

Provably Safe Reinforcement Learning via Action Projection Using Reachability Analysis and Polynomial Zonotopes

NIKLAS KOCHDUMPER ^{1,2}, HANNA KRASOWSKI ¹, XIAO WANG ¹, STANLEY BAK ²,
AND MATTHIAS ALTHOFF ¹

¹Department of Computer Engineering, Technical University of Munich, 85748 Garching, Germany

²Department of Computer Science, Stony Brook University, Stony Brook, NY 11794 USA

CORRESPONDING AUTHORS: NIKLAS KOCHDUMPER; HANNA KRASOWSKI; XIAO WANG (e-mail: niklas.kochdumper@stonybrook.edu;
hanna.krasowski@tum.de; xiao.wang@tum.de)

This work was supported by the European Research Council (ERC) through the Project justITSELF under Grant 817629, in part by the German Research Foundation through the Research Training Group ConVeY under Grant GRK 2428, and in part by the Air Force Office of Scientific Research and the Office of Naval Research under Grants FA9550-19-1-0288, FA9550-21-1-0121, FA9550-23-1-0066, and N00014-22-1-2156. (Niklas Kochdumper, Hanna Krasowski, and

Xiao Wang contributed equally to this work.)

This article has supplementary downloadable material available at <https://doi.org/10.1109/OJCSYS.2023.3256305>, provided by the authors.

ABSTRACT While reinforcement learning produces very promising results for many applications, its main disadvantage is the lack of safety guarantees, which prevents its use in safety-critical systems. In this work, we address this issue by a safety shield for nonlinear continuous systems that solve reach-avoid tasks. Our safety shield prevents applying potentially unsafe actions from a reinforcement learning agent by projecting the proposed action to the closest safe action. This approach is called action projection and is implemented via mixed-integer optimization. The safety constraints for action projection are obtained by applying parameterized reachability analysis using polynomial zonotopes, which enables to accurately capture the nonlinear effects of the actions on the system. In contrast to other state-of-the-art approaches for action projection, our safety shield can efficiently handle input constraints and dynamic obstacles, eases incorporation of the spatial robot dimensions into the safety constraints, guarantees robust safety despite process noise and measurement errors, and is well suited for high-dimensional systems, as we demonstrate on several challenging benchmark systems.

INDEX TERMS Action projection, reach-avoid problems, reachability analysis, reinforcement learning.

I. INTRODUCTION

Reinforcement learning has been successfully applied to find solutions for many challenging applications, such as robotics [1], autonomous driving [2], and power systems [3]. Many of these applications are safety-critical, so that the lack of safety guarantees for standard reinforcement learning controllers prevents their deployment in the real world. We aim to overcome this limitation with a novel safety shield for reinforcement learning agents that considers the very general case of disturbed nonlinear continuous systems with input constraints that have to avoid dynamic obstacles. Note that our safety shield can be applied to arbitrary unsafe controllers, while reinforcement learning is the main focus of this work.

A. STATE OF THE ART

We first provide a summary of the current state of the art in safety-related methods of reinforcement learning. The term *safe reinforcement learning* refers to approaches that aim to obtain safe agents, but do not provide hard safety guarantees. One example for this is constrained reinforcement learning [4], [5], where the objective of the training phase is to maximize the reward while satisfying safety constraints. While advantages of this technique are that no system model is required and that even complex temporal logic safety specifications [6], [7] can be considered, the obvious disadvantage is that hard safety guarantees can be provided during neither training nor deployment. The same is true for probabilistic

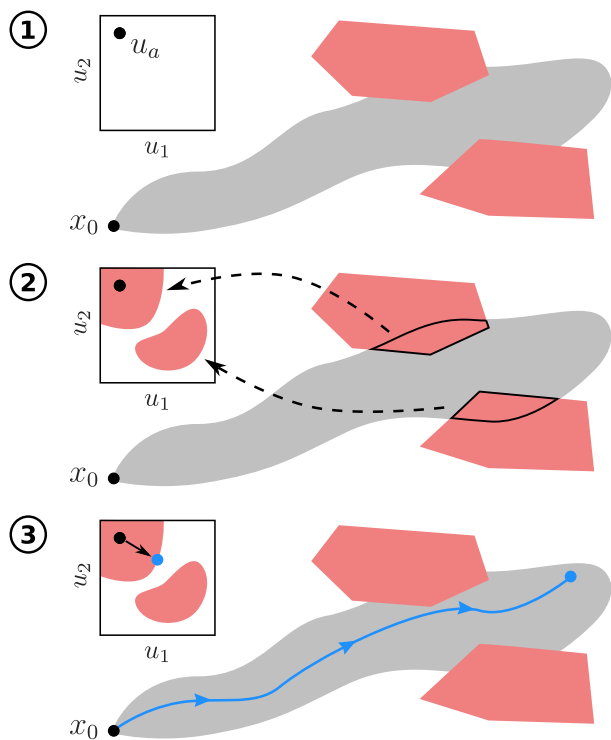


FIGURE 1. Steps for action projection using parameterized reachability analysis, where the reachable set is depicted in gray and the unsafe regions are shown in red: 1) Computation of the reachable set for all actions starting from the current state x_0 . 2) Extraction of action constraints from the intersections between the reachable set and unsafe regions. 3) Projection of the action u_a outputted by the agent to the closest safe action.

approaches [8], [9] that aim to identify the safety probability of an action. Overall, safe reinforcement learning techniques can be used for non-critical applications, where unsafe actions do not cause major damage; however, these methods are not suited for safety-critical systems.

In contrast to safe reinforcement learning, *provably safe reinforcement learning* approaches provide hard safety guarantees. They can be divided into the three main categories: *action masking*, *action replacement*, and *action projection* [10]. In action masking [11], [12], a mask that only allows the agent to choose actions from the set of safe actions is applied. One disadvantage of this method is that it is often hard to explicitly compute the set of safe actions, especially for continuous action spaces, where the set of safe actions often has a very complex non-convex shape as shown in Fig. 1. In addition, it is non-trivial to correctly consider the masking during training so that the reinforcement learning algorithm is not perturbed [13]. For action replacement [14], [15], [16], [17], unsafe actions returned by the agent are replaced by safe actions. As replacement, one can either use a single safe action obtained from a failsafe planner [14] or via human feedback [15] or one can sample from the set of safe actions [16], [17]. Also the well-known simplex architecture [18], [19], [20], where a safe controller is used as a backup for an unsafe

controller, can be categorized as action replacement. One disadvantage of action replacement is that the difference between the original action and the replacement action can be very large, which might prevent the agent from completing its task. Action projection tries to avoid this issue by finding the safe action that is closest to the action suggested by the agent.

Since our approach applies action projection, we discuss this category in more detail. The most prominent methods for action projection are *control barrier functions* [21], [22], *model predictive control* [23], [24], and *parameterized reachability analysis* [25]. A control barrier function is a level-set function that divides the state space into a safe and a potentially unsafe region. Here, action projection is formulated as an optimization problem, where the correction of the action is minimized, such that the system stays inside the safe region defined by the control barrier function. While an advantage is that control barrier functions can for static environments guarantee safety for infinite time, the method also has several disadvantages: 1) It is often not easy to find a suitable control barrier function, especially in the presence of dynamic obstacles. 2) Control barrier functions are often quite conservative since they usually exclude many states that are safe. 3) The approach is often limited to control affine systems because the optimization problems would otherwise become non-convex. 4) It is challenging to consider input constraints as well as process noise and measurement errors. The second method is model predictive control, which also formulates the projection as an optimization problem, but uses the safety constraint that the system should not enter any unsafe regions for a certain finite prediction horizon, which avoids the requirement for a control barrier function. However, one downside is that it is often not possible to guarantee that the solution is robustly safe despite process noise and measurement errors since for non-linear systems these uncertainties usually cannot be encoded directly into the optimization problem. Our safety shield is based on the parameterized reachability analysis approach, which is visualized in Fig. 1: The first step is to compute the reachable set for all available actions. Since this reachable set is parameterized by the actions, one can directly extract the safety constraints for action projection from the intersection between the reachable sets and the unsafe regions. Since process noise as well as measurement errors can conveniently be integrated into reachability analysis, this approach is very well suited for guaranteeing robust safety.

Due to its advantageous properties, several approaches apply reachability analysis to guarantee safety. One method [26] uses the Hamilton-Jacobi reachability framework [27] to compute the backward reachable set starting from the unsafe sets — a state is safe for all possible actions if it is outside of the backward reachable set. This has the disadvantage that for each unsafe set a different backward reachable set has to be computed. Moreover, the Hamilton-Jacobi framework requires gridding the state space so that the computational complexity of the approach grows exponentially with the system dimension. Another method [28] applies reachability analysis for black-box systems and uses a differentiable

collision check that is based on constrained zonotopes [29] to efficiently push the reachable set for the proposed action away from unsafe sets. This, however, has the drawback that the reachable set has to be recomputed after each correction update of the action, which is computationally demanding. The method closest to our approach is a *reachability-based trajectory safeguard* [25], which computes the parameterized reachable set for a simplified trajectory-generating model and determines a safe action satisfying the constraints extracted from the reachable set via random sampling. While this approach can be computationally efficient for some systems, sampling methods often fail to find feasible solutions, especially in high-dimensional action spaces.

B. CONTRIBUTIONS AND OUTLINE

We present a novel safety shield that is based on action projection using parameterized reachability analysis. This safety shield extends our previous work on dependency preserving reachability analysis [30], [31], [32] by a method for correcting unsafe actions, and we additionally also study the effect online verification has on the learning process. Unlike the related approach in [25], our safety shield directly operates on the original nonlinear system model rather than on a simplified trajectory-generating model. Moreover, in contrast to [25], we use conservative polynomialization [30] instead of conservative linearization [33] for reachability analysis, which enables us to efficiently capture the nonlinear effects the actions have on the system. Another advantage over [25] is that we use mixed-integer optimization instead of random sampling for projection, which always finds the action with the smallest correction. Finally, the various design choices provided by our safety shield enable the user to fine-tune its performance for the considered application.

The remainder of this paper is structured as follows: After introducing some preliminaries in Section II, we provide the problem definition in Section III. Our main contribution is the reachability-based safety shield for reinforcement learning presented in Section IV, for which we discuss several extensions in Section V. Finally, we demonstrate our approach on several numerical examples in Section VI and conclude with a discussion of its properties in Section VII.

II. PRELIMINARIES

We first introduce our notation and define the set representations that we use in this paper.

