Convex Interpolation Control with Formal Guarantees for Disturbed and Constrained Nonlinear Systems

Bastian Schürmann
Technische Universität München
bastian.schuermann@tum.de

Matthias Althoff
Technische Universität München
althoff@tum.de

ABSTRACT

A new control method for nonlinear systems is presented which solves reach-avoid problems by interpolating optimal solutions using convex combinations. It also provides formal guarantees for constraint satisfaction and safety. Reach-avoid problems are important control tasks, which arise in many modern cyber-physical systems, including autonomous driving and robotic path planning. We obtain our control policy by computing the optimal input trajectories for finitely many extreme states only and combining them using convex combinations for all states in a continuous set. Our approach has very low online computation complexity, making it applicable for fast dynamical systems. Iterating through our approach leads to a new form of feedback control with formal guarantees in the presence of disturbances. We demonstrate the new control method for a control problem in automated driving and show the advantages compared to a classical control method.

Keywords

Interpolation Control, Reach-Avoid Problems, Nonlinear Control, Robust Control, Formal Verification, Convex Combinations

1. INTRODUCTION

Reach-avoid problems are important problems in cyber-physical systems and arise in many different application fields. For instance, autonomous cars should reach a desired position while avoiding other traffic participants and staying on the road. Another example is robot manipulators whose end-effector must reach a desired position without colliding with surrounding persons or objects. The goal is to always find a controller which steers all states from a given initial set into a set around a target state. Until reaching the final set, the system trajectories must avoid unsafe sets, such as obstacles, and satisfy constraints, for example input limitations. The task becomes even harder if the final set must be reached at a given point in time. For a single initial state, this can be achieved by solving a constraint optimization problem. The difficulty arises if one wants to control all states of a given initial set, which is required when the exact initial state is not known before runtime. The traditional approach would lead to infinitely many optimization problems and would therefore be infeasible.

Our goal is to provide a solution to this problem by extending the control inputs for finitely many states to a whole set of infinitely many states, while still ensuring that the final set is reached and that the constraints are satisfied. To do so, we present a novel control approach which utilizes optimal input trajectories for the extreme states of the initial set and uses convex combinations of these inputs to control the current state. By iterating this process over several steps, we obtain a feedback policy which ensures stability and robustness against disturbances, thereby combining the advantages of optimal (open-loop) control and feedback control. This idea provides a new way of viewing control theory: instead of looking for a feedback control law, which offers a stable closed-loop behavior, we directly use finitely many optimal open-loop inputs to control the system. Because of the way we compute the inputs for arbitrary states from finitely many optimal input trajectories, we achieve formal guarantees for the whole set of controlled states. Since the online complexity is low, fast sampling times can be achieved. At the same time, the basic idea of our approach is rather simple, both to understand as well as to implement, which facilitates its applicability in industry.

We apply our technique to maneuver automata, resulting in hybrid dynamics. Maneuver automata [13, 22] are used for online path planning tasks, where it is not possible to solve reach-avoid problems for fast and complex dynamical systems online. Instead, the overall path planning task is split into smaller reach-avoid problems, which can be solved offline (Fig. 1, ①). The solutions, so-called motion primitives [13], are stored as states in a maneuver automaton. Therein, two maneuvers can be safely connected if the reachable set of the previous motion primitive is completely contained in the initial set of the succeeding one (Fig. 1, ②). The maneuver automaton can then be used for online path-planning by simply connecting the pre-computed motion primitives along the transitions of the maneuver automaton and checking whether the possible plans do not enter unsafe regions (Fig. 1, ③-⑥). [22, 14] improve the applicability of maneuver automata for real systems by making them safe in uncertain environments through tools from formal verification (see e.g., [4, 1] and references therein). While this is a big

HSCC'17, April 18 - 20, 2017, Pittsburgh, PA, USA

ISBN 978-1-4503-4590-3/17/04…$15.00

DOI: http://dx.doi.org/10.1145/3049797.3049800

advantage, the authors in [14] faced the challenge of ensuring that the reachable set of a motion primitive ends in the initial set of the next one in order to increase the number of transitions in the automaton. Our new control approach offers a solution to this problem.

Besides the hybrid dynamics of the maneuver automaton, the dynamics of the system without a controller could be hybrid as well. Our new method is based on combining optimal control with reachability analysis. Since approaches which deal with hybrid dynamics exist for both methods (e.g. [20, 4, 1]), we can also extend our approach to the case that the uncontrolled system is hybrid. Due to space limitations, however, we focus in this paper only on continuous dynamics of the uncontrolled system so that we are able to describe this new approach in detail.

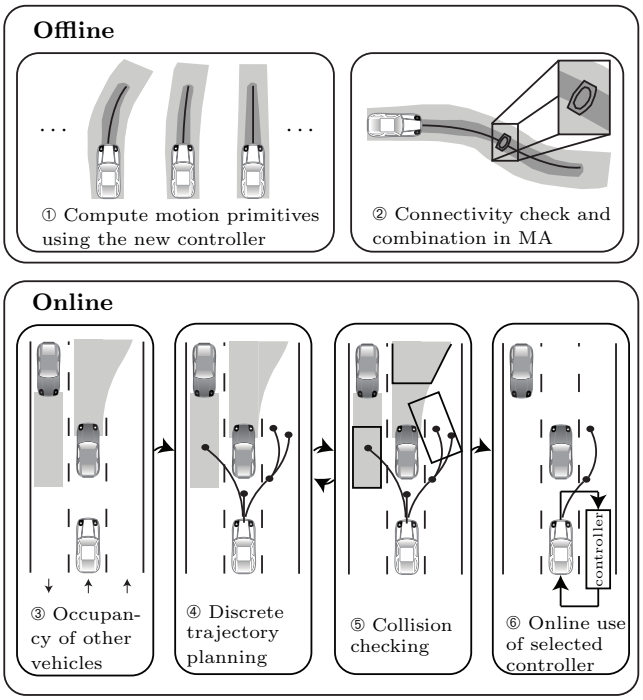


Figure 1: Overview of robust maneuver automata (MA) design for an example in automated driving using our convex control approach.

Related Literature. The idea of interpolating between finitely many offline computed solutions in order to obtain fast, near-optimal online control has been discussed in [31], although without providing guarantees or considering disturbances. Controlling all states around a single trajectory is achieved by exploiting the so-called trajectory robustness in [16]. This method is restricted to feedback-linearizable and differentially flat systems, and it also does not take disturbances into account. One way to obtain optimal control inputs which satisfy input and state constraints is to solve the Hamilton-Jacobi-Bellman (HJB) equation or use dynamic programming [5, 7, 21, 18]. 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 [8, 9] for systematically computing 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) [8, 9]. However, since they have to divide the state space into different regions, this can become easily computationally intractable if the number of dimensions and constraints grows, 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, 17, 34], which are able to take complex specifications into account. Some recent approaches [33, 12] avoid computing complete abstractions of the state space, while still being able to consider complex specifications, by solving reachability problems in a similar way as for maneuver automata. A different approach is used in [30, 22], 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. However, these methods also suffer from the curse of dimensionality, as the complexity of sum-of-squares techniques grows very fast with the dimension. Less affected by the curse of dimensionality is implicit MPC, where tube-based approaches [23, 26] use an additional feedback controller to keep the system along an optimized trajectory despite disturbances. However, 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 [27]. Also, most efficient tube-based MPC approaches are restricted to linear systems, as they depend on the superposition principle.

