Reachable Sets for Control and Planning: from Reactive Safety to Contact-Rich Manipulation

Simin Liu

August 2025
CMU-RI-TR-25-88

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania
United States of America

Thesis Committee:
Prof. Changliu Liu (Chair)
Prof. John Dolan (Chair)
Prof. Maxim Likhachev
Dr. Tao Pang (RAI)

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Robotics.

Abstract

This thesis investigates two fundamental challenges in robotics: ensuring reactive safety for agile systems and achieving scalable planning for contact-rich manipulation. Despite arising in distinct contexts, both problems hinge on reasoning about system dynamics and constraints. We argue that **reachable sets**—long studied in control theory—provide a unifying computational primitive for tackling both.

In the first part, we study reactive safety for uncertain and high-dimensional systems. We develop new methods for synthesizing control barrier functions (CBFs) using both formal optimization and learning-based techniques. Specifically, we introduce a robust-adaptive synthesis framework based on sum-of-squares programming, which guarantees safety under parametric uncertainty while reducing unnecessary conservatism. To address scalability, we propose an adversarial training framework that learns neural CBFs capable of handling higher-dimensional dynamics. Across cartpole, quadrotor, and other benchmarks, our controllers achieve 100% safety while interfering substantially less with nominal performance objectives than existing baselines.

In the second part, we extend reachable-set methods to long-horizon planning for contact-rich manipulation. We present a hierarchical planner for bimanual SE(2) reorientation tasks that uses *mutual reachable sets* as discrete motion primitives. By covering object space with a compact set of such primitives, we reduce the combinatorial complexity of hybrid contact planning to shortest-path search on a small graph. This yields efficient and expressive global plans that outperform sampling-based methods, with successful demonstrations on real robotic hardware.

Together, these contributions demonstrate how reachable sets can serve as a versatile foundation across the robotics stack: from low-level safety filters for agile systems to high-level planning abstractions for dexterous manipulation. This unifying perspective suggests broader opportunities for integrating reachability-based methods into robust and scalable robotic autonomy.

Dedicated to my parents, Xin and Xuejun

Firstly, to my advisors, John and Changliu. From Changliu, I have learned so much about strategic thinking and have always valued your candid advice, which cuts straight to the heart of the matter. I've benefitted greatly from your strong mathematical intuition in our discussions about algorithms, and your emphasis on rigor is something I will carry with me throughout my career. John, you have been a model of fairness and even-handed leadership. You offered me many opportunities and supported me in pursuing them, providing grounded advice and feedback that served as a necessary sanity check. You have taught me to always keep the bigger picture in mind. I am also grateful to my committee member, Max — despite your busy schedule and renowned stature, you always made time when I asked, offering kindness and clear, measured advice.

I am also grateful to my PIs at the Robotics and AI Institute. Pang, I appreciate your deep technical expertise and the connections you built for me within the MIT research ecosystem. You encouraged me to ask "why" more deeply and to approach experiments with greater thoroughness. Jiuguang, thank you for giving me the opportunity to grow beyond the confines of my lab and for doing everything in your power to support our project. I would also like to thank my broader research community. I am indebted to my talented collaborators — Tong, Kai, Xusheng, Weiye, Bernhard, and Peter — and also to my bright, capable labmates in both ICL and DRIVE. Our intellectual exchanges were stimulating and made me a sharper, more critical thinker. I am also thankful to Qualcomm for sponsoring a year of my PhD, which gave me the freedom to pursue ambitious research questions and encouraged me to consider real-world applications early in my doctoral work.

I would also like to thank my mentors for their guidance and inspiration: Hongkai at TRI, for believing in me, for your generosity, and for being a model of the kind and brilliant scientist we should all aspire to be; Kristen at Skydio, a beacon of technical leadership and emotional intelligence - it gives me hope that people like you are changing the culture of tech; Jiahui at RAI, for the wisest advice wrapped in humor; Lesley at Amazon, for being a charismatic and formidable leader; and Siyao and Xuewei, for believing in me and helping me transform my life from the inside out. Also, I would like to thank Prof. Dave Held for serving on my speaking qualifier committee, for his genuine interest in my research, and for his insightful questions; Prof. Sylvia Herbert for serving on my proposal committee and for being a continual source of inspiration through her seminal thesis work and the invaluable open-source resources she has contributed to the robotics community; and Prof. Andrea Bajcsy for helping me level up my presentation skills—an ability that will serve me well throughout my career.

To my family — my heart, and truly the best family I could ask for — I am endlessly grateful. My parents, my best friends, have been there for listening, pep talks, and cross-generational wisdom, often in late-night calls that helped me put my PhD trials into perspective. My grandparents, uncle Jian, and aunt Chunlin have always believed in me, offering encouragement and wry advice. Thank you for loving me unconditionally and making feel like I could overcome anything with all of you in my corner.

Finally, to my chosen family, thank you for making life richer. Ramya, my chosen sister, you have shown me what true care, compassion, grace, and strength look like — and you make the best food. Alex, you have utterly changed how I think and live, making my life more expansive, fluid,

and open. Thank you also for feeding me. Gokul, the best office mate and friend, I'm lucky we've witnessed each other's growth over the years. Vince, you have encouraged my creative pursuits, which have made me feel whole. Mono, you give the best pep talks and career-savvy advice. Shahul, I admire your unabashed originality and profundity. To Brandon, MaryGrace, and Jon F. — thank you for helping me have a life outside of academia and for our lighthearted weekend adventures. And to my old friends, fellow PhD classmates, housemates, and Cambridge community — Arka, Arpit, Ashwin, Ava, Bhavna, Chao, Chu-er, Dom, Fangzhou, Fiona, Harrison, JJ, Kate, Kaylie, Leo, Prachi, Qiwei, Ram, Ravi, Sae, Sophie, Taeyoon, Thomas, XunYi, Yun, and Zhili — thank you for the warm companionship and care that made these years unforgettable.

