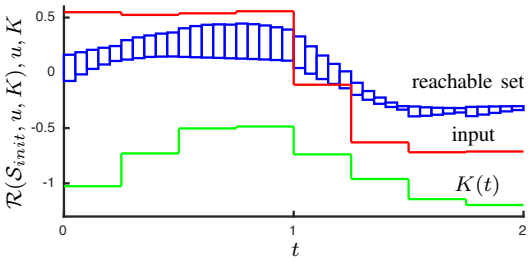
with

$$x_{ref}(t) = \xi(x_{ref}(0), u_{ref}(\cdot), 0, t), \tag{4}$$

combining the open-loop reference input $u_{ref}(t)$ and the time-varying feedback matrix $K(t)$. By solving the optimal control problem for the entire reachable set, we obtain optimal inputs for all initial states. For a simpler demonstration and notation, we restrict our consideration to piecewise-constant control inputs and feedback matrices. All presented methods can be easily adapted to general piecewise-continuous input functions.



(a) Classical, open-loop optimization



(b) Novel, set-based closed-loop optimization

Fig. 2. Illustration of our control approach: While classical optimal open-loop control optimizes a single input trajectory for a single initial state (a), our new approach optimizes a reference input and a feedback matrix to minimize the solutions of a whole set of initial states (b). To better illustrate the approach, the plots do not show the same optimization problem.

Before we present our new optimization technique, let us formally introduce reachable sets.

Definition 2 (Reachable Set) *For system* (1)*, the reachable set* $\mathcal{R}_{t,u,\mathcal{W}}(\mathcal{S}) \subset \mathbb{R}^n$ *for a time* t*, an input function* $u : \mathbb{R}_0^+ \to \mathbb{R}^m$*, disturbances* $w(\cdot) \in \mathcal{W}$*, and an initial set* $\mathcal{S} \subset \mathbb{R}^n$ *is the set of final states of trajectories starting in* $\mathcal{S}$ *after time* t*, i.e.,*

$$\mathcal{R}_{t,u,\mathcal{W}}(\mathcal{S}) = \{x(t) \in \mathbb{R}^n | \exists x(0) \in \mathcal{S}, \exists w(\cdot) \in \mathcal{W} : \xi(x(0), u(\cdot), w(\cdot), t) = x(t)\}.$$

The reachable set over a time interval $[t_1, t_2]$ is the union of all reachable sets for these points in time, i.e.,

$$\mathcal{R}_{[t_1,t_2],u,\mathcal{W}}(\mathcal{S}) = \bigcup_{t\in[t_1,t_2]} \mathcal{R}_{t,u,\mathcal{W}}(\mathcal{S}).$$

For an easier presentation and without loss of generality, we divide the time interval $[0, t_f]$ into N intervals of length $\Delta t = \frac{t_f}{N}$. Within each time interval, $u_{ref}(\cdot)$ and $K(\cdot)$ are constant. Let us denote with $t_i = i\Delta t, i \in \{0, \dots, N-1\}$ and therefore $u_{ref}(t_i)$ and $K(t_i)$ the $(i+1)-th$ piecewise constant input and matrix, respectively.

We solve Problem 1 by optimizing these piecewise constant reference inputs $u_{ref}(\cdot)$ and feedback matrices $K(\cdot)$ in the following optimization problem:

$$\min_{u_{track}} \rho(\mathcal{R}_{t_f,u_{track},\mathcal{W}}(\mathcal{S}_{init}), x^{(f)}) \tag{5}$$

w.r.t.

$$\mathcal{R}_{[0,t_f],u_{track},\mathcal{W}}(\mathcal{S}_{init}) \subseteq \mathcal{X}, \tag{6}$$

$$u_{track}\left(\mathcal{R}_{[0,t_f],u_{track},\mathcal{W}}(\mathcal{S}_{init})\right) \subseteq \mathcal{U}. \tag{7}$$

with

$$\rho(\mathcal{R}(\cdot), x^{(f)}) = \frac{1}{2}\|\mathcal{R}(\cdot)\|_1 + \|x^{(f)} - center(\mathcal{R}(\cdot))\|_1.$$

Therein, we use $u_{track}\left(\mathcal{R}(\cdot)\right)$ as a shorthand for the set of inputs which result if we evaluate u_{track} for each point in the set $\mathcal{R}(\cdot)$, i.e.