Organization. The remainder of the paper is organized as follows: We begin with a formal problem statement in Sec. 2. In Sec. 3, we describe the convex control approach. For faster online computation, we discuss closed-form expressions of convex combinations in Sec. 4. Sec. 5 shows a linear approximation of the convex control approach for computational reasons. We demonstrate the applicability of our approach in a numerical example in Sec. 6. In Sec. 7 we discuss the complexity and optimality of our approach, before we conclude with a summary and an outlook in Sec. 8.

2. PROBLEM FORMULATION

In this paper, we consider a disturbed, nonlinear, time-continuous system of the form

$$\dot{x}(t) = f(x(t), u(t), w(t)), \tag{1}$$

with states $x(t) \in \mathbb{R}^n$, inputs $u(t) \in \mathbb{R}^m$, and disturbances $w(t) \in \mathcal{W} \subset \mathbb{R}^d$ ($\mathcal{W}$ is compact, i.e., closed and bounded). We do not require any stochastic properties for $w(\cdot)$; we only assume that any possible disturbance trajectory is bounded at any point in time in the compact set $\mathcal{W}$. We denote this by $w(\cdot) \in \mathcal{W}$, which is a shorthand for $w(t) \in \mathcal{W}, \forall t \in [0, t_f]$, where $t_f \in \mathbb{R}_0^+$ is the final time. The same shorthand is also used for states and inputs throughout the paper. We denote the solution of (1) with initial state $x(0)$, input $u(\cdot)$, and disturbance $w(\cdot)$ at time t as $\xi(x(0), u(\cdot), w(\cdot), t)$. The solution satisfies the following two properties:

1. $\xi(x(0), u(\cdot), w(\cdot), 0) = x(0)$

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

Sometimes, when we consider the undisturbed nominal system, we use $\xi(x(0), u(\cdot), 0, \cdot)$ to denote the solution without disturbances, i.e., $\mathcal{W} = 0$.

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 $\mathcal{S}_f \subset \mathbb{R}^n$ around an end state $x^{(f)}$ after time t_f, despite the disturbance set $\mathcal{W}$. We minimize the size of the final set by solving

$$\min_{u_{control}} \rho(\mathcal{S}_f, x^{(f)}), \tag{2}$$

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.,

$$\xi(x(0), u(\cdot), w(\cdot), \cdot) \in \mathcal{X}, \tag{3}$$

$$u(\cdot) \in \mathcal{U}, \tag{4}$$

where $\mathcal{X}$ and $\mathcal{U}$ are both convex sets in $\mathbb{R}^n$ and $\mathbb{R}^m$, respectively. The distance function $\rho(\mathcal{S}_f, x^{(f)})$ can have different possible forms. One example would be

$$\rho(\mathcal{S}_f, x^{(f)}) = max\{\|x - x^{(f)}\|_2 | x \in \mathcal{S}_f\}.$$

Note that during offline computation, the locations of most non-convex constraints, such as other traffic participants in automated driving or human workers surrounding robots, are not known. Therefore, we use this approach to compute the motion primitives offline in advance, while taking convex input constraints, e.g., maximum acceleration or steering, and convex state constraints, e.g., maximum velocity, into account. The non-convex dynamical constraints are handled during the online planning using the maneuver automaton as described in the introduction, see Fig. 1.

For the majority of this paper, we want to find a final set $\mathcal{S}_f$ which is as small as possible. If the task is instead to steer all states into a given final set, then we would have to adapt the algorithms by adding this as an additional constraint. In this case, however, it might be possible that no solution can be found, depending on the choice of constraints, final time, and final set.

3. CONVEX CONTROL

In order to solve the previously-stated problem, we propose a convex control approach: The basic idea is to use a convex combination of control inputs computed for the extreme states of an initial set $\mathcal{S}_{init}$ in order to ensure that all trajectories starting in this initial set will end in a desired set $\mathcal{S}_f$. A very similar idea has been used in parametrized tube MPC [26], where control inputs are also computed as convex combinations of extreme input values. This approach, however, is only used for linear systems and not for disturbed, nonlinear systems, as considered in this paper. Computing convex combinations of extreme inputs has also been used for reachability analysis of linear systems, see e.g. [11]. The focus in this case is on the reachable sets rather than optimal control inputs, however.

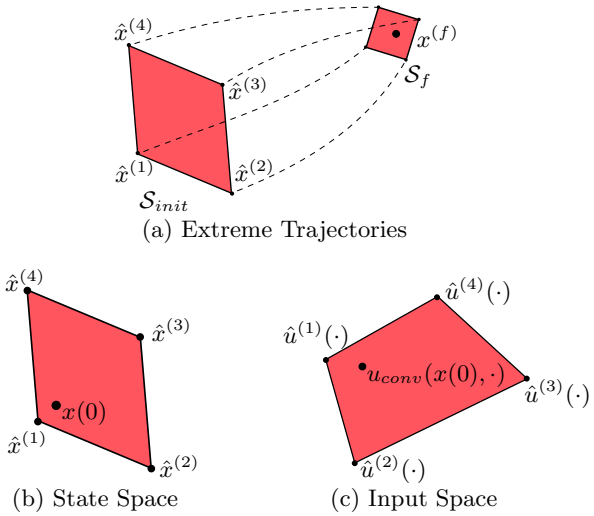
For our convex control approach, we consider the initial set $\mathcal{S}_{init}$ to be given by a polytope P, defined by its p extreme states $\hat{x}^{(i)}, i = 1, \dots, p$. We assume to know for each of the extreme states $\hat{x}^{(i)}$ a piecewise-continuous control input $\hat{u}^{(i)}(\cdot), i = 1, \dots, p$, which steers this state into the desired set $\mathcal{S}_f$. To obtain the control input which steers an arbitrary state $x(0) \in P$ into the desired set $\mathcal{S}_f$, we express $x(0)$ as a convex combination of the extreme states $\hat{x}^{(i)}$ by choosing $\lambda_i(x(0))$ such that

$$x(0) = \sum_{i=1}^{p} \lambda_i(x(0)) \hat{x}^{(i)}, \tag{5}$$

with $\lambda_i(x(0)) \geq 0, \sum_{i=1}^{p} \lambda_i(x(0)) = 1$. We then use the same parameters $\lambda_i(x(0))$ to compute the corresponding control input $u_{conv}(x(0), \cdot)$ for the state $x(0)$ as a convex combination of the control inputs $\hat{u}^{(i)}(\cdot)$ of the extreme states $\hat{x}^{(i)}$, i.e.,

$$u_{conv}(x(0), \cdot) = \sum_{i=1}^{p} \lambda_i(x(0)) \hat{u}^{(i)}(\cdot). \tag{6}$$

This is illustrated in Fig. 2. Note that the controller $u_{conv}(x(0), \cdot)$ provides an open loop control input. Feedback is realized by iteratively applying (6). As we see in Sec. 4, it is even possible to obtain closed-form expressions of the convex combinations, such that the system of inequalities in (5) does not to have to be solved online.



(a) Extreme Trajectories

(b) State Space (c) Input Space

Figure 2: Basic idea of the convex control approach: Compute input trajectories which control the extreme states $\hat{x}^{(i)}$ of the initial set $\mathcal{S}_{init}$ close to the desired final state $x^{(f)}$ (a). Express state $x(0) \in \mathcal{S}_{init}$ as a convex combination of extreme states $\hat{x}^{(i)}$ (b). Use the same convex combination to compute the corresponding control input $u_{conv}(x(0), \cdot)$ using the control inputs $\hat{u}^{(i)}(\cdot)$ of the extreme states (c).