Contents

List of Tables

List of Figures

1 Introduction

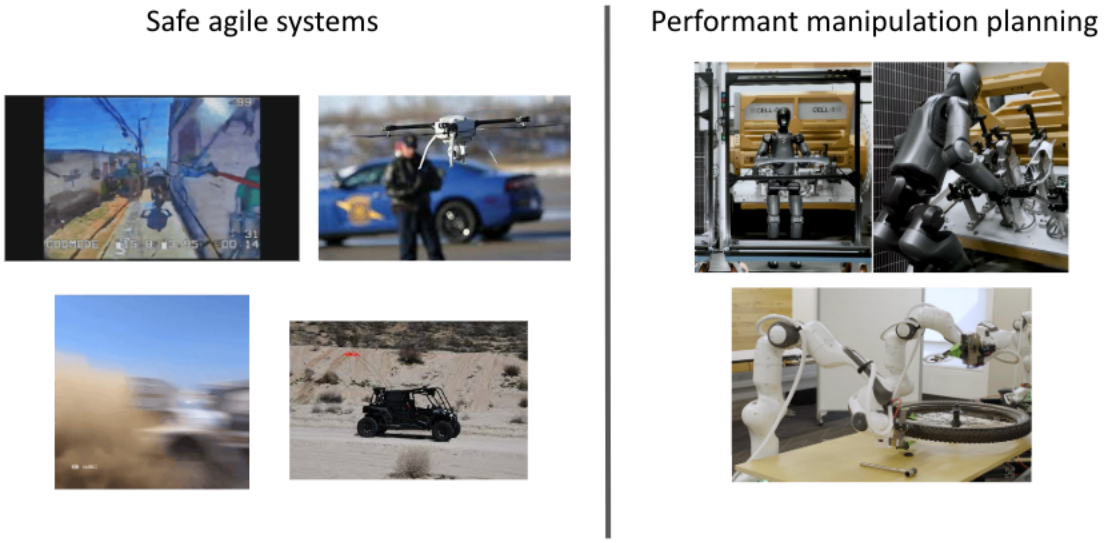


Figure 1.1: This figure illustrates our vision for society. (Left) Drones are already a mature technology, but enhancing collision avoidance at higher speeds would expand their applicability to broader tasks—such as autonomous active tracking (shown here in human-piloted mode). Potential applications include law enforcement patrols, pursuit and emergency escorts, and first-responder support. Similar considerations apply to fast autonomous ground vehicles. (Right) High-quality manipulation could transform manufacturing, logistics, home robotics, and assistive applications. At present, however, most efforts remain in the research phase and are not yet reliable or robust. For example, BMW and Figure's humanoid platform is still in testing and development, while TRI's work is research-oriented rather than a deployable product.

In this thesis, we investigate two fundamental challenges in robotics: ensuring **safety and performance for agile systems** and achieving **scalable, high-quality planning for contact-rich manipulation**. Although these problems arise in very different contexts, they share a common thread: both demand reliable reasoning about how systems evolve under dynamics and constraints.

For agile systems, the central requirement is to operate both safely and efficiently—avoiding collisions while meeting time-sensitive objectives. Safety becomes especially critical in high-performance settings, where failures can have catastrophic consequences. For instance, at the speeds character-

istic of autonomous drone racing, a single crash can end an attempt outright, with no opportunity for recovery. The ability to combine safety with speed is therefore indispensable.

For manipulation, the challenge is different but equally pressing: how to generate high-quality plans for systems of practical complexity, such as bimanual robots performing reorientation tasks. Manipulation of this kind requires reasoning over complex, multi-contact interactions, where existing approaches struggle to balance scalability and solution quality.

The societal benefits of solving these challenges would be immense (Fig. 1.1). For agile systems, advancing collision avoidance to function reliably at higher speeds would unlock new applications, such as autonomous active tracking in cluttered environments. These capabilities would enable law enforcement and emergency services to use drones for patrolling, pursuit, escorts, and first-responder support. Similar benefits apply to fast autonomous ground vehicles. For manipulation, more scalable and higher-performing planning would allow robot arms and humanoids to operate with efficiency approaching that of humans. This would make them far more practical and attractive to deploy, opening up a wide range of applications in manufacturing, logistics, and home robotics.

Yet, despite recent advances, we are not there today. High-speed drone tracking still requires human piloting, as navigating dense environments autonomously at speed remains beyond reach. Similarly, the dexterous robotic skills often highlighted in demonstrations from groups such as TRI or FigureAI remain in research and development, far from being reliable products. Technically, the limitations are clear: for agile systems, many existing approaches are restricted to toy systems or rely on restrictive assumptions, while in manipulation, model-based planners often fail to scale without compromising plan quality.

Surprisingly, both problems can be addressed using a single mathematical tool: **reachable sets**. A second thrust of this thesis, beyond tackling the technical gaps in agile systems and manipulation planning, is to argue that reachable sets are a far more versatile computational primitive than is commonly appreciated. They can unify safety analysis and global planning, providing a foundation for reasoning across both domains.