$$u_{track}(\mathcal{R}(\cdot)) = \{u_{track}(x) | x \in \mathcal{R}(\cdot)\}.$$

For a set $\mathcal{S} \subset \mathbb{R}^n$ we denote by $\|\mathcal{S}\|_1$ the sum of the edges of the axis-aligned bounding box, i.e.,

$$\|\mathcal{S}\|_1 = \sum_{i=1}^{n} (\sup_{x\in\mathcal{S}} x_i - \inf_{x\in\mathcal{S}} x_i),$$

and by $center(\mathcal{S})$ its center. Therein, x_i refers to the $i-$th entry of vector x. The cost function $\rho(\mathcal{R}(\cdot), x^{(f)})$ penalizes the size of the final set in every dimension and the distance from the center of the final set to the desired final state. This is equivalent to the cost function

$$\rho(\mathcal{R}(\cdot), x^{(f)}) = \sum_{i=1}^{n} \sup\left(|x_i - x_i^{(f)}| \big| x \in \mathcal{R}(\cdot)\right).$$

The advantage of this choice of cost function is that the axis-aligned bounding box is easy to compute in contrast to e.g., a cost function based on extreme points, as we see later.

The optimal control problem in (5) can be written in the standard form of nonlinear programming problems:

$$\min_{u_{track}} Costs(u_{track}(\cdot), A, B, \mathcal{W}, \mathcal{S}_{init}, x^{(f)}, t_f) \tag{8}$$

$$\text{w.r.t. } Constraints(u_{track}(\cdot), A, B, \mathcal{X}, \mathcal{U}, \mathcal{W}, \mathcal{S}_{init}, t_f) \leq 0, \tag{9}$$

where $Costs$ and $Constraints$ represent the cost and constraint functions, which we describe later. This allows us to use nonlinear programming tools [5] to efficiently solve the problem. Let us describe in the following how our problem can be transformed into such a problem and how the different functions can be implemented. Each optimization step consists of three major parts: the reachability analysis, the cost function, and the constraint function. To improve the optimization, we obtain an initial guess for the reference input $u_{ref}(\cdot)$ by solving a standard optimal control problem starting at the center of the initial set and with the nominal system, i.e., without considering the disturbances. The solution K_{LQR} of the Riccati equation for the infinite time-horizon case is used as the initial guess for all feedback matrices $K(t_i) = K_{LQR}, \forall i \in \{0, \dots, N-1\}$.

1) Reachability Analysis: For linear systems, the computation of overapproximative reachable sets is well understood (see [1, Sec. 3.2], [10], [13], [3]), and there exist efficient tools, e.g., the Matlab toolbox CORA [2] and SpaceEx [12]. The algorithms in [1] and [2] use zonotopes as set representation for reachability computation:

Definition 3 (Zonotope) *A set is called a zonotope if it can be written as*

$$Z = \left\{ x \in \mathbb{R}^n \middle| x = c + \sum_{i=1}^{p} \alpha_i g^{(i)}, \alpha_i \in [-1, 1] \right\}.$$

Therein $c \in \mathbb{R}^n$ defines the center of the zonotope, and $g^{(i)} \in \mathbb{R}^n, i \in \{1, \dots, p\}$, are $p = o\,n$ generators, with o denoting the order of the zonotope. Sometimes, we use $< c, g^{(1)}, \dots, g^{(p)} >$ as a more concise notation of Z. The generators can also be combined to the generator matrix $G \in \mathbb{R}^{n\times p}$, which contains the generators as its columns.

A zonotope with n linear independent generators is called a parallelotope.

To enable the use of the reachability analysis as in [1], we have to express our closed-loop dynamics in closed-form. We obtain this by inserting (3) and (4) into (1) and by combining the actual system state with the state of the reference trajectory x_{ref} into an extended state $\hat{x}$:

$$\underbrace{\begin{bmatrix} \dot{x}(t) \\ \dot{x}_{ref}(t) \end{bmatrix}}_{\dot{\hat{x}}} = \underbrace{\begin{bmatrix} A + BK(t) & -BK(t) \\ \mathbf{0}_{n\times n} & A \end{bmatrix}}_{\hat{A}} \underbrace{\begin{bmatrix} x(t) \\ x_{ref}(t) \end{bmatrix}}_{\hat{x}} + \underbrace{\begin{bmatrix} B & \mathbf{0}_{n\times m} \\ \mathbf{0}_{n\times m} & B \end{bmatrix}}_{\hat{B}} \underbrace{\begin{bmatrix} u_{ref}(t) + w(t) \\ u_{ref}(t) \end{bmatrix}}_{\hat{u}},$$

where $\mathbf{0}_{n\times m} \in \mathbb{R}^{n\times m}$ is a matrix which contains only zeros. The reachability algorithm in [1] computes the reachable sets for time intervals with constant system dynamics. Since we change our feedback matrices $K(\cdot)$ after each time step Δt, the extended system matrix $\hat{A}$ changes after each step as well. After computing the reachable sets for time interval $[t_i, t_{i+1}]$, we change the dynamics and reference inputs and compute the next reachable set with the new dynamics and reference input, using the reachable set from the last step as the initial set. For the first step, we use the extended initial set $\hat{\mathcal{S}}_{init} = \{\hat{x} \in \mathbb{R}^{2n} | x \in \mathcal{S}_{init}, x_{ref} = center(\mathcal{S}_{init})\}$ as the starting set. Let us now summarize the reachability analysis for each step.