A. NOTATION

Sets are denoted by calligraphic letters, matrices by uppercase letters, and vectors by lowercase letters. Given a vector $a \in \mathbb{R}^n$, $a_{(i)}$ is the i -th entry and the p -norm is denoted by $\|a\|_p$. Given a matrix $A \in \mathbb{R}^{n \times m}$, $A_{(i,\cdot)}$ represents the i -th matrix row, $A_{(\cdot,j)}$ the j -th column, and $A_{(i,j)}$ the j -th entry of matrix row i . The concatenation of two matrices C and D is denoted by $[CD]$, $I_n \in \mathbb{R}^{n \times n}$ is the identity matrix, and the symbols $\mathbf{0}$ and $\mathbf{1}$ represent vectors of zeros and ones of proper dimension. We further introduce an n -dimensional interval as $\mathcal{I} :=$

$[\underline{x}, \bar{x}]$, $\forall i \underline{x}_{(i)} \leq \bar{x}_{(i)}$, $\underline{x}, \bar{x} \in \mathbb{R}^n$. Given two sets $\mathcal{S}_1, \mathcal{S}_2 \subset \mathbb{R}^n$, their Minkowski sum is $\mathcal{S}_1 \oplus \mathcal{S}_2 = \{s_1 + s_2 \mid s_1 \in \mathcal{S}_1, s_2 \in \mathcal{S}_2\}$ and their Cartesian product is $\mathcal{S}_1 \times \mathcal{S}_2 = \{[s_1^T \ s_2^T]^T \mid s_1 \in \mathcal{S}_1, s_2 \in \mathcal{S}_2\}$.

B. SET REPRESENTATIONS

Our approach relies on several different set representations, which we introduce here. Let us begin with polytopes, for which we use the halfspace representation:

Definition 1 (Polytope): Given a constraint matrix $A \in \mathbb{R}^{s \times n}$ and a constraint offset $b \in \mathbb{R}^s$, the halfspace representation of a polytope $\mathcal{P} \subseteq \mathbb{R}^n$ is

$$\mathcal{P} := \{x \in \mathbb{R}^n \mid Ax \leq b\}.$$

We use the shorthand $\mathcal{P} = \langle A, b \rangle_{\mathcal{P}}$.

Zonotopes are a special type of polytopes that can be represented efficiently using generators:

Definition 2 (Zonotope): Given a center vector $c \in \mathbb{R}^n$ and a generator matrix $G \in \mathbb{R}^{n \times p}$, a zonotope $\mathcal{Z} \subset \mathbb{R}^n$ is

$$\mathcal{Z} := \left\{ c + \sum_{i=1}^p G_{(\cdot,i)} \alpha_i \mid \alpha_i \in [-1, 1] \right\}$$

with so-called factors α_i . We use the shorthand $\mathcal{Z} = \langle c, G \rangle_{\mathcal{Z}}$.

An extension to zonotopes are polynomial zonotopes [30], which can represent non-convex sets. We use the sparse representation of polynomial zonotopes [32]:¹

Definition 3 (Polynomial Zonotope): Given a constant offset $c \in \mathbb{R}^n$, a generator matrix of dependent generators $G \in \mathbb{R}^{n \times h}$, a generator matrix of independent generators $G_I \in \mathbb{R}^{n \times q}$, and an exponent matrix $E \in \mathbb{N}_0^{p \times h}$, a polynomial zonotope $\mathcal{PZ} \subset \mathbb{R}^n$ is

$$\mathcal{PZ} := \left\{ c + \sum_{i=1}^h \left(\prod_{k=1}^p \alpha_k^{E_{(k,i)}} \right) G_{(\cdot,i)} + \sum_{j=1}^q \beta_j G_{I(\cdot,j)} \mid \alpha_k, \beta_j \in [-1, 1] \right\}.$$

The scalars α_k are called dependent factors and β_j independent factors. We use the shorthand $\mathcal{PZ} = \langle c, G, G_I, E \rangle_{\mathcal{PZ}}$.

Polynomial zonotopes can equivalently represent intervals, zonotopes, polytopes, and Taylor models [32, Sec. II.B]. Moreover, due to their polynomial nature, they are closely related to polynomial level sets:

Definition 4 (Polynomial Level Set): Given a vector of coefficients $a \in \mathbb{R}^h$, an offset $b \in \mathbb{R}$, and an exponent matrix $E \in \mathbb{N}_0^{n \times h}$, a polynomial level set $\mathcal{LS} \subseteq \mathbb{R}^n$ is

$$\mathcal{LS} := \left\{ x \in \mathbb{R}^n \mid \sum_{i=1}^h \left(\prod_{k=1}^n x_{(k)}^{E_{(k,i)}} \right) a_{(i)} \leq b \right\}.$$

We use the shorthand $\mathcal{LS} = \langle a, b, E \rangle_{\mathcal{LS}}$.

¹In contrast to [32, Def. 1] we do not integrate the constant offset c into G . Moreover, we omit the identifier vector used in [32] for simplicity

III. PROBLEM FORMULATION

We consider general nonlinear disturbed systems with input constraints defined by the ordinary differential equation

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

where $x(t) \in \mathbb{R}^n$ is the system state, $u(t) \in \mathbb{R}^m$ is the control input, $w(t) \in \mathbb{R}^z$ is the process noise, $f: \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^z \rightarrow \mathbb{R}^n$ is a Lipschitz continuous function, and $t \in \mathbb{R}^+$ is time. The process noise is bounded by a compact set $w(t) \in \mathcal{W} \subset \mathbb{R}^z$ and the system has to satisfy the input constraints defined by the convex set $u(t) \in \mathcal{U} \subseteq \mathbb{R}^m$. The set \mathcal{W} can for example be determined from measurements of the real physical system using conformance checking [34].

Given a nonlinear system defined as in (1), the goal is to solve a reach-avoid problem, where the system state should be steered from the current state $x_0 = x(0)$ to a goal set $\mathcal{G} \subseteq \mathbb{R}^n$ while avoiding collisions with potentially time-varying unsafe sets $\mathcal{F}_i \subset \mathbb{R}^n$, $i = 1, \dots, o$, where o denotes the number of unsafe sets. In case the measurements of the system state are subject to a measurement error $v(t) \in \mathcal{V}$, the goal becomes to steer all states in the set $x_0 \oplus \mathcal{V}$ to the goal set. We aim to solve reach-avoid problems with reinforcement learning, where we train an agent to return the control inputs $u_a(t)$ for a given state $x(t)$ steering the system to the goal set while avoiding obstacles. However, we have no guarantee that the behavior learned by the agent is safe. Therefore, we add a safety shield that is based on reachability analysis to obtain formal guarantees:

Definition 5 (Reachable Set): Let $\xi(t, x_0, u(\cdot), w(\cdot))$ denote the solution of (1) at time t for an initial state $x_0 = x(0)$, control input trajectory $u(\cdot)$ and process noise trajectory $w(\cdot)$. The reachable set at time t is

$$\mathcal{R}(t) := \left\{ \xi(t, x_0, u(\cdot), w(\cdot)) \mid x_0 \in \mathcal{X}_0, \forall \tau \in [0, t] : w(\tau) \in \mathcal{W} \right\},$$

where $\mathcal{X}_0 \subset \mathbb{R}^n$ is the initial set and $\mathcal{W} \subset \mathbb{R}^z$ is the set of process noise.

For our safety shield, we consider that \mathcal{U} , \mathcal{W} , and \mathcal{V} are represented as zonotopes, and \mathcal{G} and \mathcal{F}_i are represented as polytopes in halfspace representation. Moreover, we use polynomial zonotopes to represent reachable sets. In case other agents are present in the environment, we can apply set-based methods [35] to safely predict their future behavior and obtain the corresponding time-varying unsafe sets.

IV. SAFETY SHIELD

As visualized in Fig. 1, the high-level idea behind our safety shield is to compute the reachable set for a time horizon of t_f and the set of all control inputs satisfying the input constraints $\forall t \in [0, t_f] : u(t) \in \mathcal{U}$ rather than a single control input trajectory $u(\cdot)$. The intersection of this reachable set with the unsafe sets then yields constraints that define safe control inputs, which we can use to formulate the projection of the control input u_a provided by the reinforcement policy to the closest safe control input as an optimization problem. We

first consider input trajectories that are constant over time for simplicity and discuss more advanced control strategies later in Section V-A. For constant control inputs, we can compute the reachable set using the extended system dynamics

$$\begin{bmatrix} \dot{x}(t) \\ \dot{u}(t) \end{bmatrix} = \begin{bmatrix} f(x(t), u(t), w(t)) \\ \mathbf{0} \end{bmatrix} \quad (2)$$

together with the initial set $\mathcal{X}_0 = x_0 \times \mathcal{U}$, where we omit the set of measurement errors \mathcal{V} for simplicity. For reachability analysis, we use the conservative polynomialization algorithm [30], which encloses the nonlinear dynamics in (1) by a differential inclusion $\dot{x} \in p(x(t), u(t), w(t)) \oplus \mathcal{E}$ consisting of a polynomial approximation $p(x(t), u(t), w(t))$ and the abstraction error \mathcal{E} . This reachability algorithm explicitly preserves dependencies between the initial states and the reachable states [31]. Since with the extended system dynamics in (2), the control inputs become part of the system state, we can therefore directly determine from the reachable set which control inputs steer the system to unsafe regions. Let us demonstrate this dependency preservation by an example:

Example 1: As a running example we consider the system

$$\dot{x}_1 = 4 + 2x_2u_1 + w_1, \quad \dot{x}_2 = 1.7 + u_1u_2$$

with initial state $x_0 = [0 \ 0]^T$, set of process noise $\mathcal{W} = [-0.01, 0.01]$, and time horizon $t_f = 1$ s. Moreover,

$$\mathcal{F} = \langle \langle [-4 \ -1]^T \ [-1 \ -4]^T \rangle, [-14 \ -8]^T \rangle_P$$

is the unsafe set and

$$\mathcal{U} = \left\{ \left[\begin{array}{c} -0.5 \\ 1 \end{array} \right] + \left[\begin{array}{c} 0.5 \\ 0 \end{array} \right] \alpha_1 + \left[\begin{array}{c} 0 \\ 1 \end{array} \right] \alpha_2 \mid \alpha_1, \alpha_2 \in [-1, 1] \right\}$$

is the set of control inputs. With the conservative polynomialization algorithm we obtain the final reachable set

$$\begin{aligned} \mathcal{R}(t_f) = & \left\{ \left[\begin{array}{c} 3.4 \\ 1.2 \end{array} \right] + \left[\begin{array}{c} 0.34 \\ 0.5 \end{array} \right] \alpha_1 + \left[\begin{array}{c} 0.25 \\ -0.5 \end{array} \right] \alpha_2 + \left[\begin{array}{c} -0.49 \\ 0.5 \end{array} \right] \alpha_1 \alpha_2 \right. \\ & \left. + \left[\begin{array}{c} 0.25 \\ 0 \end{array} \right] \alpha_1^2 + \left[\begin{array}{c} 0.25 \\ 0 \end{array} \right] \alpha_1^2 \alpha_2 + \left[\begin{array}{c} 0.1 \\ 0 \end{array} \right] \beta_1 \mid \alpha_1, \alpha_2, \beta_1 \in [-1, 1] \right\}, \end{aligned}$$

which is visualized in Fig. 2. Since the reachable set $\mathcal{R}(t_f)$ and the input set \mathcal{U} are parameterized by the same factors α_1 and α_2 , we have a direct analytical relation between the control inputs and the corresponding reachable states.