1.1 Background on Reachability

In this section, we define the different kinds of reachable sets and identify which ones we will be using in this thesis. First, we define the system dynamics. Let $x \in \mathbb{R}^n$ be the system state. Then, we consider it to evolve according to the following ordinary differential equation:

$$\dot{x}(\tau = f(x(\tau), u(\tau), d(\tau)) \tag{1.1}$$

$$\tau \in [t, 0], u(\tau) \in \mathcal{U}, d(\tau) \in \mathcal{D} \tag{1.2}$$

where $u(\tau)$ and $d(\tau)$ denote the control and the disturbance respectively. We assume that these inputs are drawn from compact sets ($\mathcal{U} \subset \mathbb{R}^{n_u}$ and $\mathcal{D} \subset \mathbb{R}^{n_d}$). This assumption typically holds, as the magnitude of control that we can assert on the system is typically bounded. Assuming that the disturbance is bounded is often true in practice, but can become an issue if these bounds are too

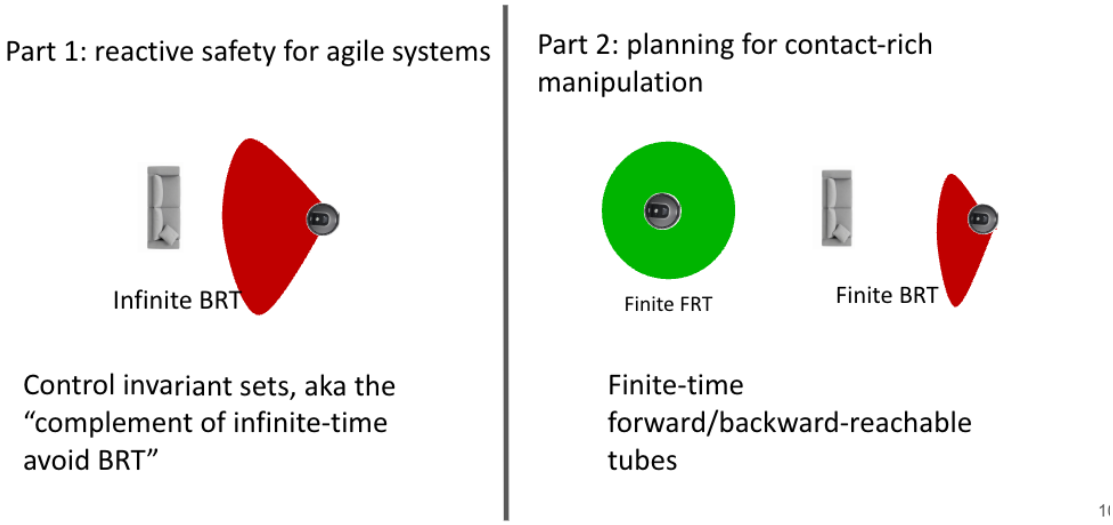


Figure 1.2: This figure illustrates the two kinds of reachable sets we will be studying in this thesis.

big (and therefore allow the disturbance to have a huge effect on the system) or too small (and are therefore invalid compared to the true disturbance). Bounded disturbance can capture a wide range of phenomena, including unmodeled dynamics, parameter uncertainty (as in Sec. 3), or physical disturbances, like wind.

The system dynamics $f : \mathbb{R}^n \times \mathcal{U} \times \mathcal{D} \to \mathbb{R}^n$ is assumed to be uniformly continuous, and Lipschitz continuous in x, uniformly Lipschitz continuous in $u(\cdot)$ and $d(\cdot)$, such that given $u(\cdot)$ and $d(\cdot)$, there is a unique solution to Eqn. 1.1. Let us denote such solutions, which start from state x at time t under control $u(\cdot), d(\cdot)$ as

$$\zeta(\tau, x, t, u(\cdot), d(\cdot)) : [t, T] \to \mathbb{R}^n \tag{1.3}$$

This is the state at time τ, given initial state x, initial time t, and input functions $u(\cdot)$ and $d(\cdot)$ applied over $[t, \tau]$.

Now, we are ready to define the core terminology in reachability. There are two types of reachability: backwards and forwards. Backwards is further split between avoid problems and reach problems. For all these kinds of reachability problems, there is a set $\mathcal{L}$ that specifies the task - it may be a set of states to avoid, reach, or initialize the problem from.

Backwards-avoid problems. In backwards-avoid problems, we are given $\mathcal{L}$, a set of unsafe states, and want to compute all initial states that are "inevitably unsafe". That is, we want to find the states for which there is **no** valid input sequence that avoids $\mathcal{L}$, under worst-case disturbances (i.e. disturbances driving the system toward $\mathcal{L}$).

$$\mathcal{V}_{avoid-BRS}(t) = \left\{ x : \exists d(\cdot) \in \mathcal{D}, \forall u \in \mathcal{U}, \zeta_{x,t}^{u,d}(T) \in \mathcal{L} \right\} \quad \text{Avoid Backward Reachable Set} \tag{1.4}$$

We also distinguish between reachable *sets* and *tubes*. For backwards-avoid problems, sets contain states that would become unsafe at exactly time T, whereas tubes contain all states that collide up to time T:

$$\mathcal{V}_{avoid-BRT}(t) = \left\{x : \exists d(\cdot) \in \mathcal{D}, \forall u \in \mathcal{U}, \exists \tau \in [t, T], \zeta_{x,t}^{u,d}(T) \in \mathcal{L}\right\} \quad \text{Avoid Backward Reachable Tube} \tag{1.5}$$