3.1 Convex Control for Linear Systems

Before we consider disturbed, nonlinear systems, we illustrate first how our approach works for undisturbed, linear systems of the form

$$\dot{x}(t) = Ax(t) + Bu(t), \tag{7}$$

with $x(t) \in \mathbb{R}^n, u(t) \in \mathbb{R}^m, A \in \mathbb{R}^{n \times n}$, and $B \in \mathbb{R}^{n \times m}$. In order to solve the control problem, we perform the following two steps:

Step 1: For each of the extreme states $\hat{x}^{(i)}, i = 1, \ldots, p$, of the initial set, we solve a constrained optimization problem of the form

$$\forall \hat{x}^{(i)} : \min_{\hat{u}^{(i)}(\cdot)} J_{linear}(\|\xi(\hat{x}^{(i)}, \hat{u}^{(i)}(\cdot), 0, t_f) - x^{(f)}\|), \quad \text{w.r.t. (3), (4),}$$

to find an input sequence which returns a solution that ends as close as possible to $x^{(f)}$ while satisfying the state and input constraints (3)-(4). The norm and the exact form of the cost function can be freely chosen depending on the specific problem.

Step 2: For a given state in the initial set: express it as a convex combination of the extreme states by solving (5) (or using a closed-form expression from Sec. 4) and use the same convex combination of the corresponding input sequences (6) to control it to the final set.

By applying these two steps, we obtain for each extreme state $\hat{x}^{(i)}$ an input sequence $\hat{u}^{(i)}$, such that the corresponding state trajectory ends close (see (2)) to the desired final state $x^{(f)}$ after a fixed time t_f. If all input sequences $\hat{u}^{(i)}(\cdot)$ and corresponding state trajectories $\xi(\hat{x}^{(i)}, \hat{u}^{(i)}(\cdot), 0, \cdot)$ satisfy the input and state constraints, respectively, then, all trajectories starting in the initial set $\mathcal{S}_{init}$ under the convex control law (6) end in the a priori known compact set

$$\mathcal{S}_f = \mathbf{conv}(\xi(\hat{x}^{(1)}, \hat{u}^{(1)}(\cdot), 0, t_f), \ldots, \xi(\hat{x}^{(p)}, \hat{u}^{(p)}(\cdot), 0, t_f)),$$

where $\mathbf{conv}(\cdot)$ denotes the convex hull. Moreover, all trajectories satisfy the state constraints (3) and input constraints (4) at all times.

This directly results from the way we compute the control law and from the convexity of linear systems. Using (5) and (6) it follows from the superposition principle that $\forall t \in [0, t_f]$:

$$\begin{aligned}
&\xi(x(0), u_{conv}(x(0), \cdot), 0, t) \\
&= \xi\left(\sum_{i=1}^{p} \lambda_i(x(0))\hat{x}^{(i)}, \sum_{i=1}^{p} \lambda_i(x(0))\hat{u}^{(i)}(\cdot), 0, t\right) \\
&= \sum_{i=1}^{p} \lambda_i(x(0))\xi(\hat{x}^{(i)}, \hat{u}^{(i)}(\cdot), 0, t),
\end{aligned}$$

i.e., any trajectory starting in the initial set lies inside the convex set of the extreme trajectories. Since the extreme trajectories satisfy the convex state constraints, any inner trajectory satisfies the state constraints as well. The control inputs are convex combinations of the extreme inputs, which are contained in the convex input constraint $\mathcal{U}$. Therefore, it follows from convexity that the control inputs are in the set $\mathcal{U}$ as well.

3.2 Convex Control for Nonlinear Systems with Disturbances

Before we extend this idea to nonlinear systems with disturbances (1), let us first define reachable sets and zonotopes, as they are important for the rest of the paper.

Definition 1 (Reachable Set). *For a 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 end 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 time points, i.e.,*

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

If we consider the reachable set for a system with feedback $u_{fb}(x(t))$, then we denote by $\mathcal{R}_{t,u_{fb},\mathcal{W}}(\mathcal{S})$ the reachable set obtained if we consider the closed-loop dynamics $\dot{x}(t) = f(x(t), u_{fb}(x(t)), w(t))$ and no open-loop inputs.

All methods used in the reachablility analysis are over-approximative, i.e., the real reachable set is guaranteed to lie inside the computed reachable set. We have to compute over-approximations, as it is impossible to compute exact reachable sets for most systems [24]. Since we guarantee the constraint satisfaction for the over-approximated reachable set, the real reachable set satisfies the constraints as well, i.e., our algorithm is sound. Because we cannot compute the exact reachable set, we are also unable to provide error bounds for the approximations. Simulations indicate however, that over-approximations can be tightly computed [1]. In this work, we apply the reachability algorithms from [1], which use zonotopes as a set representation:

Definition 2 (Zonotope). *A set is called a zonotope if it can be written as*

$$\mathcal{Z} = \{x \in \mathbb{R}^n | x = c_\mathcal{Z} + G_\mathcal{Z}\alpha, \alpha_i \in [-1, 1]\},$$

with $\alpha \in \mathbb{R}^q$ *and* α_i *denoting the* $i-th$ *entry of the vector* α*. Therein* $c_\mathcal{Z} \in \mathbb{R}^n$ *defines the center of the zonotope, and the matrix* $G_\mathcal{Z} \in \mathbb{R}^{n \times q}$ *contains the* $q = o\,n$ *generators as its columns, with* o *denoting the order of the zonotope. A zonotope with* n *linearly independent generators is called a parallelotope.*

Since nonlinear dynamics in general do not preserve convexity [32], convex control cannot be applied in the same way as in the linear case. Instead, we divide the control problem into intermediate steps and iteratively apply the convex control law to steer the system along a reference trajectory. For each time step, we compute the reachable set using techniques from formal verification [1], thereby ensuring that the constraints are always satisfied despite disturbances and nonlinear dynamics. By recomputing the convex control inputs in each time step, we realize feedback and counteract the effects from disturbances. Through the use of reachability analysis, we have a separation of concerns: the optimization provides performance, while the reachability analysis provides guarantees for the satisfaction of all constraints. Therefore, the guarantees still hold if we cannot find optimal solutions for the optimal control problems as long as we find solutions which satisfy the constraints. The new control approach is presented in Alg. 1 and is illustrated in Fig. 3. It consists of three major steps:

Step 1: We first solve a constrained, nonlinear optimization problem in *center_optimal_control* (Alg. 1, line 1) for the nominal system in order to find a trajectory $x^{(c)}(\cdot)$ which starts in $x^{(c)}(0)$, the center of $\mathcal{S}_{init}$, and ends as close as possible to $x^{(f)}$. To solve this control problem efficiently, piecewise-continuous control inputs are computed

Algorithm 1 Offline Part of the Convex Control Algorithm for Nonlinear Systems with Disturbances