We now exploit the analytical relation between the control inputs and the reachable states to determine the set of safe control inputs. As demonstrated in the example above, the control input $u(t) \in \mathcal{U} = \langle c_u, G_u \rangle_Z$ is unambiguously defined by the factors α via the relation $u = c_u + G_u \alpha$ through the definition of a zonotope in Def. 2. Instead of determining the set of safe control inputs directly, we therefore determine the safe set for α instead, since this simplifies the computations as it becomes apparent later. The independent generators of

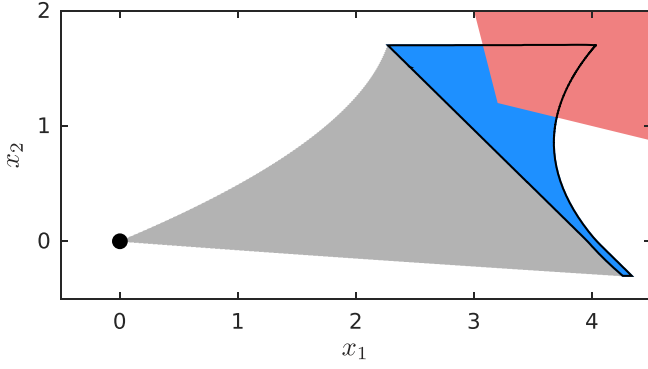


FIGURE 2. Reachable set for the system from Example 1, where the initial state x_0 is shown as a black dot, the final reachable set $\mathcal{R}(t_f)$ is depicted in blue with a black border, and the unsafe set \mathcal{F} is shown in red.

the polynomial zonotope $\mathcal{R}(t_f)$ represent uncertainties that results from abstraction errors during reachability analysis as well as from the process noise. Consequently, a control input is safe only if the reachable set does not intersect the unsafe sets for any possible value of the independent factors β_j . We formulate this in the following theorem, which extends our previous results for unsafe sets given as halfspaces [31, Sec. 4.1] to the more general case of polytopes:

Theorem 1: Given is an unsafe set $\mathcal{F} = \langle A, b \rangle_P \subset \mathbb{R}^n$ consisting of s halfspace constraints and the reachable set $\mathcal{R}(t) = \langle c, G, G_I, E \rangle_{PZ} \subset \mathbb{R}^n$ of the system in (2) computed with the conservative polynomialization algorithm [30] for the initial set $\mathcal{X}_0 = x_0 \times \mathcal{U}$, $x_0 \in \mathbb{R}^n$, $\mathcal{U} = \langle c_u, G_u \rangle_Z \subset \mathbb{R}^m$ and the set of process noise $\mathcal{W} \subset \mathbb{R}^z$. The following constraints on the zonotope factors α that parameterize the control input ensure that there exists no trajectory that enters the unsafe set:

$$\forall \alpha \in [-1, 1] \cap \bigcup_{l=1}^s \mathcal{LS}_l, \forall w(\cdot) \in \mathcal{W} : \xi(t, x_0, c_u + G_u \alpha, w(\cdot)) \notin \mathcal{F}$$

with

$$\mathcal{LS}_l = \left\langle -A_{(l,\cdot)}G, A_{(l,\cdot)}c - \sum_{j=1}^q |A_{(l,\cdot)}G_{I(\cdot,j)}| - b_{(l)}, E \right\rangle_{LS}$$

for $l = 1, \dots, s$.

Proof: A single point $x \in \mathbb{R}^n$ is located outside the unsafe set \mathcal{F} if it is fully located outside of at least one halfspace:

$$\bigvee_{l=1}^s A_{(l,\cdot)}x > b_{(l)} \Rightarrow x \notin \mathcal{F}. \quad (3)$$

Moreover, due to dependency preservation of reachability analysis, it holds according to [31, Thm. 1] that the disturbed trajectory $\xi(t, x_0, c_u + G_u \alpha, w(\cdot))$ for a specific control input $u = c_u + G_u \alpha$ is contained inside the reachable subset obtained by restricting the factors $\alpha_k \in [-1, 1]$ in the definition of polynomial zonotopes in Def. 3 to the corresponding concrete value for $\alpha = [\alpha_1 \dots \alpha_p]^T$:

$$\forall \alpha_k \in [-1, 1] : \xi(t, x_0, c_u + G_u \alpha, w(\cdot)) \in$$

$$c + \sum_{i=1}^h \left(\prod_{k=1}^p \alpha_k^{E_{(k,i)}} \right) G_{(\cdot,i)} + \left\{ \sum_{j=1}^q \beta_j G_{I(\cdot,j)} \mid \beta_j \in [-1, 1] \right\}.$$

Finally, combining this with (3) under the consideration that the constraints should hold for all values of the independent factors β_j yields

$$\forall \alpha_k, \beta_j \in [-1, 1] : \bigvee_{l=1}^s A_{(l,\cdot)}c + \sum_{i=1}^h \left(\prod_{k=1}^p \alpha_k^{E_{(k,i)}} \right) A_{(l,\cdot)}G_{(\cdot,i)} + \sum_{j=1}^q \beta_j A_{(l,\cdot)}G_{I(\cdot,j)} > b_{(l)} \Rightarrow \xi(t, x_0, c_u + G_u \alpha, w(\cdot)) \notin \mathcal{F},$$

which results in the statement of the theorem after bringing the constant offset and the independent generators to the other side of the inequality. ■

Remark 1: A geometric interpretation of Theorem 1 is that we first bloat the obstacle \mathcal{F} by the uncertainty given by the independent generators through pushing each polytope halfspace outward. Next, we obtain the constraints via intersecting with the part of the polynomial zonotope spanned by the dependent generators, where the intersection between each halfspace of the bloated polytope \mathcal{F} corresponds to a polynomial level set constraint for the factors α .

Theorem 1 defines a feasible region $\alpha \in [-1, 1] \cap \bigcup_l \mathcal{LS}_l$ for the factors α that parameterize the control input such that the intersection between a reachable set at a specific point in time and a single unsafe set is empty. However, to guarantee safety we have to consider the reachable set for the whole time horizon $t \in [0, t_f]$, which consists of a sequence of reachable sets $\mathcal{R}(\tau_0), \mathcal{R}(\tau_1), \dots, \mathcal{R}(\tau_f)$ for consecutive time intervals $\tau_0, \tau_1, \dots, \tau_f$. Moreover, we might also have more than one unsafe set. So overall we obtain one feasible region $\alpha \in [-1, 1] \cap \bigcup_l \mathcal{LS}_l$ for each pair of reachable sets and unsafe sets resulting in an intersection. The feasible region for α to guarantee safety for all time intervals and all unsafe sets is given by the intersection of the feasible regions for single pairs:

$$\alpha \in [-1, 1] \cap \bigcap_{r=1}^v \bigcup_{l=1}^{s_r} \underbrace{\langle a_{rl}, b_{rl}, E_{rl} \rangle_{LS}}_{\mathcal{LS}_{rl}}$$

where the level sets \mathcal{LS}_{rl} are obtained from Theorem 1 and v is the number of intersecting pairs. To efficiently check if a reachable set represented by a polynomial zonotope intersects an obstacle represented by a polytope, the polynomial zonotope refinement algorithm [36] can be used. This algorithm recursively splits the polynomial zonotope along the longest generator until the intersection with the polytope can either be proven or disproven using zonotope enclosures of the split polynomial zonotopes. Overall, given a vector of factors $\alpha_a \in \mathbb{R}^p$ that corresponds to the control input $u_a = c_u + G_u \alpha_a \in \mathcal{U} = \langle c_u, G_u \rangle_Z$ provided by the reinforcement learning policy, we can formulate the projection to the closest safe control

input as an optimization problem:

$$\min_{\alpha \in [-1,1]} \|\alpha - \alpha_a\|_2^2 \text{ s.t. } \alpha \in \bigcap_{r=1}^v \bigcup_{l=1}^{s_r} \langle a_{rl}, b_{rl}, E_{rl} \rangle_{LS}.$$

This is a disjunctive programming problem, which can be formulated as a mixed-integer quadratic program with polynomial constraints using the convex hull relaxation [37]:

$$\min_{\alpha \in [-1,1]} \|\alpha - \alpha_a\|_2^2 \quad (4)$$

subject to

$$\sum_{i=1}^h \left(\prod_{k=1}^p \alpha_{r_l(k,i)}^{E_{r_l(k,i)}} \right) a_{r_l(\cdot,k)} \lambda_{r_l} \leq -b_{r_l} \lambda_{r_l},$$

$$\lambda_{r_l} \in \{0, 1\}, \alpha = \sum_{l=1}^{s_r} \lambda_{r_l} \alpha_{r_l}, \sum_{l=1}^{s_r} \lambda_{r_l} = 1,$$

for $r = 1, \dots, v$ and $l = 1, \dots, s_r$. Here, the disjunction is realized using the binary variables $\lambda_{r_l} \in \{0, 1\}$ which modify the corresponding polynomial constraints to be either active ($\lambda_{r_l} = 1$) or inactive ($\lambda_{r_l} = 0$). Let us demonstrate the optimization for our running example:

Example 2: As shown in Fig. 2, for the nonlinear system in Example 1 only the final reachable set $\mathcal{R}(t_f)$ intersects the unsafe set \mathcal{F} . We consequently obtain the feasible region for α by applying Theorem 1 to the sets $\mathcal{R}(t_f)$ and \mathcal{F} , which yields $\alpha \in \mathcal{LS}_1 \vee \mathcal{LS}_2$ with

$$\mathcal{LS}_1 = \left\{ [\alpha_1 \alpha_2]^T \in \mathbb{R}^2 \mid 1.86 \alpha_1 + 0.5 \alpha_2 - 1.46 \alpha_1 \alpha_2 + \alpha_1^2 + \alpha_2^2 \leq -1.2 \right\}$$

and

$$\mathcal{LS}_2 = \left\{ [\alpha_1 \alpha_2]^T \in \mathbb{R}^2 \mid 2.34 \alpha_1 - 1.75 \alpha_2 + 1.51 \alpha_1 \alpha_2 + 0.25 \alpha_1^2 + 0.25 \alpha_2^2 \leq -0.3 \right\}.$$

Given $\alpha_a = [0.3 \ 0]^T$, the optimization problem (4) becomes

$$\min_{\alpha_1, \alpha_2 \in [-1,1]} (\alpha_1 - 0.3)^2 + \alpha_2^2$$

subject to

$$1.86 \alpha_{11(1)} \lambda_{11} + 0.5 \alpha_{11(2)} \lambda_{11} - 1.46 \alpha_{11(1)} \alpha_{11(2)} \lambda_{11} + \alpha_{11(1)}^2 \lambda_{11} + \alpha_{11(2)}^2 \lambda_{11} \leq -1.2 \lambda_{11},$$

$$2.34 \alpha_{12(1)} \lambda_{12} - 1.75 \alpha_{12(2)} \lambda_{12} + 1.51 \alpha_{12(1)} \alpha_{12(2)} \lambda_{12} + 0.25 \alpha_{12(1)}^2 \lambda_{12} + 0.25 \alpha_{12(2)}^2 \lambda_{12} \leq -0.3 \lambda_{12},$$

$$\lambda_{11}, \lambda_{12} \in \{0, 1\}, \lambda_{11} + \lambda_{12} = 1$$

$$\begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} = \lambda_{11} \begin{bmatrix} \alpha_{11(1)} \\ \alpha_{11(2)} \end{bmatrix} + \lambda_{12} \begin{bmatrix} \alpha_{12(1)} \\ \alpha_{12(2)} \end{bmatrix},$$

which has the optimal solution $\alpha = [\alpha_1 \ \alpha_2]^T = [0.04 \ 0.2]^T$. The feasible regions for α_1 and α_2 are shown in Fig. 3.