Backwards-reach problems. In backwards-reach problems, we are given $\mathcal{L}$, a set of goal states, and want to compute all initial states that can reach the goal set. That is, we want to find the states for which there is **some** valid input sequence that enters $\mathcal{L}$, under worst-case disturbances (i.e. disturbances driving the system away from $\mathcal{L}$).

$$\mathcal{V}_{reach-BRS}(t) = \left\{x : \exists u \in \mathcal{U}, \forall d(\cdot) \in \mathcal{D}, \zeta_{x,t}^{u,d}(T) \in \mathcal{L}\right\} \quad \text{Reach BRS} \tag{1.6}$$

Analogously, the backward reachable tube for the reach problem is:

$$\mathcal{V}_{reach-BRT}(t) = \left\{x : \exists u \in \mathcal{U}, \forall d(\cdot) \in \mathcal{D}, \exists \tau \in [t, T], \zeta_{x,t}^{u,d}(T) \in \mathcal{L}\right\} \quad \text{Reach BRT} \tag{1.7}$$

We now introduce the concept of *finite-time* vs. *infinite-time* problems. Finite-time problems, which we have exclusively considered so far, assume a finite time horizon T. In the case of infinite-time problems, avoid-BRT will tend to converge, whereas reach-BRT will continue to expand indefinitely.

Forward reachability problems. In forward-reach problems, we are given $\mathcal{L}$, the set of initial states. We want to compute all states that are reachable from $\mathcal{L}$ under any possible disturbance.

$$\mathcal{V}_{FRS}(t) = \left\{y : \exists u \in \mathcal{U}, \forall d(\cdot) \in \mathcal{D}, x \in \mathcal{L}, \zeta_{x,t}^{u,d}(T) = y \in \mathcal{L}\right\} \quad \text{Finite-time FRS} \tag{1.8}$$

Similar to the backward reachability case, we can define the concepts of sets vs. tubes, finite-time vs. infinite-time here as well.

We study two kinds of reachable set in this thesis (Fig. 1.2). In Part I where we study safe control, we compute "control invariant sets" which are the complement of the infinite-time avoid BRT. In other words, given an unsafe set $\mathcal{L}$, we want to compute all inevitably unsafe initial states and then exclude them. The remaining states form a set that we can stay inside for all time ("invariant") and from which the system can indefinitely avoid the unsafe states. Knowing the control invariant set, our safe control strategy is merely to keep inside this set. In Part II, we consider long-horizon planning for the hybrid problem of contact-rich manipulation. We want to chain together plans from a local planner over a number of "neighborhoods" in state space. However, we want to do this in a principled way, so we need to explicitly compute these neighborhoods as reachable sets. In this part of the thesis, we construct "finite-time forward/backward reachable tubes" - these encapsulate the

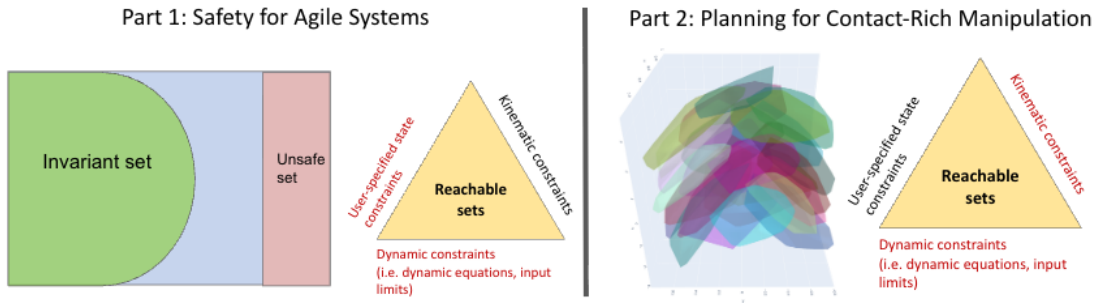


Figure 1.3: This figure illustrates how reachable sets are a unifying concept across the two parts of this thesis.

range in object space of our local planner initialized at a certain seed configuration.

We also argue in this thesis that reachable sets are a highly versatile computational primitive, as they encapsulate multiple notions of feasibility (Fig. 1.3). These include kinematic constraints, which are central in manipulation; dynamic constraints such as equations of motion and input limits, which are critical for agile systems operating near their dynamic limits; and user-specified state constraints, typically representing safety requirements. Because reachable sets naturally integrate all of this information, they can be applied at both ends of the control spectrum—from low-level safety filters to long-horizon planning.

At the low level (Part I), where safety filters must respect dynamic constraints, reachable sets allow us to anticipate when the system is on a trajectory toward violation and intervene early, well before a crash occurs. At the high level (Part II), they provide a useful abstraction between short-horizon local planners and long-horizon reasoning. In this way, the same concept—reachable sets—serves both as a tool for reactive safety enforcement and as a building block for global planning in contact-rich manipulation.

1.2 Related Works

1.2.1 Invariant Set Synthesis and Safe Control Filters

In general, computing reachable sets can be framed as an optimal control problem. This makes them solvable through the Hamilton–Jacobi–Bellman (HJB) framework, a general tool for optimal control [1], or via geometric set propagation techniques [2, 3, 4, 5, 6, 7].