1: $(x^{(c)}(\cdot), u^{(c)}(\cdot)) \leftarrow$ *center_optimal_control*$(x^{(c)}(0), x^{(f)}, t_f, N, \bar{\mathcal{X}}, \mathcal{U})$
2: Initialize: $\mathcal{S}_{reach,1} \leftarrow \mathcal{S}_{init}$
3: **for** $k = 1, \dots, N$ **do**
4: $\quad \bar{\mathcal{P}}_k \leftarrow$ *compute_parallelotope_approx*$(\mathcal{S}_{reach,k})$
5: $\quad (\hat{x}^{(1,k)}, \dots, \hat{x}^{(2^n,k)}) \leftarrow$ *comp_extreme_pts*$(\bar{\mathcal{P}}_k)$
6: $\quad$ **for** $i = 1, \dots, 2^n$ **do**
7: $\quad\quad \hat{u}^{(i,k)}(\cdot) \leftarrow$ *optimal_corner_control*$(\hat{x}^{(i,k)}, x^{(c)}(t_{k+1}), \bar{\mathcal{X}}, \mathcal{U})$
8: $\quad$ **end for**
9: $\quad \mathcal{S}_{reach,k+1} \leftarrow$ *comp_reach_set*$(\mathcal{S}_{reach,k}, \hat{u}^{(1,k)}(\cdot), \dots, \hat{u}^{(2^n,k)}(\cdot), \mathcal{W})$
10: **end for**

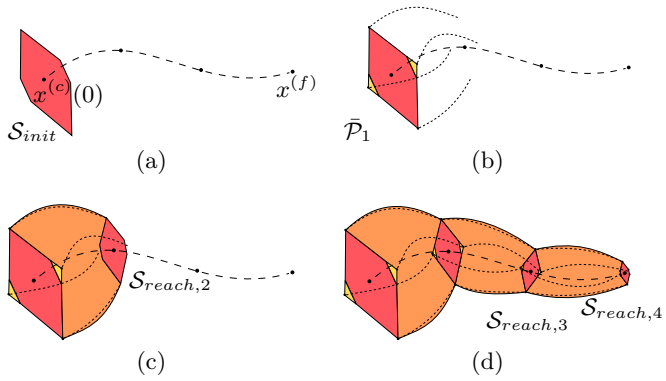


Figure 3: Convex control for nonlinear systems: Compute a reference trajectory $x^{(c)}(\cdot)$ from the center $x^{(c)}(0)$ of the initial set $\mathcal{S}_{init}$ to the final state $x^{(f)}$ (a). Over-approximate the initial set by a parallelotope $\bar{\mathcal{P}}_1$, and compute optimal trajectories for the extreme states (b). Compute the reachable set $\mathcal{S}_{reach,2}$ for one time step under the convex control law while taking all possible disturbance effects into account (c). Repeat the procedure (d).

for N time steps of length $\Delta t = \frac{t_f}{N}$. For an easier notation, we introduce $t_i = i\Delta t$.

Since we solve the optimization problem for the undisturbed system, we have to tighten the nominal state constraints such that the disturbed system still satisfies the original state constraints. We therefore define tightened state constraints as

$$\xi(x(0), u(\cdot), 0, \cdot) \in \bar{\mathcal{X}} \subseteq \mathcal{X}. \tag{8}$$

We discuss how to compute them at the end of this section.

The cost function of the optimization problem is chosen as

$$\min_{u^{(c)}(\cdot)} J_{center}(\|\xi(x^{(c)}(0), u^{(c)}(\cdot), 0, t_f) - x^{(f)}\|, \|u^{(c)}(\cdot)\|),$$
$$\text{w.r.t. } (8), (4).$$

Therein, $\|u^{(c)}(\cdot)\|$ denotes some norm on the input trajectory. We aim for a center trajectory which ends close to the desired end state and whose control inputs are not too large, such that we have some input capacities left for the inputs of the extreme states, which are explained in the next step. The resulting center trajectory is denoted by

$$x^{(c)}(\cdot) = \xi(x^{(c)}(0), u^{(c)}(\cdot), 0, \cdot).$$

Step 2: While step 1 is performed only once, we perform steps 2 and 3 for each time step k, $k = 1, \dots, N$ (see Alg. 1). At time step k, we over-approximate the reachable set $\mathcal{S}_{reach,k}$ of the previous time step by a parallelotope $\bar{\mathcal{P}}_k$ in *compute_parallelotope_approx* (Alg. 1, line 4), i.e., $\mathcal{S}_{reach,k} \subseteq \bar{\mathcal{P}}_k$. At the first time step $k = 1$, we over-approximate the initial set $\mathcal{S}_{init}$ (line 2). There exist efficient algorithms to formally compute parallelotope over-approximations for zonotopes, see e.g., [1] and the references therein. We use parallelotopes, since they offer a good combination of a small number of extreme states and enclosed volume, and since analytical, closed-form expressions for convex combinations in parallelotopes exist (see Sec. 4).

For the parallelotope $\bar{\mathcal{P}}_k$, we compute the 2^n extreme states $\hat{x}^{(1,k)}, \dots, \hat{x}^{(2^n,k)}$ in *comp_extreme_pts* (line 5). Computing the extreme states of a parallelotope $\mathcal{P}$ with center $c_{\mathcal{P}}$ and generator matrix $G_{\mathcal{P}}$ can be done in a numerically stable way by adding all 2^n combinations of the generators, i.e.,

$$\{\hat{x}^{(1)}, \dots, \hat{x}^{(2^n)}\} = c_{\mathcal{P}} \pm g_{\mathcal{P}}^{(1)} \pm \dots \pm g_{\mathcal{P}}^{(n)}, \tag{9}$$

where $g_{\mathcal{P}}^{(i)}$ denotes the $i-$th column of $G_{\mathcal{P}}$. This is an advantage over general polytopes in half-space representation, for which it is much harder and numerically less reliable to compute the extreme states. We use the extreme states for the convex control strategy by solving a nonlinear optimal control problem with the nominal system dynamics for one time step. This is done in *optimal_corner_control* (lines 6-8) by solving for each extreme state $\hat{x}^{(i,k)}$:

$$\min_{\hat{u}^{(i,k)}(\cdot)} J_{corner}(\|\xi(\hat{x}^{(i,k)}, \hat{u}^{(i,k)}(\cdot), 0, \Delta t) - x^{(c)}(t_{k+1})\|),$$
$$\text{w.r.t. } (8), (4). \tag{10}$$

This steers all extreme states as close as possible to the optimal center trajectory.

Step 3: In the third step, we use the control inputs of the extreme states of the parallelotope $\bar{\mathcal{P}}_k$ to obtain the convex control law in (6) for each state inside the reachable set of the last step $\mathcal{S}_{reach,k}$. After computing the convex control law $u_{conv}(x, \cdot)$, we use formal reachability analysis tools [1] in *comp_reach_set* (line 9) to obtain an over-approximation of the reachable sets $\bar{\mathcal{R}}_{\Delta t, u_{conv}, \mathcal{W}}(\mathcal{S}_{reach,k}) \supseteq \mathcal{R}_{\Delta t, u_{conv}, \mathcal{W}}(\mathcal{S}_{reach,k})$ and $\bar{\mathcal{R}}_{[0,\Delta t], u_{conv}, \mathcal{W}}(\mathcal{S}_{reach,k}) \supseteq \mathcal{R}_{[0,\Delta t], u_{conv}, \mathcal{W}}(\mathcal{S}_{reach,k})$. We require the reachable sets for time intervals to ensure the satisfaction of state constraints for all points in time. In addition, we use the reachable sets at times t_k to compute the parallelotope over-approximations and to initialize the reachability analysis for the next time step. Therefore, we compute reachable sets for time intervals and time points.