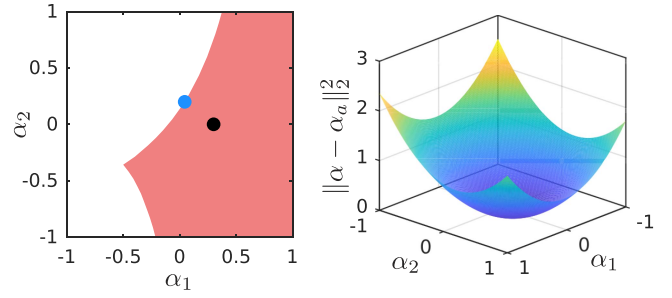


FIGURE 3. Domain (left) and objective function (right) for the optimization problem from Example 2. For the domain plot the set of infeasible values is shown in red, the desired value α_a is visualized as a black dot, and the optimal solution to the optimization problem is depicted as a blue dot.

In the presence of measurement errors $v(t) \in \mathcal{V}$ we can apply the same overall approach but have to change the initial set to $\mathcal{X}_0 = (x_0 \oplus \mathcal{V}) \times \mathcal{U}$, where the set \mathcal{V} has to be represented by independent generators to ensure that safety is guaranteed for all possible values of the measurement errors. While we focused on the conservative polynomialization algorithm [30] for simplicity, our safety shield is also compatible with other reachability approaches as long as they preserve dependencies between initial states and reachable states. This is for example the case for algorithms that compute reachable sets using the Picard-Lindelöf iteration together with Taylor models [38].

The safety shield can be used during reinforcement learning or for a learned agent. For every decision step, the action suggested by the agent is corrected to the closest safe action by (4) only if it violates safety constraints. If the safety shield is used during learning, it can be beneficial to adapt the reward to inform the agent about corrections of actions [10].

V. EXTENSIONS

We now discuss several extensions for our safety shield.

A. DIFFERENT TYPES OF CONTROL LAWS

For the basic safety shield presented in Section IV, for simplicity we considered that the control input is kept constant for the whole planning horizon. Since this is very restrictive and would in practice often prevent us from finding a feasible solution, we now discuss how to realize more advanced control strategies. Note that the reinforcement learning agent has to match the control law used for the safety shield.

a) *Piecewise Constant Control Law:* One simple but very effective extension to constant control inputs are piecewise constant control inputs. Instead of determining a single control input from the input set \mathcal{U} , we determine control inputs for all piecewise constant segments from the set $\mathcal{U} \times \dots \times \mathcal{U}$. We can still use the extended system in (2), but have to reset the initial set for reachability analysis to $\mathcal{R}(t_i) \times \mathcal{U}$ after each of the $i = 1, \dots, \mu$ piecewise constant time segments $[t_{i-1}, t_i]$ with $t_i = i \cdot t_f / \mu$, where $\mathcal{R}(t_i)$ is the final reachable set from the previous segment.

b) *Polynomial Control Law:* Another possibility is to use control laws that are polynomial functions with respect to

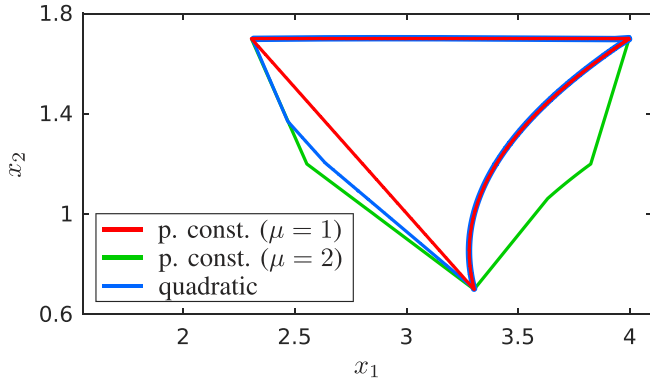


FIGURE 4. Final reachable set for the system in Example 1 for a quadratic control law and piecewise constant control laws with different numbers of segments μ .

time. We consider the quadratic case for simplicity since the extension to general polynomials is straightforward. For a quadratic control law $u(t) = c_{(1)} + c_{(2)}t + c_{(3)}t^2$ parameterized by the vector of coefficients $c \in \mathbb{R}^3$, we can use the extended system

$$\begin{bmatrix} \dot{x}(t) \\ \dot{c} \\ \dot{i} \end{bmatrix} = \begin{bmatrix} f(x(t), c_{(1)} + c_{(2)}t + c_{(3)}t^2, w(t)) \\ \mathbf{0} \\ 1 \end{bmatrix}$$

together with the initial set $x_0 \times \mathcal{C} \times 0$. In the optimization problem (4) we then determine the values for the parameter vector c , where we add the constraint $c_{(1)} + c_{(2)}t + c_{(3)}t^2 \in \mathcal{U}$ to ensure that the input constraints are satisfied. The initial set $\mathcal{C} \subset \mathbb{R}^3$ for the coefficient vector c can be determined by estimating the feasible values for c such that the constraint $c_{(1)} + c_{(2)}t + c_{(3)}t^2 \in \mathcal{U}$ is satisfied for the whole time horizon.

c) Feedback Control: We can also apply a feedback control law $u(t) = u_{ref}(t) + K(x(t) - x_{ref}(t))$ with a fixed feedback matrix $K \in \mathbb{R}^{m \times n}$, where both piecewise constant or polynomial control inputs can be used for the reference control input $u_{ref}(t)$ corresponding to the reference trajectory $x_{ref}(t)$. For the safety shield, we then compute the reachable set for the extended system

$$\begin{bmatrix} \dot{x}(t) \\ \dot{u}_{ref}(t) \\ \dot{x}_{ref}(t) \end{bmatrix} = \begin{bmatrix} f(x(t), u_{ref}(t) + K(x(t) - x_{ref}(t)), w(t)) \\ \mathbf{0} \\ f(x_{ref}(t), u_{ref}(t), w(t)) \end{bmatrix}$$

using the initial set $x_0 \times \mathcal{U} \times x_0$. In the optimization problem (4) we then determine the optimal parameter for the reference control inputs $u_{ref}(t)$, where we add the constraint $u_{ref} + K(x(t) - x_{ref}(t)) \in \mathcal{U}$ to satisfy the input constraints.

A comparison of the different control laws presented in this section is shown in Fig. 4 for the system in Example 1. The results demonstrate that even for a piecewise constant control law with only $\mu = 2$ segments we already obtain a larger reachable set than with a quadratic control law, which

increases our chances to find a safe control input. While piecewise constant control laws therefore seem to be preferable, their rapidly changing values often negatively impact comfort or durability for many systems, which can be avoided with polynomial control laws.

For all control strategies we apply the following control scheme: We plan for a time horizon of t_f , but execute the resulting control law for only a shorter time period $t_c < t_f$ before planning a new trajectory. This increases the chances to avoid getting stuck in dead ends and ensures that we can react quickly to dynamic changes in the environment.

B. SPATIAL DIMENSIONS OF MOBILE ROBOTS

So far we considered the case where the safety constraints are specified directly by the system state. For collision avoidance, however, this setup is usually not sufficient since we additionally have to consider the shape and spatial dimension of mobile robots, e.g., cars, vessels, or drones, which we want to control safely. While for many other approaches this poses a huge problem, incorporating spatial dimensions of the robot into our safety shield is quite straightforward since we simply have to replace the reachable set with the occupancy set. Given the reachable set $\mathcal{R}(t)$ that typically describes all possible positions of a reference point on the robot as well as all possible robot orientations, the occupancy set is defined as

$$\mathcal{O}(t) = \{o(x, d) \mid x \in \mathcal{R}(t), d \in \mathcal{D}\}, \quad (5)$$

where the function $o: \mathbb{R}^n \times \mathbb{R}^\delta \rightarrow \mathbb{R}^Y$ describes how the space occupied by the robot is computed from the system state and the set \mathcal{D} specifying the spatial dimension of the robot.

Example 3: Let us consider a car where the states $x_{(1)}$ and $x_{(2)}$ describe the x- and y-position of its center, and state $x_{(3)}$ describes the orientation of the car. Then the function $o(x, d)$ that defines the space occupied by the car is given as

$$o(x, d) = \begin{bmatrix} x_{(1)} + \cos(x_{(3)})d_{(1)} - \sin(x_{(3)})d_{(2)} \\ x_{(2)} + \sin(x_{(3)})d_{(1)} + \cos(x_{(3)})d_{(2)} \end{bmatrix}, \quad (6)$$

where the shape of the car is for simplicity enclosed by a rectangle, so that $d \in \mathcal{D} = [-l/2, l/2] \times [-w/2, w/2]$ with l and w denoting the length and width of the car.

To compute the occupancy set (5) from the reachable set $\mathcal{R}(t)$ and the set \mathcal{D} using polynomial zonotopes, we suggest two approaches:

- 1) We can compute a Taylor series expansion enclosure of the function $o(x, d)$ and evaluate it in a set-based way to obtain the occupancy set $\mathcal{O}(t)$ [39, Sec. 4.4].
- 2) Since polynomial zonotopes can be converted to Taylor models [32, Prop. 4] we can apply Taylor model arithmetic [40] to evaluate (5) and then convert the resulting set back to a polynomial zonotope.

The resulting safety constraints that we obtain from the intersections between the occupancy set $\mathcal{O}(t)$ and obstacles have to hold for all values $d \in \mathcal{D}$. To ensure this, we could represent the set \mathcal{D} with independent generators before computing $\mathcal{O}(t)$, similarly as we did for the set of measurement

errors in Section IV. However, since the set \mathcal{D} is in general much larger than the set of measurement errors \mathcal{V} , this would often yield very conservative results. A better approach is to project out all factors that correspond to the set \mathcal{D} using Fourier-Motzkin elimination [41, Chapter 4.4]. Let us demonstrate this by an example:

Example 4: We consider the constraint

$$\forall \alpha_3 \in [-1, 1] : \alpha_1 + \alpha_2 + \alpha_3 + \alpha_1^2 \alpha_3 \leq 1.5, \quad (7)$$

from which we want to eliminate α_3 . The first step of Fourier-Motzkin elimination is to solve all constraints for α_3 , which yields

$$\alpha_3 \leq \frac{1.5 - \alpha_1 - \alpha_2}{1 + \alpha_1^2}, \quad \alpha_3 \leq 1, \quad \alpha_3 \geq -1. \quad (8)$$

Next, we have to form all combinations of the constraints in (8) that result in a non-empty solution, yielding the constraints

$$\begin{aligned} \frac{1.5 - \alpha_1 - \alpha_2}{1 + \alpha_1^2} \geq -1 &\Rightarrow -\alpha_1^2 + \alpha_1 + \alpha_2 \leq 2.5 \\ \frac{1.5 - \alpha_1 - \alpha_2}{1 + \alpha_1^2} \leq 1 &\Rightarrow \alpha_1^2 + \alpha_1 + \alpha_2 \geq 0.5, \end{aligned}$$

which represent an equivalent formulation of (7).