Within safe control, a well-established principle is that if we can compute the system's *invariant set*, we can directly derive a corresponding safety policy. A set is *forward invariant* if, for some admissible control inputs, any trajectory starting inside the set can be kept inside it for all time. For safety applications, we seek an invariant set lying entirely within the complement of the avoid region. Once such a set is found, constructing a controller to keep the system inside it—and thus away from danger—is straightforward. This controller is typically implemented as a *filter* layered on top of a performance-oriented controller (e.g., goal-reaching, trajectory tracking), modifying its outputs only when the system approaches the set boundary.

As mentioned earlier, invariant sets are a subcategory of reachable sets—specifically, they are the *complement of infinite-time avoid-backward reachable tubes*. Consequently, they can be computed using the same general-purpose tools for optimal control and geometric reachability analysis mentioned above. However, these "set-based approaches", which explicitly compute the set itself, scale poorly (typically to $\leq$6D). In HJB, for instance, the value function is computed on a discretized state space via Bellman updates, while geometric set propagation techniques also discretize the state space. In both cases, the curse of dimensionality quickly renders the computation intractable for higher-dimensional systems.

A complementary family of approaches—"function-based methods"—represents invariant sets implicitly as the level sets of functions. The goal is to identify a *valid* function, meaning one whose level set corresponds to a true invariant set, with validity encoded as constraints on the function itself. These methods tend to scale more effectively, since a function representation typically has much lower complexity (e.g., lower Vapnik–Chervonenkis (VC)-dimension) than an explicit representation of an arbitrary set. Control Barrier Functions (CBFs)—the focus of this thesis—belong to this category. CBFs are closely related to energy functions and Lyapunov methods, and are often synthesized by hand-design, where both the functional form and parameters are selected using domain expertise and intuition [8, 9, 10], or derived analytically [10, 11, 12, 13, 14, 15, 16, 17]. Other approaches parameterize CBFs as polynomials and synthesize them via semidefinite programming (SDP), which provides rigorous correctness guarantees [18, 19]. Yet others hand-design the parametrization but use evolutionary algorithms to optimize parameters [20, 21], though such methods lack formal guarantees.

In this thesis, we focus on the more scalable *function-based* synthesis paradigm, in which we identify several key gaps:

Gap for uncertain systems. Most CBF synthesis work assumes a fully known model. While standard CBF synthesis for known systems is well studied [22, 23, 24], robust CBF synthesis remains preliminary [25], and adaptive synthesis has not, to our knowledge, been explored. Our proposed method (see Chapter 3.A) is one of the few to handle parametric uncertainty and the first to synthesize an *adaptive-type* CBF.

Gap for high-dimensional systems. CBF synthesis has so far been limited to small, low-dimensional benchmarks like the inverted pendulum. Hand-crafted or proof-based synthesis typically scales to only 4D; geometric set propagation techniques to 5D; Hamilton–Jacobi reachability to around 7D; and SOSP methods to 8D. Scaling beyond these limits remains an open challenge.

1.2.2 Planning for Contact-Rich Manipulation

In the first part of this thesis, we deal with high-dimensional but smooth systems, requiring only continuous state space analysis. In contrast, the second part addresses *hybrid* systems—those with both discrete and continuous components. For contact-rich manipulation, the discrete portion involves a combinatorial search over sequences of contact *states* or *modes*. Even for planar SE(2)

reorientation tasks, the branching factor can be significant: the bimanual manipulation system studied here has between 4^{10} and 4^{20} distinct contact modes. The continuous portion, which optimizes motion within a fixed contact mode, is inherently nonconvex. The resulting joint hybrid search space is high-dimensional, nonsmooth, and nonconvex, posing severe challenges for generating high-quality trajectories.

Existing approaches fall into three categories. Firstly, RL-based methods can produce strong policies for complex dexterous manipulation [26, 27, 28, 29, 30, 31, 32], but they are often sample-inefficient and require extensive reward shaping. Secondly, supervised learning methods like behavioral cloning (BC) can handle simpler contact-rich tasks such as planar pushing [33, 34] but struggle with complex cases due to the impracticality of collecting large teleoperation datasets for intricate, contact-heavy behaviors. Thirdly, model-based planners offer a more data-efficient path. We chose to work within this paradigm. Local planners are typically fast and scalable but heavily dependent on good initial guesses [35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47], while global planners can exhaustively explore the hybrid space but face a scalability–optimality trade-off. Sampling-based global planners [45, 48, 49, 50, 51] can handle dexterous reorientation but often produce inefficient trajectories, whereas mixed-integer programming (MIP) [52, 53, 54] can produce near-optimal solutions but scales poorly.

Gap for high-quality, scalable planners. To address these challenges, we explore *hierarchical planning*, which reduces the complexity of discrete search by introducing intermediate abstractions. Our central innovation is to consider *reachable sets* as our discrete unit. Each reachable set consists the range of object configurations that a local planner can reach from a given seed object-robot configuration. By covering the object space with such sets, we can construct global plans by chaining together localized ones.

For our exemplar task—object reorientation in SE(2) with a bimanual system—we can concisely cover the space with only 36 sets. As a result, the combinatorial search reduces to finding shortest paths on a small 36-node graph. This makes our approach highly efficient while still retaining generality: because the reachable-set "motion primitives" are not constrained to rigid templates, the planner can still generate fine-grained, expressive motions across a wide range of queries. In this way, we leverage reachable sets to improve both the scalability and quality of planning for contact-rich manipulation.