The over-approximation of the reachable set is the initial set for the next time step $k + 1$, i.e.,

$$\mathcal{S}_{reach,k+1} = \bar{\mathcal{R}}_{\Delta t, u_{conv}, \mathcal{W}}(\mathcal{S}_{reach,k}),$$

and we use this to continue with Step 2. By iterating Steps 2 and 3 for all N time steps, we obtain with $\mathcal{S}_f = \mathcal{S}_{reach,N+1}$

the over-approximation of the final reachable set of all states starting in the initial set $\mathcal{S}_{init}$ despite disturbances.

After the controller is computed offline using Alg. 1, we save the computed extreme states $\hat{x}^{(i,k)}$ together with the corresponding input trajectories $\hat{u}^{(i,k)}(\cdot)$ in a look-up table. In each time step during the online application of the convex controller, the current state $x(t_k)$ is expressed as a convex combination of the corresponding extreme states $\hat{x}^{(i,k)}$. The control input $u_{conv}(x(t_k),\cdot)$ is obtained from the convex combination of the extreme inputs $\hat{u}^{(i,k)}(\cdot)$ using the same parameters $\lambda_{i,k}(x(t_k))$ as described at the beginning of this section in (5) and (6).

The results of the convex control approach for nonlinear systems are summarized in the following theorem, which is the main result of this paper:

THEOREM 1. *We consider a nonlinear system with disturbances* (1) *and with state and input constraints* (3)-(4). *We assume that we have found a convex controller for this system using Algorithm 1. If*

$$\bar{\mathcal{R}}_{[0,t_f],u_{conv},\mathcal{W}}(\mathcal{S}_{init}) \subseteq \mathcal{X}, \tag{11}$$

then any trajectory $\xi(x(0),u_{conv}(x(0),\cdot),w(\cdot),\cdot)$ *which starts in the initial set will end after time* t_f *in* $\mathcal{S}_f = \mathcal{S}_{reach,N+1}$, *i.e.,* $\xi(x(0),u_{conv}(x(0),\cdot),w(\cdot),t_f) \in \mathcal{S}_f, \forall x(0) \in \mathcal{S}_{init}, \forall w(\cdot) \in \mathcal{W}$. *Moreover, every trajectory satisfies the state constraints and the applied inputs satisfy the input constraints despite the presence of disturbances, i.e.,* $\forall x(0) \in \mathcal{S}_{init}, \forall w(\cdot) \in \mathcal{W}, \forall t \in [0,t_f]$:

$$\xi(x(0),u_{conv}(x(0),\cdot),w(\cdot),t) \in \mathcal{X} \wedge u_{conv}(x(0),t) \in \mathcal{U}.$$

PROOF. From the definition of the reachable set and the way we compute $\mathcal{S}_f$ as the over-approximation of the final reachable set, it follows that $\forall x(0) \in \mathcal{S}_{init}, \forall w(\cdot) \in \mathcal{W}$:

$$\begin{aligned}\xi(x(0),u_{conv}(x(0),\cdot),w(\cdot),t_f) &\in \mathcal{R}_{t_f,u_{conv},\mathcal{W}}(\mathcal{S}_{init}) \\ &\subseteq \bar{\mathcal{R}}_{t_f,u_{conv},\mathcal{W}}(\mathcal{S}_{init}) = \mathcal{S}_f.\end{aligned}$$

In the same way, it follows from assumption (11) that

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

and therefore, the state constraint is satisfied for any trajectory starting in the initial set.

When computing the control inputs $\hat{u}^{(i,k)}(\cdot)$ of the extreme states $\hat{x}^{(i,k)}$ in (10), we restrict them to lie in the input set $\mathcal{U}$ at all times. Since the input set is convex, and since any convex combination of points in a convex set lies again in the convex set [10], it follows that any convex combination of these inputs, and therefore any control input from our convex controller, satisfies the input constraints. □

Tightened State Constraints. In general it is hard to know in advance how much tighter the new state constraints have to be, and there exists no general solution for this problem in literature for disturbed, nonlinear systems. Since we combine controller synthesis with reachable set computation, we can check offline if the controller satisfies all constraints, and if not, adapt the tightened state constraints. Since we know which state constraints have been violated, we can adapt exactly these constraints. By iteratively tightening the nominal state constraints offline in advance until all constraints are satisfied by the real system, we obtain a formally correct controller for the online application. The reachability analysis in [1] relies on linearizing the dynamics and over-approximating the linearization errors and disturbance. Therefore, we are able to use these over-approximations to obtain good initial estimates of how much we have to tighten the state constraints.

4. CLOSED-FORM EXPRESSION OF CONVEX COMBINATIONS

When applying the proposed convex control law (6), at each new time step, we have to find the parameters $\lambda_i(x)$ to express a state x as a convex combination of the extreme states $\hat{x}^{(i)}$. Utilizing solvers for this problem is computationally expensive, especially for higher dimensional systems, and they would only provide an implicit solution. The computation time would restrict the sampling times of our controller, and the implicit solutions would prohibit the application of reachability analysis, which relies on an explicit, closed-form expression of the closed-loop dynamics.

To overcome these problems, closed-form expressions of convex combinations of simplices, parallelotopes, and general polytopes are presented in [28]. As mentioned before, parallelotopes offer a good combination of enclosed volume and a small number of extreme states. While simplices for example have the advantage that they have only $n+1$ extreme states, the enclosed volume is smaller and has an "impractical" shape for our application purposes. On the other hand, objects like higher-order zonotopes or general polytopes may better describe certain shapes, but when applied, the number of vertices increases significantly. Therefore we use parallelotopes to over-approximate the reachable set for the convex control computation in Step 2 in Sec. 3.2. We use these parallelotope over-approximations only to obtain the input combination and use the actual high-order zonotope for reachability analysis. In doing so, we avoid the error due to this over-approximation such that it has no significant impact on the reachability analysis.

The following theorem shows how to obtain closed-form expressions of convex combinations for parallelotopes:

THEOREM 2 ([28]). *We consider a parallelotope* $\mathcal{P} \subset \mathbb{R}^n$ *given by*

$$\mathcal{P} = \{x \in \mathbb{R}^n | x = c_{\mathcal{P}} + G_{\mathcal{P}}\alpha(x), \alpha_i(x) \in [-1,1]\},$$

with 2^n *extreme states* $\hat{x}^{(1)},\dots,\hat{x}^{(2^n)}$, *see* (9). *Given a state* $x \in \mathcal{P}$, *this state can be expressed as a convex combination of the extreme states as* $x = \sum_{i=1}^{2^n} \lambda_i(x)\hat{x}^{(i)}$, *where the parameters* $\lambda_i(x)$ *are given by the following closed-form expression*

$$\lambda_i(x) = \prod_{j=1}^{n} \mu_{i,j}, \tag{12}$$

where

$$\mu_{i,j} = \begin{cases} x'_j & if\ \alpha_j(\hat{x}^{(i)}) = 1 \\ 1 - x'_j & if\ \alpha_j(\hat{x}^{(i)}) = -1. \end{cases} \tag{13}$$

Thereby, x' *is the transformed state of* x *under the affine transformation*

$$x' = \frac{1}{2}G_{\mathcal{P}}^{-1}(x - c_{\mathcal{P}}) + \frac{1}{2}\mathbf{1} \tag{14}$$

and x'_j *denotes its* $j-th$ *entry.*

The proof as well as results for sets other than parallelotopes can be found in [28]. Note that for a point $\hat{x}^{(i)} \in \mathcal{P}$ to be an extreme point, the entries in the corresponding parameter vector $\alpha(\hat{x}^{(i)})$ must all be ± 1; therefore, one of the cases in (13) is always satisfied. Also, $G_{\mathcal{P}}^{-1}$ always exists since $G_{\mathcal{P}}$ has full rank.