Since Fourier-Motzkin elimination requires that the constraints are solvable for the variable that is eliminated, all terms that violate this condition have to be removed first by applying a zonotope enclosure [32, Prop. 5].

C. MIXED-INTEGER LINEAR PROGRAM FORMULATION

For some systems, solving the nonlinear mixed-integer optimization problem (4) might be computationally too expensive, especially when we have to evaluate the safety shield in real-time for online application. Therefore, we now discuss how to obtain a feasible and close to optimal solution using mixed-integer linear programming, which is significantly faster. To achieve this, we enclose the polynomial zonotopes that represent the reachable set with zonotopes using [32, Prop. 5]. Since zonotopes are linear in the factors α , the feasible region for α calculated using Theorem 1 is then given as a union of polytopes $\bigcup_l \langle A_l, b_l \rangle_P$ instead of a union of polynomial level sets. Consequently, if we additionally minimize the L^1 -norm instead of the L^2 -norm, we can simplify the optimization problem (4) to

$$\min_{\alpha \in [-1, 1]} \|\alpha - \alpha_a\|_1 \text{ s.t. } \alpha \in \bigcap_{r=1}^v \bigcup_{l=1}^{s_r} \langle A_{rl}, b_{rl} \rangle_P,$$

which can be formulated as a mixed-integer linear program using Balas' Theorem [42]:

$$\min_{\alpha \in [-1, 1]} \|\alpha - \alpha_a\|_1 \quad (9)$$

subject to

$$A_{rl} \widehat{\alpha}_{rl} \leq \lambda_{rl} b_{rl}, \quad -\mathbf{1} \lambda_{rl} \leq \widehat{\alpha}_{rl} \leq \mathbf{1} \lambda_{rl},$$

$$\lambda_{rl} \in \{0, 1\}, \quad \alpha = \sum_{l=1}^{s_r} \widehat{\alpha}_{rl}, \quad \sum_{l=1}^{s_r} \lambda_{rl} = 1,$$

for $r = 1, \dots, v$ and $l = 1, \dots, s_r$. The structure of this optimization problem is very similar to (4), except that we introduced the new variables $\widehat{\alpha}_{rl} = \alpha_{rl} \lambda_{rl}$ to avoid the bilinear terms and obtain a linear program. Due to the over-approximation of all nonlinear terms of the polynomial zonotope by the zonotope enclosure, it holds that every feasible solution for (9) is a feasible solution to the original problem (4), but some values that are feasible for (4) will not be feasible for (9). Note that if the system dynamics (1) is linear, we directly obtain a mixed-integer linear program in the form of (9). Moreover, we can always first check if the desired value α_a satisfies the original nonlinear constraints and only perform the simplification to a mixed-integer linear program if it does not. A mixed-integer quadratic program can be obtained in a similar way as the mixed-integer linear program by enclosing all generators that belong to higher-order polynomials by a zonotope. Finally, since mixed-integer programming can be highly parallelized, the computation time for optimization can always be reduced by using a more powerful machine with more cores.

D. CONSTRAINT GROUPING

Since the time step size for reachability analysis is usually relatively small, it often happens that many reachable sets for consecutive time intervals intersect the same obstacle, resulting in a lot of very similar constraints. We can reduce the computation time by grouping similar constraints together, as we demonstrate with the following example:

Example 5: The two constraints

$$\begin{aligned} 1.1 \alpha_1 + 0.7 \alpha_1 \alpha_2 &\leq 0.3 \\ 1.3 \alpha_1 + 0.5 \alpha_1 \alpha_2 &\leq 0.3 \end{aligned}$$

on $\alpha_1, \alpha_2 \in [-1, 1]$ can be grouped to the single constraint

$$\forall \epsilon_1 \in [1.1, 1.3], \forall \epsilon_2 \in [0.5, 0.7] : \epsilon_1 \alpha_1 + \epsilon_2 \alpha_1 \alpha_2 \leq 0.3.$$

To eliminate the new variables ϵ_1 and ϵ_2 we represent their domains as a summation of the center with a zero-centered uncertainty as $\epsilon_1 \in 1.2 + \tilde{\epsilon}_1$, $\epsilon_2 \in 0.6 + \tilde{\epsilon}_2$ with $\tilde{\epsilon}_1, \tilde{\epsilon}_2 \in [-0.1, 0.1]$, which finally yields

$$1.2 \alpha_1 + 0.6 \alpha_1 \alpha_2 \leq \min_{\substack{\alpha_1, \alpha_2 \in [-1, 1] \\ \tilde{\epsilon}_1, \tilde{\epsilon}_2 \in [-0.1, 0.1]}} 0.3 - \epsilon_1 \alpha_1 - \epsilon_2 \alpha_1 \alpha_2,$$

where a lower bound for the optimal value of the minimization problem can be computed using interval arithmetic [43].

In addition to the number of constraints, constraints grouping also decreases the number of integer variables for the optimization in (4), which reduces computation time. Since integer variables are required only if the safe region for the agent is non-convex, another strategy to accelerate the optimization is to replace non-convex safe regions by the largest convex subset [44].

TABLE 1. States n , control inputs m , planning horizon t_f , number of pre-computed reachable sets, and extensions applied for each benchmark.

Benchmark	n	m	t_f	Sets	Extensions
F1tenth car	5	2	2 s	5	V-A.a, V-B, V-C, V-E
Auto. driving	4	2	0.8 s	60	V-A.c, V-B, V-C, V-D, V-E
Quadrotor 2D	6	2	0.5 s	256	V-E
Quadrotor 3D	10	3	3 s	1	V-B, V-C, V-E

E. REACHABLE SET PRE-COMPUTATION

In order to reduce the computation time for our safety shield, we can pre-compute the reachable set starting from an initial set \mathcal{X}_0 offline, and then apply the reachable subset approach [31] to efficiently extract the reachable set for the current state $x_0 \in \mathcal{X}_0$ during online execution. Since for nonlinear systems the accuracy of the reachable set enclosure depends on the size of the initial set, we cannot make \mathcal{X}_0 too large but instead have to divide the relevant state space into sets of suitable size. The number of required sets for such a division grows exponentially with the system dimension, so that this approach is not suited for high-dimensional systems. However, for many systems the differential equation $\dot{x}(t) = f(x(t), u(t), w(t))$ describing the system dynamics is invariant with respect to transformations of certain states [45, Sec. 4.1]. For example, the dynamics of a car are invariant with respect to translations of the car’s position and with respect to rotations of the car’s orientation. In this case only the state space for the states that are not invariant has to be divided since we can always apply a suitable state space transformation to set the invariant states to 0.

VI. EXPERIMENTAL EVALUATION

We now demonstrate the performance of our safety shield on several benchmark systems, where each benchmark highlights different properties of our approach. If not explicitly stated otherwise, all computations are carried out in Python on a 2.9 GHz quad-core i7 processor with 32 GB memory. We use the CORA toolbox [46] to pre-compute reachable sets, proximal policy optimization [47] for reinforcement learning, Gurobi to solve the mixed-integer linear and quadratic programs, and CasADi together with the BONMIN solver to solve mixed-integer nonlinear programs.² Benchmark parameters as well as the applied extensions from Section V are listed in Table 1. We published our implementation on CodeOcean³ and created a video showing our results.⁴

A. FITENTH RACECAR

To demonstrate that our safety shield is fast and robust enough to be applied to a real system, we conduct experiments on an F1tenth racecar [48], whose dynamics are described by

a kinematic single-track model. Moreover, the car contains a low-level PI controller with gains $k_P = 8$ and $k_I = 1$ that takes as input the desired velocity and realizes the required acceleration. Overall, this results in the model

$$\begin{aligned} \dot{s}_x &= \cos(\psi) v, \quad \dot{s}_y = \sin(\psi) v, \quad \dot{\psi} = u_2 + w_2, \\ \dot{v} &= k_P(u_1 - v) + k_I e_I + w_1, \quad \dot{e}_I = u_1 - v, \end{aligned} \quad (10)$$

where the system state consists of the x - and y -position of the center s_x, s_y , the velocity v , the orientation ψ , and the integrated error of the PI controller e_I . The control inputs are the desired velocity u_1 and the steering angle u_2 , which are bounded by the set $\mathcal{U} = [0, 0.5] \text{m s}^{-1} \times [-0.3, 0.3] \text{rad}$. To ensure that the model (10) encloses all possible behaviors of the real system, we performed conformance checking using the AROC toolbox [49] to determine the process noise as well as the measurement error from data traces we recorded from the real car, which results in the sets

$$\begin{aligned} \mathcal{W} &= [-0.007, 0.0035] \text{m s}^{-2} \times [-0.0104, 0.0132] \text{rad s}^{-1} \\ \mathcal{V} &= [-0.0584, 0.0446] \text{m s}^{-1} \times [-0.0438, 0.0466] \text{m s}^{-1} \\ &\quad \times [-0.0933, 0.0561] \text{m s}^{-2} \times [-0.0446, 0.0593] \text{rad s}^{-1} \\ &\quad \times [-0.0005, 0] \text{m s}^{-2}. \end{aligned}$$

To incorporate the size of the car, we use the output function in (6) with length 0.51 m and width 0.31 m.

For control we use a piecewise constant control law with $\mu = 2$ segments and a planning horizon of $t_f = 2$ s, and we replan as soon as the previous computation is finished. Moreover, we simplify the optimization problem for action projection to a mixed-integer quadratic program, which on average took 0.14 s to solve during our experiments. The car uses a 1.9 GHz six-core ARMv8 processor with 7.6 GB memory and is equipped with a LiDAR sensor. To obtain the unsafe sets \mathcal{F}_i , we enclose all points measured by the LiDAR by a union of polytopes. Moreover, while the velocity and the integrated error can be directly obtained from the car’s internal sensors, we use a particle filter [50] to determine the position and orientation of the car in the environment from LiDAR measurements. For our experiments, we then applied reinforcement learning to train an agent on four environment maps that were similar to but slightly different from the map we used for the experiments on the real F1tenth car. In addition to the system state, we used the LiDAR measurements and the position of the goal set as observations for the agent, and we did not use the safety shield during training.

As shown in Fig. 5, without the safety shield, the trained agent is unsafe since the car crashes into the obstacle. With our safety shield, however, the car avoids the obstacle and safely reaches the goal set. This not only demonstrates that our safety shield successfully works on a real system, but also that the modifications to the control inputs suggested by the reinforcement learning policy are small enough for the agent to still fulfill its objective.