1.3 Thesis Overview and Summary of Contributions

Part I focuses on reactive safety for agile systems. This part opens with an overview of concepts like forward invariance and defines control barrier functions. The rest focuses on how we extend existing reachability-based safe control methods to make them more applicable to real-world systems, which are often uncertain and high-dimensional. Namely:

- Sec. 3 adopts a formal methods perspective. Using sum-of-squares programming (SOSP), we synthesize controllers with rigorous mathematical guarantees of safety, while relaxing the

common assumption of a perfectly known model. Instead, we target a class of uncertain systems with fixed but unknown affine parameters, developing a robust-adaptive safe controller. We also consider systems of moderate dimensionality (up to 8D)—more challenging than the toy examples typical in SOSP literature. Importantly, the resulting safe controller minimizes conservatism, reducing unnecessary interference with performance-oriented objectives such as time-to-goal. Experiments show that our controller maintains 100% safety while interfering 55% less with performance objectives compared to an alternative safety controller for uncertain systems.

- Sec. 4 addresses the scalability limitations of formal methods observed in Chapter 3.A. Formal approaches can become prohibitively slow for systems above 8D and often yield overly conservative controllers. To address this, we turn to machine learning, representing the CBF as a neural network and training it adversarially to ensure both validity and minimal conservatism. This yields a safe controller that is nearly 100% safe and substantially less conservative than a state-of-the-art MPC baseline, demonstrating that function-based synthesis methods can be made more scalable without sacrificing safety.

Part II turns to motion planning for contact-rich manipulation, extending reachable set techniques into the domain of long-horizon, hybrid search. We develop a hierarchical planner for complex bimanual tasks in which reachable sets, computed in object space, form the fundamental planning units. Planning at this granularity allows us to search efficiently while retaining the richness needed to capture complex contact interactions. The result is a method that scales to challenging systems while producing significantly higher-quality trajectories than state-of-the-art RRT-based approaches.

Finally, Chap. 6 concludes the thesis, summarizing contributions across both domains and outlining directions for future research, with an emphasis on the broader potential of reachable sets as a unifying computational primitive for robotics.

I

Reactive Safety for Agile Systems

2 Background on Invariant Sets and Safe Control

In this section, we review invariant sets and safe control.

Notation For a function $c : \mathbb{R}^n \to \mathbb{R}$, define its zero-sublevel set as $\mathcal{C} \triangleq \{c\}_{\leq 0}$, its boundary as $\partial\mathcal{C} = \{c\}_{=0}$, and its interior as $\text{Int}(\mathcal{C}) = \{c\}_{<0}$. We assume the following:

1. A control-affine system

$$\dot{x} = f(x) + g(x)u, \quad x \in \mathcal{D} \subset \mathbb{R}^n,\ u \in \mathcal{U} \subseteq \mathbb{R}^m,$$

   where $f : \mathbb{R}^n \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ are locally Lipschitz.

2. An input set $\mathcal{U}$, modeled as a bounded convex polytope.

3. A safety specification $\rho : \mathbb{R}^n \to \mathbb{R}$, which defines the *allowable set*

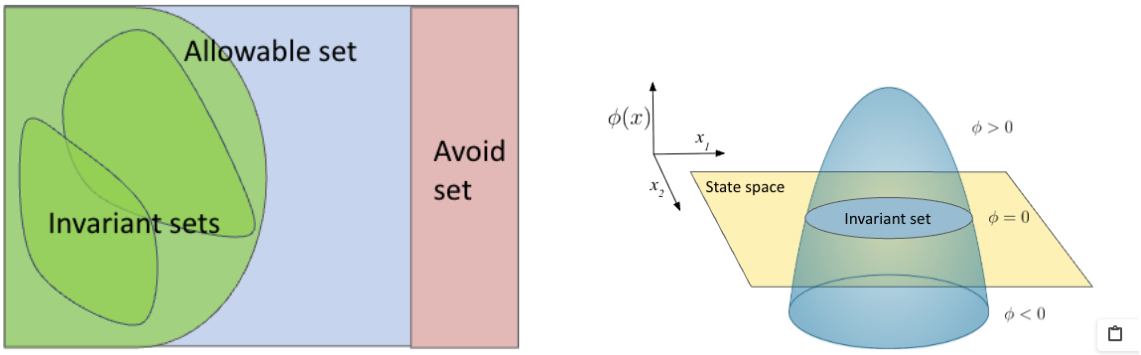
$$\mathcal{A} \triangleq \{\rho\}_{\leq 0}.$$

   We assume ρ is smooth and continuous.

The *relative degree* $r \in \mathbb{Z}^+$ of $\rho(x)$ with respect to u is the smallest derivative of $\rho(x)$ in which u explicitly appears.

Safe Control and Invariant Sets Safe control is the problem of finding a *control invariant set* within $\mathcal{A}$ (Fig. 2.1a). A set is control invariant if there exist admissible inputs such that any trajectory starting inside it remains inside for all time. Such a set immediately induces a safety controller: the system is repelled from its boundary. Thus, synthesizing a safe controller reduces to computing an invariant set.

Because many invariant sets may exist, we define an *optimal set* as the largest control invariant set, since it yields the least conservative safety filter. The larger the set, the more the system is able to roam freely inside the set to achieve performance-oriented objectives.