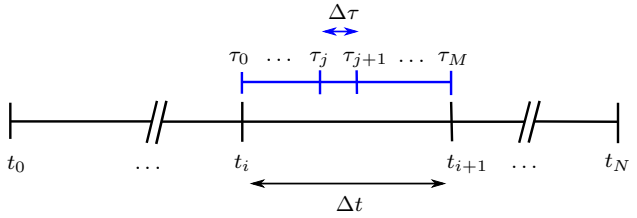


Fig. 3. Illustration of the two time scales: Δt for the control inputs and $\Delta\tau$ for the reachability analysis.

The algorithm in [1] divides the time in small time intervals $[\tau_k, \tau_{k+1}]$ of length $\Delta\tau$. Without loss of generality, we assume that $\Delta t = M\Delta\tau$, $M \in \mathbb{N}$, i.e., the time steps are smaller than the time intervals in which we change the reference inputs u_{ref} and feedback matrices $K(\cdot)$. This is illustrated in Fig. 3. The algorithm uses the superposition principle to divide the reachable set into two parts: one with the autonomous dynamics and the reference input, and one resulting from the disturbance $\mathcal{W}$, i.e., $\xi(x(\tau_k), u_{ref}, w(\cdot), \Delta\tau) = \xi(x(\tau_k), u_{ref}, 0, \Delta\tau) + \xi(0, 0, w(\cdot), \Delta\tau)$. Given an initial set $\hat{\mathcal{S}}_{init}$, the reachable set of a time interval $\mathcal{R}_{[0,t_1],u,\mathcal{W}}(\hat{\mathcal{S}}_{init})$ is computed in four steps (see Fig. 4 for the first three steps). These steps involve the addition of sets ($\mathcal{A} \oplus \mathcal{B} := \{a + b | a \in \mathcal{A}, b \in \mathcal{B}\}$) and the multiplication of sets ($\mathcal{A} \otimes \mathcal{B} := \{a\, b | a \in \mathcal{A}, b \in \mathcal{B}\}$):

(i) Compute the reachable set for the first interval $[0, \tau_1]$ neglecting the disturbances: $\mathcal{R}^h_{\tau_1} = \mathcal{R}_{\Delta\tau,u_{ref},0}(\hat{\mathcal{S}}_{init}) = e^{\hat{A}\Delta\tau} \otimes \hat{\mathcal{S}}_{init} \oplus \int_0^{\Delta\tau} e^{\hat{A}(\Delta\tau - r)}\, \mathrm{d}r\, \hat{B} \begin{bmatrix} u_{ref} \\ u_{ref} \end{bmatrix}$.

(ii) Compute the convex hull $\mathrm{CH}(\hat{\mathcal{S}}_{init}, \mathcal{R}^h_{\tau_1})$ for the approximation within $[0, \tau_1]$.

(iii) Enlarge the reachable set by $\mathcal{R}^{\mathcal{W}}_{\Delta\tau} = \mathcal{R}_{\Delta\tau,0,\mathcal{W}}(\emptyset)$ to account for the disturbances and by $\mathcal{D}$ for the curvature of the trajectories (see [1] for details for $\mathcal{D}$) making the result overapproximative: $\mathcal{R}_{0,\tau_1} = \mathrm{CH}(\hat{\mathcal{S}}_{init}, \mathcal{R}^h_{\tau_1}) \oplus \mathcal{R}^{\mathcal{W}}_{\Delta\tau} \oplus \mathcal{D}$.

(iv) For all following time intervals $[\tau_k, \tau_{k+1}]$ compute $\mathcal{R}_{[\tau_k,\tau_{k+1}]} = e^{\hat{A}\Delta\tau} \otimes \mathcal{R}_{[\tau_{k-1},\tau_k]} \oplus \mathcal{R}^{\mathcal{W}}_{\Delta\tau} \oplus \int_0^{\Delta\tau} e^{\hat{A}(\Delta\tau - r)}\, \mathrm{d}r\, \hat{B} \begin{bmatrix} u_{ref} \\ u_{ref} \end{bmatrix}$.

We iteratively repeat this procedure over all time intervals from $t = [0, t_f]$, where we update the dynamics after each time step Δt. Note that we can also obtain the reachable sets at single points in time $\mathcal{R}_{\tau_k}$ with this technique.