²[Online]. Available: <https://www.gurobi.com/> and <https://web.casadi.org/>

³[Online]. Available: <https://codeocean.com/capsule/9949621/tree/v1>

⁴[Online]. Available: <https://youtu.be/6ISKXO4DDWA>

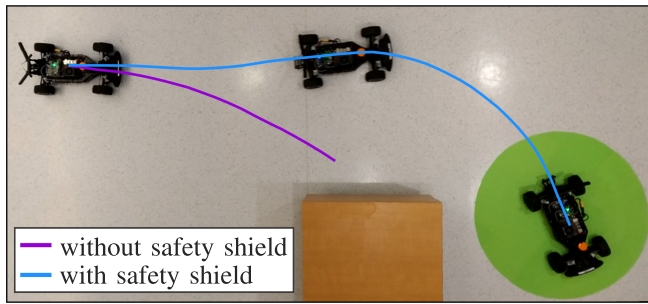


FIGURE 5. Trajectories driven by the F1 tenth racecar with and without the safety shield, where the green area is the goal set and the orange area is the obstacle.

TABLE 2. Results for the evaluation of our safety shield on 2000 common road traffic scenarios.

Agent	Collisions	Goal Reached	Time
without safety shield	10 (0.5%)	1907 (95.35%)	-
with safety shield	0 (0%)	1925 (96.25%)	0.043 s
constraint grouping	0 (0%)	1924 (96.20%)	0.035 s

B. AUTONOMOUS DRIVING

In order to show that our safety shield can handle very complex reach-avoid problems that include dynamic obstacles, we consider the motion planning benchmarks for autonomous cars provided by the CommonRoad database [51]. As system dynamics we use the kinematic single track model from [20, Sec. VII] with the same input set \mathcal{U} and set of process noise \mathcal{W} as in [20, Sec. VII]. This model is very similar to the model in (10), with the only difference that the acceleration instead of the desired velocity is used as a control input. The car we consider is a BMW 320i, which has a length of 4.51 m and a width of 1.61 m. To guarantee safety even though the intentions of the other cars are unclear, we use the tool SPOT [52] to compute all possible occupancies of the other traffic participants that apply to traffic rules using set-based prediction.

To counteract the large process noise for this benchmark, we use a feedback controller $u(t) = u_{ref} + K(x(t) - x_{ref}(t))$ for the safety shield, where the reference input u_{ref} is piecewise constant with $\mu = 2$ segments. The feedback matrix $K \in \mathbb{R}^{m \times n}$ is determined by applying an LQR control approach with state weighting matrix $Q = I_4$ and input weighting matrix $R = I_2$ to the linearized system. Moreover, we use a planning horizon of $t_f = 0.8$ s and replan after $t_c = 0.4$ s. We apply reinforcement learning to train an agent that aims to safely control the car, where we do not use the safety shield during training. The observations for the agent are selected from [53, Table II]. In particular, we use the state of the ego vehicle, the distances of the ego vehicle to road/lane boundaries as well as to the goal set, and the states of surrounding vehicles.

The effect of the safety shield is highlighted by the results for 2000 traffic scenarios shown in Table 2: While the original

agent collides with other traffic participants in 10 scenarios, our safety shield successfully prevents all collisions. Moreover, applying the safety shield does not lead to a reduced goal-reaching percentage, but instead even increases the number of scenarios for which the goal set is reached. Table 2 also demonstrates the effect of constraint grouping (see Section V-D), which reduces the average computation time for solving the optimization problem, but slightly decreases the goal reaching percentage due to the increased conservatism. In Fig. 6 the results for one specific traffic scenario are visualized. There, the agent without the safety shield changes the lane too early and collides with the adjacent truck, whereas the agent with the safety shield changes the lane just in time and finally reaches the goal set in the end.

C. QUADROTOR 2D

Next, we compare our safety shield with a safe reinforcement learning approach that modifies the optimization criterion. In particular, we incorporate the safety specification as a violation penalty in the reward function. For this, we consider a benchmark problem from the safe-control-gym [54] featuring a trajectory tracking task for a two-dimensional quadrotor. As shown in Fig. 7, the trajectory that should be tracked is partially located inside an unsafe region, so that there exists a conflict between tracking performance and safety constraint satisfaction. The dynamics of the quadrotor are according to [54, Eq. (3)] given as

$$\begin{aligned}\ddot{s}_x &= \sin(\psi)(u_1 + u_2)/m + w_1 \\ \ddot{s}_z &= \cos(\psi)(u_1 + u_2)/m - g + w_2 \\ \ddot{\psi} &= (u_2 - u_1)a/(\sqrt{2}I_{yy}) + w_3,\end{aligned}$$

where $m = 0.027$ kg is the mass, $g = 9.81$ m s⁻² is the gravitational acceleration, $a = 0.0397$ m is distance from each motor pair to the center of mass of the quadrotor, and $I_{yy} = 1.4 \cdot 10^{-5}$ kg/m² is the moment of inertia. The system state consists of the x- and z-positions s_x, s_z as well as the pitch angle ψ of the quadrotor together with the corresponding velocities. To decouple forward thrust and tilting torque, the input set for the control inputs u_1 and u_2 that represent the thrusts generated by the two rotors is restricted to

$$\mathcal{U} = \left\langle \begin{bmatrix} 0.1323 \text{ N} \\ 0.1323 \text{ N} \end{bmatrix}, \begin{bmatrix} 0.0125 \text{ N} & 0.0015 \text{ N} \\ 0.0125 \text{ N} & -0.0015 \text{ N} \end{bmatrix} \right\rangle_Z.$$

The process noise w_1, w_2, w_3 is bounded by the set $\mathcal{W} = 0.01 \cdot [-\mathbf{1}, \mathbf{1}]$.

For the safety shield we use a constant control input with a planning horizon of $t_f = 0.5$ s, where we replan after $t_c = 0.02$ s. To perform action projection, we solve the original nonlinear optimization problem, which takes 0.004 s on average during our experiments. The main reason for the fast computation time is that the safe region for the quadrotor is convex, which results in an optimization problem without any integer variables. We train three different agents: A baseline agent that should track the trajectory and gets no information

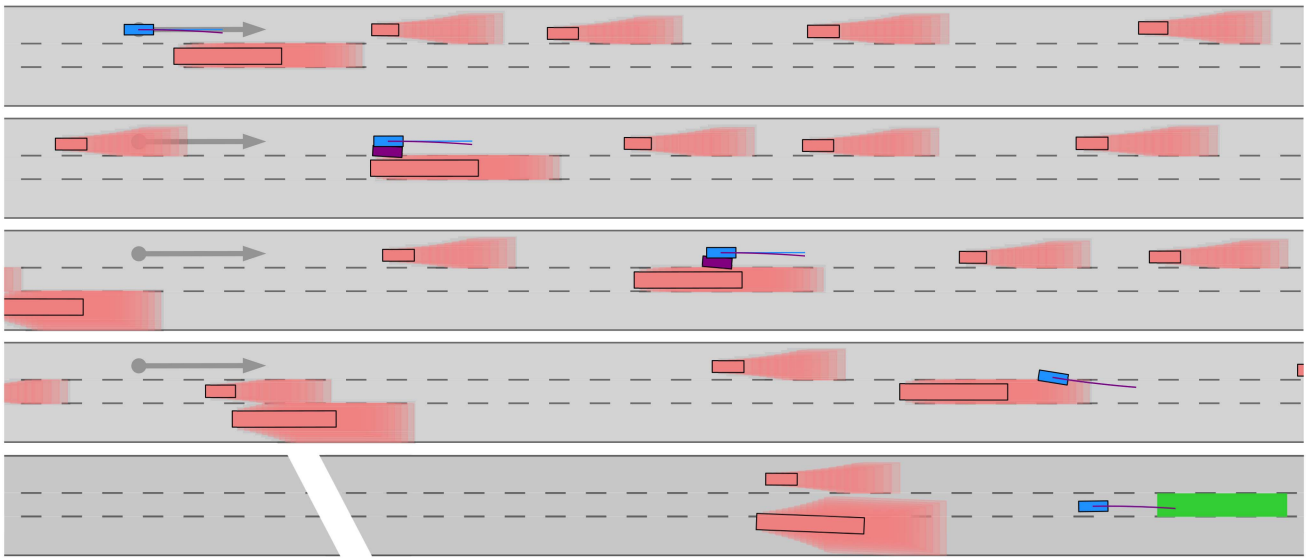


FIGURE 6. Results for the CommonRoad scenario *DEU_LocationALower-33_16_T-1* visualized at times 0 s, 1.2 s, 2.8 s, 4.4 s, and 9.2 s (from top to bottom), where the agent without safety shield is depicted in purple, the agent with safety shield is depicted in blue, the dynamic obstacles are depicted in red, and the goal set is depicted in green.

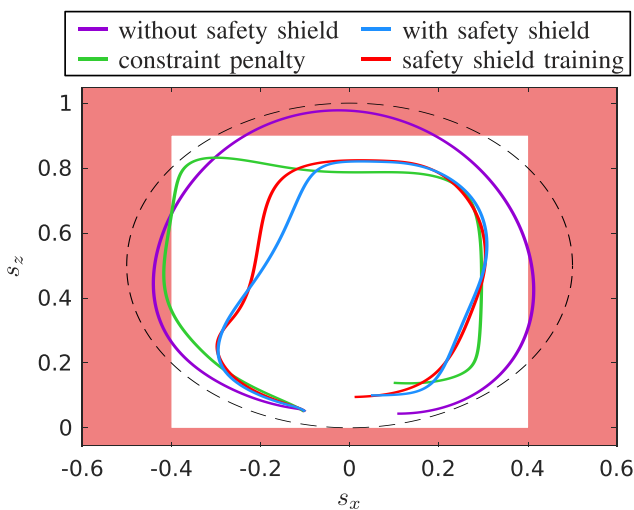


FIGURE 7. Trajectories for the 2D quadrotor benchmark featuring the baseline agent with and without safety shield, the constraint-penalty agent, and the agent that is trained with the safety shield. The trajectory that should be tracked is visualized by the dashed black line and the unsafe regions are depicted in red.

about the constraints, a constraint-penalty agent where the reward is extended with a penalty for constraint violation, and a safe agent that is trained with the safety shield. As shown in Fig. 8, the safe and baseline agents converge after 400 000 training steps while the agent with constraint penalty needs 2 million training steps to converge. Moreover, only the safe agent never violates any constraints during training, and could therefore also be used for training directly on the real physical system.

The results for deploying the different trained agents are shown in Fig. 7. As expected, the baseline agent without the safety shield violates the safety constraints since they were

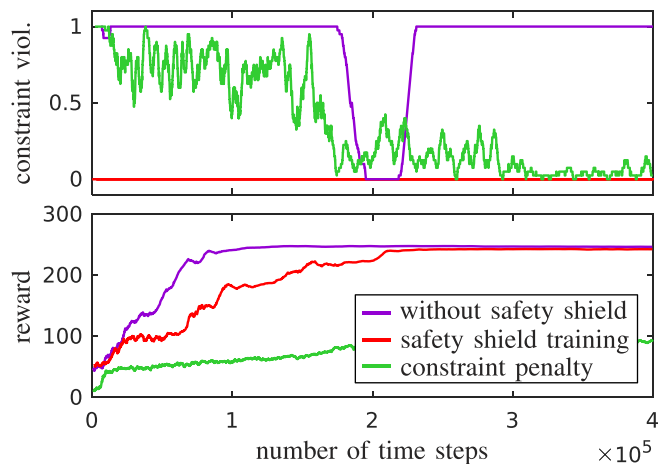


FIGURE 8. Episode rewards and constraint violations for the 2D quadrotor benchmark observed during training without safety shield, with safety shield, and with constraint penalty.

not considered during training. Also, the constraint-penalty agent violates the constraints, which demonstrates that it is not sufficient to incorporate the safety constraints into the training process. Only the two agents that apply our safety shield stay inside the safe region for all times, where the agent that uses the safety shield during training achieves a smoother trajectory compared to the baseline agent.