With this theorem, we are able to pre-compute the convex combinations for the parallelotopes at different points in time. Since we know the parallelotopes in advance, we can compute all matrices and matrix inverses of Thm. 2 offline. During the online computation, we simply plug in the current state in (14) and use the result to compute the $\lambda_i(x)$. The computation of all $\lambda_i(x)$ for a ten-dimensional parallelotope can be performed in around 0.1ms which is over 200 times faster than using linear programming solvers [28].

5. LINEAR APPROXIMATION OF THE CONVEX CONTROL APPROACH

The convex controller described in Sec. 3 is a nonlinear controller as can be seen by looking at the closed-form expressions of $\lambda_i(x)$ in (12), where the different entries x_j' are multiplied with each other. Although a convex combination is a linear combination of the extreme states, the parameters $\lambda_i(x)$ have a nonlinear dependency on the initial state for systems with dimensions greater than one. This increases the nonlinearity of the whole closed-loop system, which leads to larger computation errors during the reachability computation, as well as to a higher computational complexity of the reachability computation itself [1].

In order to overcome these problems, we present an alternative approach where we use a linear approximation of the convex control approach. To do so, we take advantage of the fact that efficient techniques for reachability analysis use zonotopes as set representation [3, 1]. We use the zonotope representation of the state set to compute a corresponding zonotope representation for the inputs. We modify only the third step in Sec. 3.2 by adding the input approximation, as shown in Alg. 2. For simpler computations, we restrict our considerations to piecewise-constant inputs, and we use the constant value u to also denote the constant input trajectory $u(\cdot)$, with $u(t) = u, \forall t$.

Algorithm 2 Convex Control Algorithm for Nonlinear Systems with Disturbances Using Linear Input Combinations

1: ... (Lines 1 to 8 as in Alg. 1)
2: $\mathcal{Z}_{U,k} \leftarrow$ *opt_input_zonotope*$(\bar{\mathcal{P}}_k, \hat{u}^{(1,k)}, \dots, \hat{u}^{(2^n,k)})$
3: $\mathcal{S}_{reach,k+1} \leftarrow$ *compute_reachable_set*$(\mathcal{S}_{reach,k}, \mathcal{Z}_{U,k}, \mathcal{W})$
4: **end for** (from last line of Alg. 1)

Any state x in a parallelotope $\mathcal{P}$ is uniquely defined by the parameters $\alpha(x)$, i.e., $x = c_{\mathcal{P}} + G_{\mathcal{P}}\alpha(x)$, and therefore $\alpha(x)$ can be obtained by

$$\alpha(x) = G_{\mathcal{P}}^{-1}(x - c_{\mathcal{P}}), \tag{15}$$

where $G_{\mathcal{P}}^{-1}$ exists since $\mathcal{P}$ is a parallelotope.

In every time step $k = 1, \dots, N$, we want to find an input zonotope $\mathcal{Z}_{U,k}$ with center $c_{\mathcal{Z}_{U,k}}$ and generator matrix $G_{\mathcal{Z}_{U,k}}$ such that we obtain the corresponding control input for any state $x(t_k) \in \bar{\mathcal{P}}_k$ by just using $\alpha(x(t_k))$ in

$$u_{zono}(x(t_k)) = c_{\mathcal{Z}_{U,k}} + G_{\mathcal{Z}_{U,k}}\alpha(x(t_k)). \tag{16}$$

By plugging (15) into (16), we obtain

$$u_{zono}(x(t_k)) = c_{\mathcal{Z}_{U,k}} + G_{\mathcal{Z}_{U,k}} G_{\bar{\mathcal{P}}_k}^{-1}(x(t_k) - c_{\bar{\mathcal{P}}_k}), \tag{17}$$

which is linear in $x(t_k)$ as desired.

We now have to choose the center $c_{\mathcal{Z}_{U,k}}$ and generator matrix $G_{\mathcal{Z}_{U,k}}$ of $\mathcal{Z}_{U,k}$, which best match the desired inputs. Clearly, as this is a linear approximation of the nonlinear convex input combinations, we cannot match the input for every state in $\bar{\mathcal{P}}_k$ exactly. We choose $c_{\mathcal{Z}_{U,k}}$ and $G_{\mathcal{Z}_{U,k}}$ in *opt_input_zonotope* (Alg. 2, line 2) by solving an optimization problem such that the difference between the optimal inputs for the extreme states $\hat{u}^{(i,k)}$ and the inputs from the input zonotope $\mathcal{Z}_{U,k}$ for the corresponding $\alpha(\hat{x}^{(i)})$ is minimized:

$$\min_{c_{\mathcal{Z}_{U,k}}, G_{\mathcal{Z}_{U,k}}} \sum_{i=1}^{2^n} \left\| \hat{u}^{(i,k)} - (c_{\mathcal{Z}_{U,k}} + G_{\mathcal{Z}_{U,k}}\alpha(\hat{x}^{(i,k)})) \right\|$$
$$\text{w.r.t. } c_{\mathcal{Z}_{U,k}} + G_{\mathcal{Z}_{U,k}}\alpha(\hat{x}^{(i,k)}) \in \mathcal{U}, \forall i \in \{1, \dots, 2^n\}. \tag{18}$$

Now, we have the desired linear control law $u_{zono}(\cdot)$ for any state in $\bar{\mathcal{P}}_k$ with (17). Using Alg. 2 and therefore replacing u_{conv} by u_{zono} in Thm. 1 ensures that the results of Thm. 1 also hold in the case of the new control law. We do not change anything about the reachability computation, with the exception of considering the new control law u_{zono}, in *compute_reachable_set* (line 3). Therefore, our algorithm is still sound and we have guarantees for satisfaction of the state constraints. Because of (18) and since $\mathcal{U}$ is convex it holds that $\forall k$

$$\mathcal{Z}_{U,k} = \mathbf{conv}\left(u_{zono}(\hat{x}^{(1,k)}), \dots, u_{zono}(\hat{x}^{(2^n,k)})\right) \subseteq \mathcal{U},$$

and therefore the input constraints are satisfied as well.

6. NUMERICAL EXAMPLE

In this section, we provide a numerical example to show the applicability of the proposed control approach for a constrained, nonlinear system. We choose a kinematic model of a vehicle, which is broadly used to model the most important dynamics of a car [25]:

$$\dot{v} = a + w_1,\ \dot{\Psi} = b + w_2,\ \dot{x} = v\,cos(\Psi),\ \dot{y} = v\,sin(\Psi), \tag{19}$$

where the states v, Ψ, x, and y are the velocity, the orientation, and the positions in x and in y directions, respectively. The acceleration a and the normalized steering angle b are the inputs, and w_1 and w_2 are additive disturbances. They are constrained to lie in the intervals $a \in [-9.81, 9.81]\frac{m}{s^2}$, $b \in [-0.4, 0.4]\frac{rad}{s}$, $w_1 \in [-0.5, 0.5]\frac{m}{s^2}$, and $w_2 \in [-0.02, 0.02]\frac{rad}{s}$.