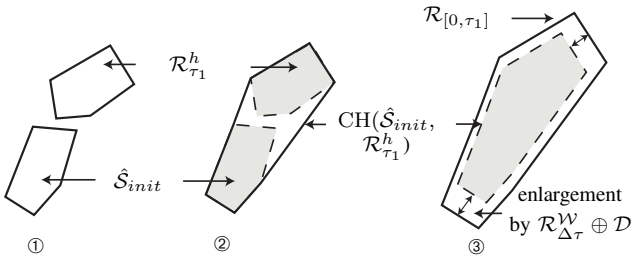


Fig. 4. Computation of the reachable set for the first time interval $[0, \tau_1]$.

2) Cost Function: In the cost function (8), we compute the reachable set for the whole time-horizon $[0, t_f]$ with the current controller $u_{track}(\cdot)$. We then compute the cost of the final reachable set $\mathcal{R}_{t_f}$:

$$cost = \frac{1}{2}\|\mathcal{R}_{t_f}\|_1 + \|x^{(f)} - c\|_1,$$

where c is the center of $\mathcal{R}_{t_f}$. The size of the axis-aligned bounding box for a zonotope is easily computed by

$$\|Z\|_1 = \| < c, g^{(1)}, \ldots, g^{(p)} > \|_1 = 2\sum_{i=1}^{n}\sum_{j=1}^{p} |g_i^{(j)}|.$$

Therefore, if we denote the final reachable set as $\mathcal{R}_{t_f} =< c, g^{(1)}, \ldots, g^{(p)} >$, the cost function can be calculated by:

$$cost = \sum_{i=1}^{n}\sum_{j=1}^{p} |g_i^{(j)}| + \sum_{i=1}^{n} |c_i - x_i^{(f)}|. \tag{10}$$

3) Constraint Function: In the constraint function (9), we check whether the overapproximation of the reachable set is always inside the state constraints and whether the input constraints are satisfied at all times. For each reachable set $\mathcal{R}_{[t_l+\tau_k,t_l+\tau_{k+1}]}$ we check whether $\mathcal{R}_{[t_l+\tau_k,t_l+\tau_{k+1}]} \subseteq \mathcal{X}$. To check the input constraints, we have to ensure that the inputs cannot be violated for any point in $\mathcal{R}_{[t_l+\tau_k,t_l+\tau_{k+1}]}$. We therefore compute the overapproximation of the set of possible inputs $u_{track}(\mathcal{R}_{[t_l+\tau_k,t_l+\tau_{k+1}]})$ in the time interval $[t_l + \tau_k, t_l + \tau_{k+1}]$ as

$$u_{track}(\mathcal{R}_{[t_l+\tau_k,t_l+\tau_{k+1}]}) \subseteq u_{ref}(t_l) \oplus K(t_l) \otimes (Z_x \oplus (-1 \otimes Z_{ref})),$$

where Z_x is the set of states in the first n coordinates of $\mathcal{R}_{[t_l+\tau_k,t_l+\tau_{k+1}]}$, i.e., the reachable states, and Z_{ref} the states of $n+1$ to $2n$ coordinates, i.e., the reference states. The computation of the input set contains only linear maps and set additions; both operators can be efficiently computed for zonotopes [1].

In order to use (6) and (7) inside a solver for nonlinear programming, we have to transform them from a set-based representation into a system of inequalities, see (9). The implementation for checking whether the reachable sets and input sets are inside the constraint sets depends on the representation of the state and input constraints. The most common constraints in practice are box constraints, i.e., where a maximal x_i^{max} and a minimal value x_i^{min} for each dimension is given. For this type of constraint, we check for each reachable set $\mathcal{R}_{[t_l+\tau_k,t_l+\tau_{k+1}]} =< c, g^{(1)}, \ldots, g^{(p)} >$ whether $\forall i \in \{1, \ldots, n\}$:

$$c_i + \sum_{j=1}^{p} |g_i^{(j)}| \leq x_i^{max}$$

$$c_i - \sum_{j=1}^{p} |g_i^{(j)}| \geq x_i^{min}$$

and therefore

$$c_i + \sum_{j=1}^{p} |g_i^{(j)}| - x_i^{max} \leq 0 \tag{11}$$

$$-c_i + \sum_{j=1}^{p} |g_i^{(j)}| + x_i^{min} \leq 0, \tag{12}$$

which is a set of inequalities smaller than or equal to zero as desired. We do the same for the input zonotopes $u_{track}(\mathcal{R}_{[t_l+\tau_k,t_l+\tau_{k+1}]})$ with u^{max} and u^{min}. For other types of constraints, there exist other similar, efficient methods, e.g., if the constraint sets are parallelotopes of the form $\mathcal{P} = \{x \in \mathbb{R}^n | x = c_\mathcal{P} + G_\mathcal{P}\alpha, \alpha_i \in [-1, 1]\}$, with $c_\mathcal{P} \in \mathbb{R}^n$, $G_\mathcal{P} \in \mathbb{R}^{n\times n}$ with full rank, then we can transform the parallelotopes into axis-aligned boxes by multiplying it with $G_\mathcal{P}^{-1}$ and use the previously described method. Another example is polytopic constraint sets in the form $P = \{x \in \mathbb{R}^n | Cx \leq d\}$, with $C \in \mathbb{R}^{q\times n}$, $d \in \mathbb{R}^q$. Since we can compute the potential extreme points $\hat{x}^{(i)}$ of a zonotope as the 2^p combinations of