(a) Figure defining concepts like "allowable set", "control invariant sets". Note that many invariant sets are possible and we seek the largest one.

(b) Figure illustrating the function-based invariant set synthesis paradigm. We seek a function ϕ that implicitly defines an invariant set as $\{\phi\}_{\geq 0}$.

Figure 2.1: Figure defining basic invariant set and control barrier function concepts.

Within the function-based paradigm (Sec. 1.2.1), we seek a function whose zero-superlevel set encodes a control invariant set (Fig. 2.1b). Such a function is called a *control barrier function (CBF)*, denoted ϕ.

Next, to define a safe controller using ϕ is straightforward. A safe controller simply needs to repel the system back into the safe set whenever it reaches the set boundary. The following theorem formalizes this idea:

Theorem 2.0.1 (Taken from [11])**.** *Given a CBF ϕ and safe set $\mathcal{X}_{safe}$, any feedback controller $k(x) : \mathbb{R}^n \to \mathbb{R}^m$ satisfying*

$$\dot{\phi}(x, k(x)) \triangleq \underbrace{\nabla\phi(x)^\top f(x)}_{L_f\phi(x)} + \underbrace{\nabla\phi(x)^\top g(x)}_{L_g\phi(x)} k(x) \geq 0 \quad \forall x \in \partial\mathcal{X}_{safe} \tag{2.1}$$

renders the system forward invariant over $\mathcal{X}_{safe}$.

Note that you can also require a CBF to satisfy a stricter inequality $\dot{\phi}(x) \geq -\alpha(\phi(x))$ for all $x \in \mathcal{D}$, where $\alpha(\cdot)$ is a class-κ function. We elaborate on this alternative in Appendix Sec. A2.2. The theorem above requires the controller to repel the system (increase ϕ) from the boundary ($x \in \partial\mathcal{X}_{safe}$). We are allowed to use any nominal controller, k_{nom}, as long as we modify its inputs to satisfy Eqn. 4.1 at the boundary. Thus, a CBF-based safe controller simply filters (modifies) a nominal controller online by applying the QP below at every step of control execution:

$$k(x) = \underset{u \in \mathcal{U}}{\operatorname{argmin}} \ \frac{1}{2} \|u - k_{nom}(x)\|_2^2 \tag{CBF-QP}$$

$$\text{s.t.} \qquad L_f\phi(x) + L_g\phi(x)u \leq \begin{cases} 0 & \text{if } x \in \partial\mathcal{X}_{safe} \\ \infty & \text{o.w.} \end{cases} \tag{2.2}$$

3 SOSP Synthesis of Robust-adaptive CBF for Uncertain Systems

3.1 Introduction

Although much work has been done on safety for known systems, comparatively little work has been done for systems whose dynamical models cannot perfectly be known. These uncertain systems are a challenging domain for safe control on both a theoretical and methodological level. Effective safe controllers must meet two demands: (1) provide mathematical guarantees of safety and (2) be high-performance, interfering minimally with other control objectives (i.e., stabilizing, tracking). Currently, many safe control synthesis methods for uncertain systems either lack guarantees or only obtain safety at a high cost to performance. Secondly, algorithms for controller search should be both generic and tractable, so they can handle many different systems of varying dimensions. This work aims to address all of these requirements.

The technique we use is "sum-of-squares programming" (SOSP) [55, 56]. Any robust-adaptive CBF (raCBF) produced by our SOSP-based algorithm is guaranteed to provide safety. SOSP allows us to pose the search for a valid raCBF, given a partially-known system and a safety requirement, as a constrained optimization problem. This optimization problem can be broken down into a series of convex optimization problems, which can each be solved with an off-the-shelf semidefinite program (SDP) solver.

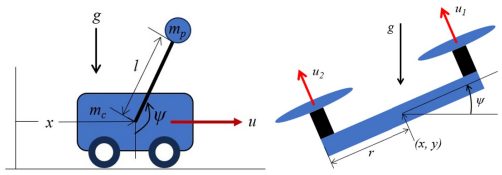


Figure 3.1: Two of our three test systems: cartpole with unknown joint friction and quadrotor with unknown drag coefficients.

Typically, uncertain systems are handled with adaptive, robust, or robust-adaptive control [57]. We choose the robust-adaptive paradigm, which combines the merits of robust and adaptive control. In adaptive control, the controller depends on estimates of the unknown system. Typically, the

system will be estimated online and the controller *adapts* with each new estimate. In adaptive fashion, our raCBF is a function of the estimated parameters as well as the state. This raCBF is also paired with a special parameter estimation law. This controller structure allows it to effectively *select* a CBF online with the choice of parameter. The robust-adaptive scheme leverages this to attain *selective conservatism*: more conservative CBFs are only "selected" when necessary, like when the state suddenly moves toward danger.

The raCBF also incorporates elements of robust control. In robust control, the controller only depends on worst-case disturbance bounds. Our raCBF follows robust principles by accounting for maximum estimation error. However, it is able to account for it in a less conservative way compared to robust CBF (rCBF), as it assumes parameter estimation occurs. Ultimately, these differences lead our synthesized raCBF to perform better. That is, it provides safety while minimally hindering control efficiency at tasks like stabilization and tracking.