We use our convex control approach to compute a maneuver automaton for this model. Due to space limitations, we present the maneuver automaton only for the three discrete states "drive straight", "turn left", and "turn right", at velocities around $20\frac{m}{s}$. However, we can add additional maneuvers simply by repeating the procedure for other final states. Following the techniques introduced in [14], we can only connect two motion primitives if the reachable set of the first one lies completely in the initial set of the second one, see Fig. 1, ②. Since the car dynamics are independent of the absolute position and orientation, it suffices if we end in a set with the same size as the initial set, but which can be shifted in position and orientation. However, in the velocity dimension, we have to end in the initial set again. Therefore,

it must hold that $\mathcal{S}_{reach,N+1} - \left[0, \Psi^{(f)}, x^{(f)}, y^{(f)}\right]^T \subseteq \mathcal{S}_{init}$. We choose for all maneuvers the initial set as the box $[19.8, 20.2]\frac{m}{s} \times [-0.02, 0.02] rad \times [-0.2, 0.2] m \times [-0.2, 0.2] m$. We apply our convex control approach three times for different final states, corresponding to the three maneuvers. The final states are given for the "drive straight" maneuver by $\left[20\,\frac{m}{s}, 0\,rad, 20\,m, 0\,m\right]^T$ and for the "turn left" and "turn right" maneuvers by $\left[20\,\frac{m}{s}, \pm 0.2\,rad, 19.87\,m, \pm 1.99\,m\right]^T$. Each final state must be reached in one second. We divide the main trajectory into 10 sections and apply four different piece-wise constant control values for each section of the reference trajectory. For the local controllers, we choose the cost function as

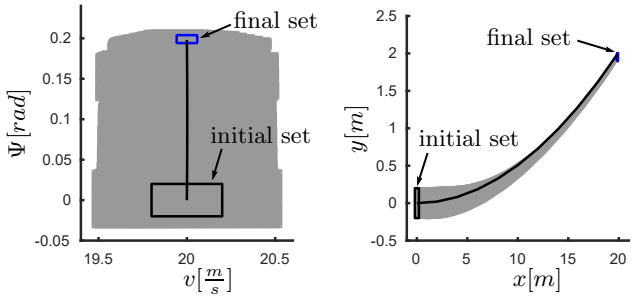
$$J_{corner} = (\hat{x}^{(i,k+1)} - x^{(c)}(t_{k+1}))^T Q (\hat{x}^{(i,k+1)} - x^{(c)}(t_{k+1})),$$

with $\hat{x}^{(i,k+1)} = \xi(\hat{x}^{(i,k)}, \hat{u}^{(i,k)}(\cdot), 0, \Delta t)$ and Q being a diagonal matrix with $[2, 5, 1, 1]$ on the diagonal.

6.1 Computational Results

We implement our approach in MATLAB and use the ACADO toolbox [15] to solve the optimal control problems with a multiple shooting algorithm. For the reachability computation, we use the CORA toolbox [2], where the disturbances are handled as an uncontrollable input. Since the CORA toolbox works with zonotopes, we use the techniques presented in Sec. 5 to obtain a parallelotope approximating the inputs.

The computation of each maneuver takes around 10 seconds, which can be performed offline. The computations are performed on a computer with a 3.1 GHz dual-core i7 processor and 16 GB memory and without using parallel computing. The online computation of the input values can be performed in around 0.01ms, making it applicable to fast systems.



(a) Turn Left, (v, Ψ) plane (b) Turn Left, (x, y) plane

Figure 4: Reachable sets for the "turn left" maneuver with the convex controller. The initial set is plotted in black, the final set in blue, and the reachable set for all times between in gray. The black line shows the center trajectory $x^{(c)}(\cdot)$.

The whole reachable set is shown for the "turn left" maneuver in Fig. 4. In Fig. 5, the initial sets (black) and (shifted) final sets for all motion primitives are plotted. For the convex controller, all final sets (blue) lie around the final states and are completely contained in the desired set, i.e., the shifted initial set. Therefore, we are able to connect all maneuvers with each other and obtain a fully connected

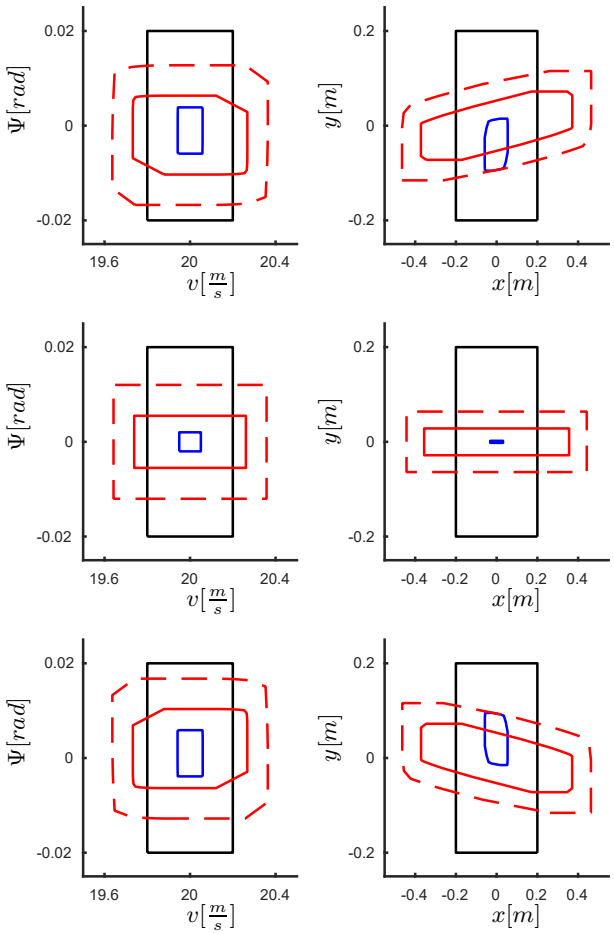


Figure 5: Initial (black) and shifted final sets (blue) for the convex controller, projected to the (v, Ψ) and the (x, y) planes, for the "turn left" (top), "drive straight" (center), and "turn right" (bottom) maneuvers. For comparison the final sets of two LQR controllers (red).

maneuver automaton. The controller satisfies the input constraints at all times due to the way we computed it.

6.2 Comparison with LQR Controller

For comparison, we have also implemented an LQR-tracking controller (red, solid in Fig. 5). It uses the same center-trajectory as a reference trajectory. In each step, we linearize the system around the corresponding state on the center trajectory and compute an LQR controller. We use the same Q matrix as for the corner trajectories. To weight the inputs we choose the R matrix for the Ricatti equation to be the identity matrix. As we see in Fig. 5, the shifted reachable sets exceed the initial sets, making it impossible to combine the maneuver with other maneuvers. Moreover, since the LQR controller does not take input constraints into account, it uses inputs for b in (19) up to $0.54\frac{rad}{s}$, which is more than the maximally allowed value. By decreasing the weights of the inputs, we can move the final sets inside the initial sets; however, the input constraints violation increases even more in this case. If we increase the weights

on the inputs to $R = 4I$, we obtain a maximal $b = 0.40\frac{rad}{s}$, which barely satisfies the constraint. However, the reachable sets (red, dashed in Fig. 5) are very large and not suitable for a maneuver automaton at all. Therefore, we cannot achieve both a good final set and input satisfaction with LQR controllers. This shows the advantage of the convex control approach, which optimizes the reachable set while ensuring the satisfaction of the constraints.