$$\{\hat{x}^{(1)}, \ldots, \hat{x}^{(2^p)}\} = c \pm g^{(1)} \pm \cdots \pm g^{(p)},$$

we can check if $\forall i = 1, \ldots, 2^p$, $C\hat{x}^{(i)} \leq d$, which can also be expressed in one large system of inequalities, if needed. Due to the special structure of zonotopes, the computation of the extreme points is numerically robust in contrast to computing the extreme points of a polytope defined by a system of inequalities, for example.

Clearly, we can apply the described methods in the same way if we have constraints on the final set (2) by checking the constraints on $\mathcal{R}_{t_f}$.

Complexity of the Algorithm

As mentioned before, we use nonlinear programming tools to efficiently solve the optimization problem. Due to the structure of nonlinear programs, it is not possible to give bounds on the complexity for the optimization program. Since the optimization problem is nonconvex, we can only expect it to converge to local minima, and even this cannot be guaranteed. However, these are general problems one faces when using nonlinear programming to compute controllers for disturbed systems. As mentioned before, there are approaches which use convex methods like multi-parametric linear programming. However, the exponential complexity of these approaches due to dividing the state space often restricts their applicability to rather small dimensional systems.

What we can do, however, is discuss the complexity of one optimization step, i.e., to evaluate the cost function and the constraint function. In this case, we have the complexity for the reachability analysis, which scales with $\mathcal{O}(n^3)$ for the number of dimensions n, if we use the method described in [1]. After the reachability analysis, we compute the costs in (10), which scales with $\mathcal{O}(n^2)$ for a fixed zonotope order o. Lastly, we have to check the constraints. The complexity depends on the type of constraints. Box constraints as shown in (11)-(12) scale with $\mathcal{O}(n^2)$ if we have constraints in all dimensions. For parallelotope constraints, we must also multiply the inverse of a matrix which has complexity $\mathcal{O}(n^3)$. This is therefore also the resulting complexity for parallelotope constraints. For polytopic constraints however, we have to compute all extreme points of the zonotope. While this is numerically stable to compute, the number of potential extreme points, and therefore also the complexity, grows exponentially with the dimension n if we still assume a constant zonotope order o.

Therefore, for box and parallelotope constraints, the overall complexity for one step of our algorithm is bounded by $\mathcal{O}(n^3)$, while for polytopic constraints it becomes exponential.

If we wanted to optimize trajectories for the extreme points only, instead of using reachability analysis, we would be able to use more efficient algorithms, such as linear or quadratic programming; however, we would have to do it for 2^n extreme points, for the simple case of boxes, while not being able to consider disturbances. As mentioned in the beginning, other techniques which do consider disturbances often rely on dividing the state space into different sets, where the complexity also grows exponentially with the number of dimensions.

IV. Numerical Example

In this section, we show the applicability of the presented approach with a platooning example. In this example, vehicles on a highway are supposed to drive in a platoon behind each other. By driving close to the vehicle in front of them, they can save fuel and reduce driving time for human drivers. However, in this scenario there are important safety constraints which must be satisfied at all times, despite the effects of external disturbances. We consider vehicle-to-vehicle communication which allows a central controller design. This example is inspired by the benchmark example proposed in [22].

We consider a platoon with four vehicles, where the dynamics of each vehicle $i \in \{1, 2, 3, 4\}$ is given by

$$\dot{p}^{(i)} = v^{(i)}, \qquad \dot{v}^{(i)} = a^{(i)} + w^{(i)},$$

where $p^{(i)}$ denotes the position of the $i-$th vehicle, $v^{(i)}$ its velocity, and $a^{(i)}$ its acceleration, i.e., the controllable input. The disturbances, i.e., the uncontrollable inputs, are denoted by $w^{(i)}$. To model the whole platoon, we use the absolute states of the first vehicle and the relative states of the second, third, and fourth vehicle, i.e., we consider the eight-dimensional state vector $x = [p^{(1)}, v^{(1)}, p^{(1)}-p^{(2)}-c_s, v^{(1)}-v^{(2)}, p^{(2)}-p^{(3)}-c_s, v^{(2)}-v^{(3)}, p^{(3)}-p^{(4)}-c_s, v^{(3)}-v^{(4)}]^T$, the input vector $u = [a^{(1)}, a^{(2)}, a^{(3)}, a^{(4)}]$, and disturbance vector $w = [w^{(1)}, w^{(2)}, w^{(3)}, w^{(4)}]$. Therein $c_s \in \mathbb{R}^+$ denotes a safety constant, defining a minimal safe distance. The resulting dynamics are given by