D. QUADROTOR 3D

To compare our safety shield with reachability-based trajectory safeguard [25], we consider the three-dimensional quadrotor benchmark from [25, Sec. V.B]. Reachability-based trajectory safeguard [25] applies the safety shield to a simplified trajectory-generating model and the resulting trajectory is then tracked by a low-level controller that uses

the original nonlinear system model. For the quadrotor, the trajectory-generating model for each of the three spatial directions $i \in \{1, 2, 3\}$ is

$$\begin{aligned} \dot{x}_i &= v_i + a_i t - (2a_i + 3(v_i - u_i))t^2 + (a_i + 2(v_i - u_i))t^3 \\ v_i &= 0, \quad \dot{a}_i = 0, \quad \dot{t} = 1, \end{aligned}$$

where x_i is the quadrotor position, v_i and a_i are respectively the velocity and the acceleration at the beginning of the trajectory, and t is time. The input u_i to the system is the peak velocity reached at time $t = 1.5$ s, which is bounded by the set $\mathcal{U} = \{u = [u_1 \ u_2 \ u_3]^T \mid \|u\|_2 \leq 5 \text{ m s}^{-1}\}$. To apply our safety shield, we tightly inner-approximate the set \mathcal{U} with a zonotope using the method described in [55, Sec. IV]. A similar trajectory-generating model is used to decelerate the quadrotor from the peak velocity back to velocity 0, so that the overall planning horizon is $t_f = 3$ s. We consider the same control task as in [25, Sec. V.B], which is to safely navigate the quadrotor through a 100 m long tunnel with randomly generated box obstacles. For our experiments, we deployed the same trained reinforcement learning agent as used in [25] on 100 tunnels with different obstacles and compared the conservatism of the two safety shields in terms of the required control input correction $\|u - u_a\|_2$ at each intervention of the safety shield. While both safety shields had to intervene for 5078 out of 5760 time steps, the average control input modification for our approach is with 1.13 m s^{-1} smaller than the average modification 1.22 m s^{-1} for the safety shield from [25], which increases the chances that the agent can successfully complete its task.

VII. DISCUSSION

Finally, let us discuss some properties of our safety shield.

A. SAFETY GUARANTEES FOR INFINITE TIME

Our basic safety shield approach can guarantee safety only for the finite time horizon t_f . To obtain safety guarantees for an infinite time horizon, one can either combine our safety shield with a fail-safe planner [56] that takes over when the safety shield cannot determine a safe trajectory anymore, or one can modify the safety shield in such a way that the system always stops in a safe final state at the end of the planning horizon [25].

B. COMPUTATIONAL COMPLEXITY

The two main steps required for our safety shield are computing the reachable set and solving the mixed-integer optimization problem (4) for action projection. The complexity of the conservative polynomialization algorithm for reachability analysis is $\mathcal{O}(n^5)$ with respect to the system dimension according to [57, Sec. 4.1.4]. However, for many benchmarks one can apply the pre-computation discussed in Section V-E to avoid computing reachable sets online. Solving a mixed-integer optimization problem is in general NP-hard [58]. But, as we demonstrated with the numerical experiments in Section VI, by applying the simplification to a mixed-integer

linear program in Section V-C and/or constraint grouping in Section V-D we can solve this optimization efficiently.

C. SAFE COMPUTATION TIME CONSIDERATION

As demonstrated by the experiments in Section VI, even with all the speed-ups discussed in Section V, the calculations required for our safety shield still need a certain amount of computation time that, depending on the system, might be too long to simply be neglected. Therefore, in order to consider the required computation time in a formally correct manner, we can apply the following well-known procedure [59]: We allocate a certain computation time t_{comp} for the calculations and use reachability analysis to predict the reachable states for the allocated computation time. By using this set as the initial set for our safety shield, we can guarantee safety even though the required calculations are not instantaneous. If the computation does not finish in the allocated computation time, we either stick to the safe solution from the previous time step or apply a failsafe maneuver.

D. CONSERVATISM OF THE SAFETY SHIELD

Due to over-approximation errors, our safety shield might not be able to always find a feasible solution if one exists. In particular, there are four sources of conservatism:

- Since the exact reachable set cannot be computed for general nonlinear systems, we compute a tight enclosure instead (e.g., we aim to minimize the Hausdorff distance between the enclosure and the exact set).
- Due to dependency preservation, the abstraction error for reachability analysis is computed on the reachable set for the whole input set rather than the smaller reachable set for a specific control input, which results in additional conservatism.
- For bloating the obstacles by the set of uncertainties defined by the independent generators, we use an over-approximative Minkowski sum in Theorem 1 that simply pushes the obstacle halfspaces outward.
- Since we choose a certain type of control law in advance, we restrict the space of possible control inputs.

However, all of these over-approximation errors can be made arbitrarily small: The over-approximation for reachability converges to zero if the time step size is reduced and/or the reachable set is split, which also eliminates the error introduced by dependency preservation. Moreover, the approximative Minkowski sum in Theorem 1 can be replaced by the exact one and every control law can be approximated arbitrary close by a piecewise constant control law with an infinite number of piecewise constant segments.

E. PARAMETER TUNING

Since the settings for reachability analysis can be tuned automatically [60], [61], the main design parameters for our safety shield in addition to the type of control law discussed in Section V-A are the planning horizon t_f and the replanning time t_c . A longer planning horizon t_f often yields better control performance due to the larger lookahead, but also increases

the computation time. Especially in the presence of dynamic obstacles, a small replanning time t_c is desirable in order to be able to quickly react to a changing environment. However, a small t_c requires the approach to be faster in order to run in real-time. Finally, the extensions discussed in Section V-C, V-D, and V-E all reduce the computation time at the cost of introducing more conservatism.

VIII. CONCLUSION

We presented a novel safety shield for nonlinear continuous systems with input constraints that can be added to reinforcement learning agents in order to prevent them from applying unsafe actions. Since our safety shield uses set-based computations in the form of reachability analysis to determine which actions are safe and which are unsafe, it can guarantee robust safety despite process noise and measurement errors. Moreover, because our approach applies highly parallelized mixed-integer programming to project the action from the agent to the closest safe action, it is possible to reduce the computation time by using a more powerful machine with more cores. Finally, we demonstrated with several numerical examples as well as experiments on a real system that our safety shield modifies the actions proposed by the reinforcement learning agent as little as necessary for robust safety.

ACKNOWLEDGMENT

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the United States Air Force or the United States Navy.

REFERENCES

- [1] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: A survey," in *Proc. Symp. Ser. Comput. Intell.*, 2020, pp. 737–744.
- [2] B. R. Kiran et al., "Deep reinforcement learning for autonomous driving: A survey," *Trans. Intell. Transp. Syst.*, vol. 23, no. 6, pp. 4909–4926, Jun. 2022.
- [3] Z. Zhang, D. Zhang, and R. C. Qiu, "Deep reinforcement learning for power system applications: An overview," *CSEE J. Power Energy Syst.*, vol. 6, no. 1, pp. 213–225, 2019.
- [4] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 22–31.
- [5] T.-Y. Yang, J. Rosca, K. Narasimhan, and P. J. Ramadge, "Projection-based constrained policy optimization," in *Proc. Int. Conf. Learn. Representations*, 2019.
- [6] M. Hasanbeig, A. Abate, and D. Kroening, "Cautious reinforcement learning with logical constraints," in *Proc. Int. Conf. Auton. Agents Multiagent Syst.*, 2020, pp. 483–491.
- [7] X. Wang, C. Pillmayer, and M. Althoff, "Learning to obey traffic rules using constrained policy optimization," in *Proc. Int. Conf. Intell. Transp. Syst.*, 2022, pp. 2415–2421.
- [8] B. Könighofer, J. Rudolf, A. Palmisano, M. Tappler, and R. Bloem, "Online shielding for stochastic systems," in *Proc. NASA Formal Methods Symp.*, 2021, pp. 231–248.
- [9] P. Geibel and F. Wysotzki, "Risk-sensitive reinforcement learning applied to control under constraints," *J. Artif. Intell. Res.*, vol. 24, pp. 81–108, 2005.
- [10] H. Krasowski, J. Thumm, M. Müller, X. Wang, and M. Althoff, "Provably safe reinforcement learning: A theoretical and experimental comparison," 2022, *arXiv:2205.06750*.
- [11] H. Krasowski, X. Wang, and M. Althoff, "Safe reinforcement learning for autonomous lane changing using set-based prediction," in *Proc. Int. Conf. Intell. Transp. Syst.*, 2020, pp. 1–7.
- [12] D. Isele, A. Nakhaei, and K. Fujimura, "Safe reinforcement learning on autonomous vehicles," in *Proc. Int. Conf. Intell. Robots Syst.*, 2018, pp. 6162–6167.
- [13] S. Huang and S. Ontañón, "A closer look at invalid action masking in policy gradient algorithms," in *Proc. Int. FLAIRS Conf.*, 2022.
- [14] J. Thumm and M. Althoff, "Provably safe deep reinforcement learning for robotic manipulation in human environments," in *Proc. Int. Conf. Robot. Automat.*, 2022, pp. 6344–6350.
- [15] W. Saunders, G. Sastry, A. Stuhlmüller, and O. Evans, "Trial without error: Towards safe reinforcement learning via human intervention," in *Proc. Int. Conf. Auton. Agents MultiAgent Syst.*, 2018, pp. 2067–2069.
- [16] N. Hunt, N. Fulton, S. Magliacane, T. N. Hoang, S. Das, and A. Solar-Lezama, "Verifiably safe exploration for end-to-end reinforcement learning," in *Proc. 24th Int. Conf. Hybrid Systems: Comput. Control*, 2021, pp. 1–11.
- [17] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, "Safe reinforcement learning via shielding," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 2669–2678.
- [18] D. Seto, B. Krogh, L. Sha, and A. Chutinan, "The Simplex architecture for safe online control system upgrades," in *Proc. Amer. Control Conf.*, 1998, pp. 3504–3508.
- [19] D. T. Phan, R. Grosu, N. Jansen, N. Paoletti, S. A. Smolka, and S. D. Stoller, "Neural simplex architecture," in *Proc. NASA Formal Methods Symp.*, 2020, pp. 97–114.
- [20] B. Schürmann, M. Klischat, N. Kochdumper, and M. Althoff, "Formal safety net control using backward reachability analysis," *IEEE Trans. Autom. Control*, vol. 67, no. 11, pp. 5698–5713, Nov. 2022.
- [21] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 3387–3395.
- [22] Z. Marvi and B. Kiumarsi, "Safe reinforcement learning: A control barrier function optimization approach," *Int. J. Robust Nonlinear Control*, vol. 31, no. 6, pp. 1923–1940, 2021.
- [23] O. Bastani, "Safe reinforcement learning with nonlinear dynamics via model predictive shielding," in *Proc. Amer. Control Conf.*, 2021, pp. 3488–3494.
- [24] K. P. Wabersich and M. N. Zeilinger, "A predictive safety filter for learning-based control of constrained nonlinear dynamical systems," *Automatica*, vol. 129, 2021, Art. no. 109597.
- [25] Y. S. Shao, C. Chen, S. Kousik, and R. Vasudevan, "Reachability-based trajectory safeguard (RTS): A safe and fast reinforcement learning safety layer for continuous control," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 3663–3670, Apr. 2021.
- [26] J. H. Gillula and C. J. Tomlin, "Guaranteed safe online learning via reachability: Tracking a ground target using a quadrotor," in *Proc. Int. Conf. Robot. Automat.*, 2012, pp. 2723–2730.
- [27] I. M. Mitchell, A. M. Bayen, and C. J. Tomlin, "A time-dependent HamiltonJacobi formulation of reachable sets for continuous dynamic games," *IEEE Trans. Autom. Control*, vol. 50, no. 7, pp. 947–957, Jul. 2005.
- [28] M. Selim, A. Alanwar, S. Kousik, G. Gao, M. Pavone, and K. H. Johansson, "Safe reinforcement learning using black-box reachability analysis," *IEEE Robot. Automat. Lett.*, vol. 7, no. 4, pp. 10665–10672, Oct. 2022.
- [29] J. K. Scott, D. M. Raimondo, G. R. Marseglia, and R. D. Braatz, "Constrained zonotopes: A new tool for set-based estimation and fault detection," *Automatica*, vol. 69, pp. 126–136, 2016.
- [30] M. Althoff, "Reachability analysis of nonlinear systems using conservative polynomialization and non-convex sets," in *Proc. Int. Conf. Hybrid Syst.: Comput. Control*, 2013, pp. 173–182.
- [31] N. Kochdumper, B. Schürmann, and M. Althoff, "Utilizing dependencies to obtain subsets of reachable sets," in *Proc. Int. Conf. Hybrid Syst.: Comput. Control*, 2020, article 1.
- [32] N. Kochdumper and M. Althoff, "Sparse polynomial zonotopes: A novel set representation for reachability analysis," *IEEE Trans. Autom. Control*, vol. 66, no. 9, pp. 4043–4058, Sep. 2021.
- [33] M. Althoff, O. Stursberg, and M. Buss, "Reachability analysis of nonlinear systems with uncertain parameters using conservative linearization," in *Proc. Int. Conf. Decis. Control*, 2008, pp. 4042–4048.