7. DISCUSSION OF THE ALGORITHM

Let us now discuss the complexity and optimality of the proposed algorithms.

7.1 Optimality

Finding optimal solutions for a nonlinear, disturbed system is a hard task, and in most cases it is not possible to obtain globally optimal solutions. This is also true for the nonlinear programming algorithms which we use to obtain the solutions for the reference trajectory and the corner trajectories.

While we have no guarantees that our overall dynamics results in a globally optimal solution, numerical solvers for optimizing open-loop trajectories work very efficiently and converge for many initial states to (locally) optimal solutions. Direct optimization methods in particular have shown very good performance in recent years [6]. With our method we are able to transfer the good results from optimizing single, open-loop trajectories to a whole set of trajectories. Through the iterative application, we obtain therefore a robust, closed-loop algorithm. It is a general problem to efficiently obtain globally optimal solutions for nonlinear disturbed systems [29, 5].

While this general problem is still unsolved and even though we have no guarantees for optimality (which no efficient approach can provide for disturbed, nonlinear systems), we have guarantees that the computed final set is reached and all constraints are satisfied despite disturbances. As we see in Sec. 6, the optimized solution performs much better than other methods using standard LQR tracking controllers. Since the computation is done offline, one can make changes in the parameters (e.g., weighting matrices in the cost functions or number of time steps) or the desired final state if the results are not satisfying. This is of course also the case if the shifted final set is not in the initial set, as desired for maneuver automata. In this case we have to change parameters or know that these maneuvers cannot be combined.

7.2 Complexity

In order to discuss the complexity for our algorithm, we have to distinguish between offline and online complexity, where online complexity is more important, as it restricts sampling times and therefore control performance. For offline complexity, we are not able to provide an exact bound, since nonlinear programming algorithms have no convergence guarantees. As mentioned before, even though they have no guarantees, they are often quite fast, especially for short time horizons, as needed for the corner trajectories. We can argue that if the optimal control problem is hardly solvable for undisturbed open-loop trajectories, then we cannot expect it to be solved more easily for a set of initial states and disturbances. If we can solve the optimal control problems for the extreme states, then the complexity of our algorithm grows with 2^n, with n denoting the dimension of the state space, since a parallelotope has 2^n extreme states for which we have to compute the optimal input trajectories. While this is exponential, it is much better than other comparable algorithms, which rely on discretizing the state space (like explicit MPC or abstraction-based control) and have exponential complexity with a larger base: k^n, where k denotes the number of discretized states in each dimension. Therein, k can easily be 30 or 100. The reachability analysis which we use has a complexity of $\mathcal{O}(n^3)$, which is therefore negligible for the overall complexity.

The online complexity of our approach is very low, as it consists only of matrix vector multiplications and in the case of closed-form expression of the convex combinations, plugging the current state into a closed-form formula. Clearly, since we have 2^n extreme points, we have to compute 2^n weights, but each computation can be done very fast (around 0.1ms for all weights of a ten-dimensional system [28]). If we use the linear approximation of the inputs as presented in Sec. 5, the computation simplifies to a matrix vector multiplication only and has complexity $\mathcal{O}(n^2)$. Therefore, we can have high sampling times, as high as we could achieve with piecewise-constant feedback matrices. The efficiency of our approach can also be seen in the fast computation times of around 10 seconds for the offline part and around 0.01 milliseconds for the online part, see Sec. 6.

8. CONCLUSION

In this paper, we present a novel control approach which solves reach-avoid problems and can be used to generate robust maneuver automata. The approach allows us to steer all states from an initial set to a final set by only computing optimal input trajectories for the extreme states. By interpolating between these extreme state inputs using convex combinations, we obtain fast online controllers with very low computational complexity, as most computation tasks can be performed offline in advance. The use of tools from formal verification allows us to achieve provable safety and formal guarantees. The approach works for linear and nonlinear systems even in the presence of disturbances.

The presented control approach is a novel way of viewing closed-loop control by combining the optimized solutions from open-loop control with the stability and robustness of feedback control. By using closed-form expressions for the convex combinations and linear approximations for the control inputs, we provide two ways to make the convex controllers even more efficient and faster, so that the online complexity is only $\mathcal{O}(n^2)$. The applicability is shown in the numerical example, where we are able to obtain a completely connected maneuver automaton for an autonomous vehicle.

We believe that the convex control approach is a very useful tool in control theory which can be used for many applications. Therefore, there are many future extensions possible. The next steps include computing of larger maneuver automata and testing them on real autonomous cars, as well as applying our approach to other application areas.

Acknowledgments

The author gratefully acknowledges financial support from the European Commission project UnCoVerCPS under grant number 643921 and the German Research Foundation (DFG) under grant number AL 1185/3-1.

9. 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] M. Althoff and G. Frehse. Combining zonotopes and support functions for efficient reachability analysis of linear systems. In *Proc. of the 55th IEEE Conference on Decision and Control*, pages 7439–7446, 2016.

[4] 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.

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

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

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

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

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

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

[11] T. X. T. Dang. *Vérification Et Synthèse Des Systèmes Hybrides*. PhD thesis, Institut National Polytechnique de Grenoble, 2000.

[12] J. A. DeCastro and H. Kress-Gazit. Nonlinear controller synthesis and automatic workspace partitioning for reactive high-level behaviors. In *Proc. Hybrid Systems: Computation and Control*, 2016.

[13] 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.

[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] B. Houska, H. Ferreau, and M. Diehl. ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization. *Optimal Control Applications and Methods*, 32(3):298–312, 2011.

[16] 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.

[17] 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.

[18] 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.

[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. Lunze and F. Lamnabhi-Lagarrigue. *Handbook of hybrid systems control: theory, tools, applications*. Cambridge University Press, 2009.

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

[22] A. Majumdar and R. Tedrake. Robust online motion planning with regions of finite time invariance. In *Algorithmic Foundations of Robotics X*, pages 543–558. Springer, 2013.

[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] A. Platzer and E. M. Clarke. The image computation problem in hybrid systems model checking. In *International Workshop on Hybrid Systems: Computation and Control*, pages 473–486, 2007.

[25] R. Rajamani. *Vehicle Dynamics and Control*. Springer, 2012.

[26] 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.

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

[28] B. Schürmann, A. El-Guindy, and M. Althoff. Closed-form expressions of convex combinations. In *Proc. of the American Control Conference*, pages 2795–2801, 2016.

[29] O. Schütze. *Set Oriented Methods for Global Optimization*. PhD thesis, Univ. Paderborn, 2004.

[30] 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.

[31] P. Tsiotras and R. S. Diaz. Real-time near-optimal feedback control of aggressive vehicle maneuvers. In *Optimization and Optimal Control in Automotive Systems*, pages 109–129. Springer, 2014.

[32] A. Weber and G. Reissig. Classical and strong convexity of sublevel sets and application to attainable sets of nonlinear systems. *SIAM Journal on Control and Optimization*, 52(5):2857–2876, 2014.

[33] E. M. Wolff, U. Topcu, and R. M. Murray. Automaton-guided controller synthesis for nonlinear systems with temporal logic. In *Proc. of the International Conference on Intelligent Robots and Systems*, pages 4332–4339, 2013.

[34] M. Zamani, G. Pola, M. Mazo Jr., and P. Tabuada. Symbolic models for nonlinear control systems without stability assumptions. *IEEE Transactions on Automatic Control*, 57(7):1804–1809, 2012.