$$\begin{aligned}
\dot{x}_1 &= x_2, & \dot{x}_2 &= u_1 + w_1,\\
\dot{x}_3 &= x_4, & \dot{x}_4 &= u_1 - u_2 + w_1 - w_2,\\
\dot{x}_5 &= x_6, & \dot{x}_6 &= u_2 - u_3 + w_2 - w_3,\\
\dot{x}_7 &= x_8, & \dot{x}_8 &= u_3 - u_4 + w_3 - w_4.
\end{aligned}$$

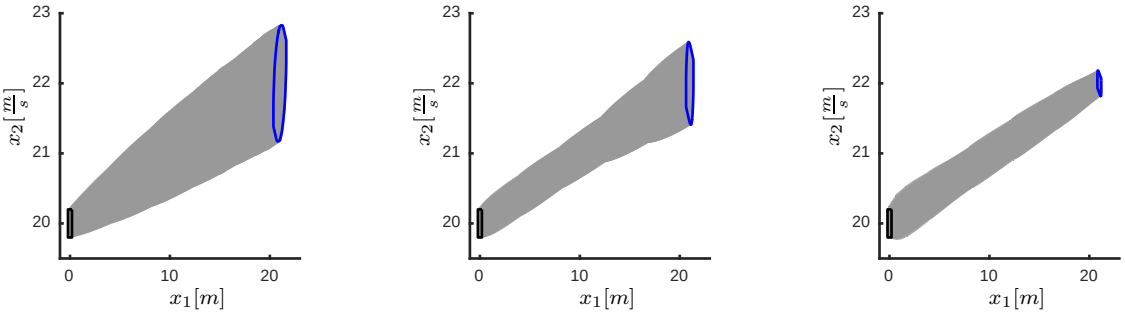


Fig. 5. Illustration of the reachable sets projected onto the (x_1, x_2) plane at different times during the optimization algorithm. On the left, the reachable set for the initial guess, in the middle during the optimization, and on the right the final reachable set after the optimization. The initial sets are shown in black and the final sets in blue.

We assume that all inputs are constrained between $u_i(\cdot) \in [-10, 10]\frac{m}{s}, i \in \{1, 2, 3, 4\}$ and all disturbances vary freely in the interval $w_i(\cdot) \in [-1, 1]\frac{m}{s}$. Moreover, we have the state constraint that the vehicles must keep the minimal safety distance, i.e., $x_3, x_5, x_7 > 0$. We consider the following scenario, which can be used as a motion primitive in a platooning maneuver automaton: The vehicles start with initial states ranging freely in the box $[-0.2, 0.2]m \times [19.8, 20.2]\frac{m}{s} \times [0.8, 1.2]m \times [-0.2, 0.2]\frac{m}{s} \times [0.8, 1.2]m \times [-0.2, 0.2]\frac{m}{s} \times [0.8, 1.2]m \times [-0.2, 0.2]\frac{m}{s}$, i.e., the vehicles drive with different velocities around $20\frac{m}{s}$ behind each other. We consider a final state $[21m, 22\frac{m}{s}, 1m, 0\frac{m}{s}, 1m, 0\frac{m}{s}, 1m, 0\frac{m}{s}]^T$ which should be reached after $1s$, i.e., the whole platoon should speed up to $22\frac{m}{s}$ and align in a safe distance of $c_s + 1m$ between each vehicle.

We use our approach to compute an optimized tracking controller using the approaches presented in Sec. III. The center trajectory is split in five time steps and therefore five constant control matrices are computed. We implement the controller in Matlab and use the CORA toolbox [2] for the reachability computation, where the disturbances are handled as uncontrollable inputs. For the optimization we use Matlab's `fmincon` function with the `active-set` algorithm. The computations are performed on a computer with a 3.1 GHz dual-core i7 processor with 16 GB memory. The computation of the optimized controller takes around five minutes. The optimization algorithm terminates after it reaches the maximal number of iterations and finds a feasible solution. We show the reachable sets at different times during the algorithm in Fig. 5. We see how the reachable set changes between different iterations, as it is directly optimized on, and how the size of the final set is minimized. In Fig. 6, we show the optimized final set in comparison to the initial set for the different dimensions. For the x_1 and x_2 coordinates we shifted the final reachable set by the final states $x^{(f)}$ for a better comparison. We see that the shifted reachable set of our controller (blue) lies completely in the initial set (black).