- [34] H. Roehm, J. Oehlerking, M. Woehrle, and M. Althoff, "Model conformance for cyber-physical systems: A survey," *Trans. Cyber-Phys. Syst.*, vol. 3, no. 3, 2018, Art. no. 30.
- [35] M. Koschi and M. Althoff, "Set-based prediction of traffic participants considering occlusions and traffic rules," *IEEE Trans. Intell. Veh.*, vol. 6, no. 2, pp. 249–265, Jun. 2021.
- [36] S. Bak, S. Bogomolov, B. Hency, N. Kochdumper, E. Lew, and K. Potomkin, "Reachability of Koopman linearized systems using random fourier feature observables and polynomial zonotope refinement," in *Proc. Int. Conf. Comput. Aided Verification*, 2022, pp. 490–510.
- [37] I. E. Grossmann and S. Lee, "Generalized convex disjunctive programming: Nonlinear convex hull relaxation," *Comput. Optim. Appl.*, vol. 26, no. 1, pp. 83–100, 2003.
- [38] X. Chen, S. Sankaranarayanan, and E. Ábrahám, "Taylor model flow-pipe construction for non-linear hybrid systems," in *Proc. Real-Time Syst. Symp.*, 2012, pp. 183–192.
- [39] N. Kochdumper and M. Althoff, "Reachability analysis for hybrid systems with nonlinear guard sets," in *Proc. 23rd Int. Conf. Hybrid Syst.: Comput. Control*, 2020, pp. 1–10.
- [40] K. Makino and M. Berz, "Taylor models and other validated functional inclusion methods," *Int. J. Pure Appl. Math.*, vol. 4, no. 4, pp. 379–456, 2003.
- [41] G. Dantzig, *Linear Programming and Extensions*. Princeton, NJ, USA: Princeton Univ. Press, 2016.
- [42] E. Balas, "Disjunctive programming: Properties of the convex hull of feasible points," *Discrete Appl. Math.*, vol. 89, no. 1, pp. 3–44, 1998.
- [43] L. Jaulin, M. Kieffer, and O. Didrit, *Applied Interval Analysis*. Berlin/Heidelberg, Germany: Springer, 2006.
- [44] R. Deits and R. Tedrake, "Computing large convex regions of obstacle-free space through semidefinite programming," in *Proc. Int. Workshop Algorithmic Found. Robot.*, 2015, pp. 109–124.
- [45] S. Bak, Z. Huang, F. A. T. Abad, and M. Caccamo, "Safety and progress for distributed cyber-physical systems with unreliable communication," *Trans. Embedded Comput. Syst.*, vol. 14, no. 4, pp. 1–22, 2015.
- [46] M. Althoff, "An introduction to CORA 2015," in *Proc. Int. Workshop Appl. Verification Continuous Hybrid Syst.*, 2015, pp. 120–151.
- [47] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, [arXiv:1707.06347](https://arxiv.org/abs/1707.06347).
- [48] M. O'Kelly, H. Zheng, D. Karthik, and R. Mangharam, "F1tenth: An open-source evaluation environment for continuous control and reinforcement learning," *Proc. Mach. Learn. Res.*, vol. 123, pp. 77–89, 2020.
- [49] N. Kochdumper, F. Gruber, B. Schürmann, V. Gaßmann, M. Klischat, and M. Althoff, "AROC: A toolbox for automated reachset optimal controller synthesis," in *Proc. 24th Int. Conf. Hybrid Syst.: Computation Control*, 2021, pp. 1–6.
- [50] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust Monte Carlo localization for mobile robots," *Artif. Intell.*, vol. 128, no. 1-2, pp. 99–141, 2001.
- [51] M. Althoff, M. Koschi, and S. Manzing, "CommonRoad: Composable benchmarks for motion planning on roads," in *Proc. IEEE Intell. Veh. Symp.*, 2017, pp. 719–726.
- [52] M. Koschi and M. Althoff, "SPOT: A tool for set-based prediction of traffic participants," in *Proc. Intell. Veh. Symp.*, 2017, pp. 1686–1693.
- [53] X. Wang, H. Krasowski, and M. Althoff, "CommonRoad-RL: A configurable reinforcement learning environment for motion planning of autonomous vehicles," in *Proc. Int. Intell. Transp. Syst. Conf.*, 2021, pp. 466–472.
- [54] Z. Yuan et al., "Safe-control-Gym: A unified benchmark suite for safe learning-based control and reinforcement learning in robotics," *IEEE Robot. Automat. Lett.*, vol. 7, no. 4, pp. 11142–11149, Oct. 2022.
- [55] V. Gaßmann and M. Althoff, "Scalable zonotope-ellipsoid conversions using the Euclidean zonotope norm," in *Proc. Amer. Control Conf.*, 2020, pp. 4715–4721.
- [56] C. Pek and M. Althoff, "Computationally efficient fail-safe trajectory planning for self-driving vehicles using convex optimization," in *Proc. Int. Conf. Intell. Transp. Syst.*, 2018, pp. 1447–1454.
- [57] N. Kochdumper, "Extensions of polynomial zonotopes and their application to verification of cyber-physical systems," Ph.D. dissertation, Tech. Univ. Munich, München, Germany, 2022.
- [58] C. H. Papadimitriou, "On the complexity of integer programming," *J. ACM*, vol. 28, no. 4, pp. 765–768, 1981.
- [59] B. Schürmann, N. Kochdumper, and M. Althoff, "Reachset model predictive control for disturbed nonlinear systems," in *Proc. Int. Conf. Decis. Control*, 2018, pp. 3463–3470.

- [60] M. Wetzlinger, A. Kulmburg, and M. Althoff, "Adaptive parameter tuning for reachability analysis of nonlinear systems," in *Proc. 24th Int. Conf. Hybrid Syst.: Comput. Control*, 2021, pp. 1–11.
- [61] M. Wetzlinger, N. Kochdumper, S. Bak, and M. Althoff, "Fully-automated verification of linear systems using inner-and outer-approximations of reachable sets," 2022, [arXiv:2209.09321](https://arxiv.org/abs/2209.09321).



NIKLAS KOCHDUMPER received the B.Sc. degree in mechanical engineering, the M.Sc. degree in robotics, cognition and intelligence, and the Ph.D. degree in computer science from Technische Universität München, Germany, in 2015, 2017, and 2022, respectively. He is currently a Post-doctoral Researcher with Stony Brook University, Stony Brook, NY, USA. His research interests include formal verification of continuous and hybrid systems, reachability analysis, computational geometry, controller synthesis, and neural network verification.



HANNA KRASOWSKI received the B.Sc. degree in mechanical engineering from Technische Universität Darmstadt, Darmstadt, Germany, in 2017, and the M.Sc. degree in robotics, cognition and intelligence in 2020 from Technische Universität München, Munich, Germany, where she is currently working toward the Ph.D. degree. Her research interests include provably safe reinforcement learning and motion planning for cyber-physical systems.



XIAO WANG received the B.Eng. degree in vehicle engineering from Tongji University, Shanghai, China, in 2015, and the M.Sc. degree in mechanical engineering in 2018 from Technische Universität München, München, Germany, where she is currently working toward the Ph.D. degree. Her research interests include motion planning for autonomous vehicles, formal methods, and safe reinforcement learning.



STANLEY BAK received the B.Sc. degree in computer science from Rensselaer Polytechnic Institute, Troy, NY, USA, in 2007, and the M.Sc. and Ph.D. degrees in computer science from the University of Illinois at Urbana-Champaign, Champaign, IL, USA, in 2009 and 2013, respectively. He is currently an Assistant Professor in computer science with Stony Brook University in Stony Brook, NY, USA. His research interests include verification and testing methods for cyber-physical systems and neural networks.



MATTHIAS ALTHOFF received the Diploma Engineering degree in mechanical engineering and the Ph.D. degree in electrical engineering from Technische Universität München, München, Germany, in 2005 and 2010, respectively. He is currently an Associate Professor in computer science with Technische Universität München. From 2010 to 2012 he was a Postdoctoral Researcher with Carnegie Mellon University, Pittsburgh, PA, USA, and from 2012 to 2013 an Assistant Professor with Technische Universität Ilmenau, Ilmenau,

Germany. His research interests include formal verification of continuous and hybrid systems, reachability analysis, planning algorithms, nonlinear control, automated vehicles, and power systems.