For comparison, we also compute three LQR tracking controllers using the traditional approach of a reference trajectory in combination with a LQR tracking controller as described in the beginning of Sec. III. Since we value all dimensions the same, we choose the state weighting matrix as the identity $Q = I$. We obtain the first LQR controller with an input weighting matrix $R = I$. We use this controller as an initial guess for our optimization problem; therefore, the reachable set can be seen in Fig. 5 on the left. The final set is very large and while it satisfies the input constraints, it violates the state constraints. To get an idea of how well a LQR controller can solve the problem, we increase the state weighting matrices manually while making sure that the state and input constraints are still satisfied. We see the resulting final set for $Q = 90I$ and $R = I$ in red, dashed in Fig. 6. It is much larger than with our controller and lies outside the initial set. We can keep increasing the state weighting matrices until $Q = 700I$, for which the final reachable sets (red, solid in Fig. 6) lie just inside the initial set; however, this controller uses more than two times of the allowed inputs. Even if we weight the positions and velocities in the state weighting matrix differently, any resulting controller which controls all final states inside the initial set uses inputs which are much larger than allowed.

In conclusion, we are not able to find an LQR controller which satisfies the constraints and lies in the final set. This example shows how using the reachability analysis inside the optimization problem leads to controllers with better performance and guaranteed constraints satisfaction, which is not possible with classical LQR controllers.

V. Conclusion and Future Work

Reach-avoid problems are an important task in control theory. In this paper, we provided a solution to this problem by extending existing approaches for optimal tracking control. To the best of our knowledge, we use, for the first time, reachability analysis inside the optimization problem to obtain optimal control inputs, not only for a single state, but for a whole continuous set of initial states. This new technique allows us to take constraints and the effects of disturbances into account. The resulting controller is robust against disturbances, and we obtain formal guarantees for the satisfaction of constraints and the resulting reachable set. By computing the reachable set inside the optimization problem, we are able to directly optimize the size of the reachable set at a final time point. We show the applicability of our approach for a platooning example, where we also

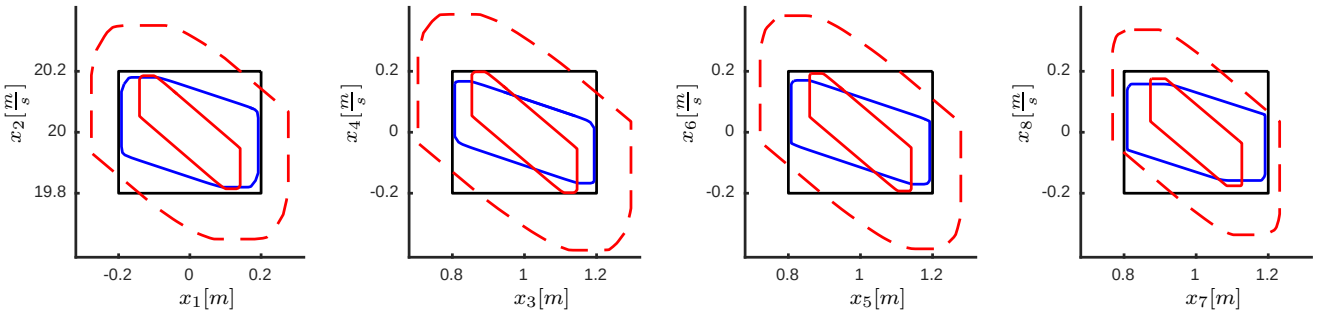


Fig. 6. Initial (black) and shifted final sets (blue) for our optimized controller u_{track}, projected onto the (x_1, x_2), the (x_3, x_4), the (x_5, x_6), and the (x_7, x_8) planes. For comparison the final sets of two LQR controllers are shown (red).

demonstrate its advantages (minimizing the final set while satisfying constraints) over a classical approach.

This paper is only the first step in the combination of optimal control and reachability analysis. Therefore, there are many future extensions possible. An important direction is to optimize the algorithms for reachability analysis in order to solve the optimization problems more quickly. This is especially important for nonlinear systems, where the current computation times of the reachability analysis prevent their use in optimization. By improving the computation times for linear systems, online application of this approach might become possible, which would offer extensions to reachability-based MPC. In addition to these extensions, we plan to apply the algorithms to the control of real systems.

Acknowledgments

The author gratefully acknowledges financial support from the European Commission project UnCoVerCPS under grant number 643921.

References

[1] M. Althoff. *Reachability Analysis and its Application to the Safety Assessment of Autonomous Cars*. PhD thesis, Technische Universität München, 2010.

[2] M. Althoff. An introduction to CORA 2015. In *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems*, pages 120–151, 2015.

[3] E. Asarin, T. Dang, G. Frehse, A. Girard, C. Le Guernic, and O. Maler. Recent progress in continuous and hybrid reachability analysis. In *Proc. of the IEEE Conference on Computer Aided Control Systems Design*, pages 1582–1587, 2006.

[4] D. P. Bertsekas. *Dynamic Programming and Optimal Control*, volume 1. Athena Scientific Belmont, MA, 3rd edition, 2005.

[5] J. T. Betts. *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*. Society for Industrial and Applied Mathematics, 2010.

[6] F. Blanchini and S. Miani. *Set-Theoretic Methods in Control*. Springer, 2008.

[7] F. Borrelli. *Constrained Optimal Control of Linear and Hybrid Systems*. Springer, 2003.

[8] F. Borrelli, A. Bemporad, and M. Morari. *Predictive Control for linear and hybrid systems*. Cambridge University Press, 2011/2015.

[9] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[10] A. Chutinan and B. H. Krogh. Computational techniques for hybrid system verification. *IEEE Transactions on Automatic Control*, 48(1):64–75, 2003.

[11] E. Frazzoli, M. Dahleh, and E. Feron. Maneuver-based motion planning for nonlinear systems with symmetries. *IEEE Transactions on Robotics*, 21(6):1077–1091, 2005.

[12] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler. SpaceEx: Scalable verification of hybrid systems. In *Proc. of the International Conference on Computer Aided Verification*, 2011.

[13] A. Girard and C. Le Guernic. Efficient reachability analysis for linear systems using support functions. In *Proc. of the 17th IFAC World Congress*, pages 8966–8971, 2008.

[14] D. Heß, M. Althoff, and T. Sattel. Formal verification of maneuver automata for parameterized motion primitives. In *Proc. of the International Conference on Intelligent Robots and Systems*, pages 1474–1481, 2014.

[15] A. A. Julius and A. K. Winn. Safety controller synthesis using human generated trajectories: Nonlinear dynamics with feedback linearization and differential flatness. In *Proc. of the American Control Conference*, pages 709–714, 2012.

[16] M. Kloetzer and C. Belta. A fully automated framework for control of linear systems from temporal logic specifications. *IEEE Transactions on Automatic Control*, 53(1):287–297, 2008.

[17] A. B. Kurzhanski, I. M. Mitchell, and P. Varaiya. Optimization techniques for state-constrained control and obstacle problems. *Journal of Optimization Theory and Applications*, 128(3):499–521, 2006.

[18] H. Kwakernaak and R. Sivan. *Linear optimal control systems*. Wiley-Interscience New York, 1972.

[19] J. Liu, U. Topcu, N. Ozay, and R. M. Murray. Reactive controllers for differentially flat systems with temporal logic constraints. In *Proc. of the Conference on Decision and Control*, pages 7664–7670, 2012.

[20] J. Lygeros, C. Tomlin, and S. Sastry. Controllers for reachability specifications for hybrid systems. *Automatica*, 35(3):349–370, 1999.

[21] A. Majumdar and R. Tedrake. Funnel libraries for real-time robust feedback motion planning. *arXiv preprint arXiv:1601.04037*, 2016.

[22] I. B. Makhlouf and S. Kowalewski. Networked cooperative platoon of vehicles for testing methods and verification tools. In *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems*, pages 37–42, 2014.

[23] D. Q. Mayne, M. M. Seron, and S. V. Raković. Robust model predictive control of constrained linear systems with bounded disturbances. *Automatica*, 41(2):219 – 224, 2005.

[24] S. V. Raković, B. Kouvaritakis, M. Cannon, C. Panos, and R. Findeisen. Parameterized tube model predictive control. *IEEE Transactions on Automatic Control*, 57(11):2746–2761, 2012.

[25] S. V. Raković, B. Kouvaritakis, R. Findeisen, and M. Cannon. Homothetic tube model predictive control. *Automatica*, 48(8):1631–1638, 2012.

[26] 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.

[27] B. Schürmann and M. Althoff. Guaranteeing constraints of disturbed nonlinear systems using set-based optimal control in generator space. In *Proc. of the 20th IFAC World Congress*, 2017.

[28] P. Tabuada and G. J. Pappas. Linear time logic control of discrete-time linear systems. *IEEE Transactions on Automatic Control*, 51(12):1862–1877, 2006.

[29] R. Tedrake, I. R. Manchester, M. Tobenkin, and J. W. Roberts. LQR-trees: Feedback motion planning via sums-of-squares verification. *The International Journal of Robotics Research*, 29(8):1038–1052, 2010.