Our contributions are as follows:

- Propose the first verification and synthesis methods for adaptive-type safe controllers for uncertain systems.
- For verification: derive convex-equivalent conditions for an raCBF to provide safety.
- For synthesis: design a multi-phase algorithm which yields a valid raCBF with a locally optimal invariant set.
- Illustrate our algorithm's genericness and scalability by applying it to three examples with varying dynamics, dimensions ($\leq$ 7D), and safety specifications.
- Illustrate our synthesized raCBF's safety and superior performance (up to 55% improvement over robust baseline) in simulation.

3.2 Related works

In this section, we briefly survey existing works on safety for uncertain systems. We primarily focus on CBF works.

Theory of (r/a)-CBFs: these works mathematically define novel forms of CBF (like robust [58, 59, 60] and adaptive [61, 62, 63] CBFs). This category has a different focus than synthesis and verification (systematically generating CBFs). Hence, works in this category typically just use hand-derivation to find CBFs for toy systems, which is an approach that is difficult, if not impossible, to apply to more complex systems.

Verification and synthesis of (r/a)-CBFs: this category, which our work belongs to, deals with developing new theories and algorithms for systematically checking and generating safety functions. While synthesis of ordinary CBFs (i.e., for known systems) has been explored extensively [22, 23, 24], to our knowledge, there are very few works for robust CBF synthesis [25] and none for adaptive synthesis. Our work is one of a handful that synthesize CBFs for uncertain systems, and the first to synthesize an adaptive-type CBF. While our synthesis technique carries guarantees

of correctness, there are ad-hoc synthesis works which do not [20, 21]. There are also sample-based methods for verification, which are unable to scale past 5D due to the curse of dimensionality [21]. Compare this to our SOSP-based verification scheme, which has polynomial time complexity and can scale to 7D and possibly beyond.

Online optimization of (r/a)-CBFs: this body of work typically assumes a valid CBF already exists. Ordinary CBFs compute safe control inputs using quadratic programs (QPs) solved at each time step [64, 65]. Works in this category investigate how best to pose this online optimization (i.e., possibly as a different type of program, like a second-order cone program) for new forms of CBFs [66, 67, 68]. Note that one advantage of our robust-adaptive formulation is that it can still be applied with a QP, which is simple and fast to solve.

Ad-hoc safe controllers: these are works which cannot guarantee safety, because they cannot prove that an admissible safe control input exists for all states and all realizations of the unknown system [69, 70, 71]. By contrast, our approach can ensure that there exists a safe control input for *all possible conditions*, therefore guaranteeing system safety.

Non-CBF approaches: there are bodies of work on using HJB reachability [1] or set-based methods [72, 73] to compute robust backward or forward reachable sets. However, a significant drawback of these methods is they scale poorly, often not beyond 5D.

3.3 Problem formulation

3.3.1 Safe control for uncertain systems

In this first subsection, we define the problem of safe control for uncertain systems. We also outline system assumptions and our definition of safety. We consider a system that is affine in the unknown parameters and the control input:

$$\dot{x} = f(x) - \Delta(x)\theta + g(x)u \tag{3.1}$$

$$u \in \mathcal{U}, x \in \mathcal{X}, \theta \in \Theta \tag{3.2}$$

where f, g, Δ are polynomial functions in x and $x \in \mathcal{X} \subseteq \mathbb{R}^n, u \in \mathcal{U} \subseteq \mathbb{R}^m, \theta \in \Theta \subset \mathbb{R}^p$. We also assume the set of admissible inputs and the robust parameter range are both hyperrectangles (i.e., $\underline{u} \leq u \leq \bar{u}$ and $\underline{\theta} \leq \theta \leq \bar{\theta}$). We want to enforce safety of this system in the sense of "set invariance" (Def. **??**).

Definition 3.3.1 (Set invariance)**.** A set $\mathcal{I}$ is forward invariant if all trajectories starting inside $\mathcal{I}$ remain inside the set for all time.

The set $\mathcal{X}_{\text{safe}}$ is provided as part of the problem specification and is represented implicitly using continuous and smooth functions $b_i(x) : \mathbb{R}^n \to \mathbb{R}$:

Definition 3.3.2 (Safety specification)**.**

$$\mathcal{X}_{\text{safe}} = \{x | b_i(x) \geq 0, \ \forall i = 1, \ldots, t\} \tag{3.3}$$

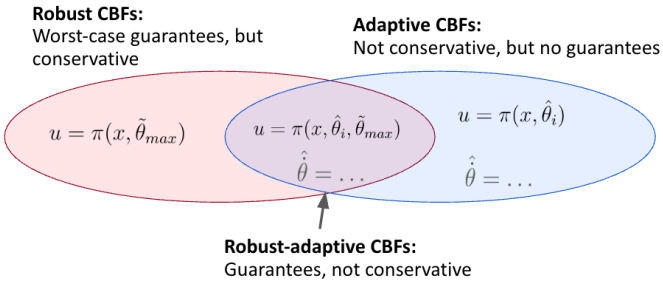


Figure 3.2: This figure illustrates why we chose a robust-adaptive type CBF.

3.3.2 Robust-adaptive CBFs

In this second subsection, we summarize the raCBF formulation and outline the constraints that valid raCBFs must satisfy. We choose the formulation in [74], which is less conservative than [58, 59, 60, 61, 63] and compatible with more nominal controllers than [62]. The raCBF has the form $\phi(x, \hat{\theta})$ where it depends on not only the state, but also on the estimate of the unknown parameters. The estimate, $\hat{\theta}$, is initialized with a guess and then updated online according to:

$$\dot{\hat{\theta}} = \gamma \nu(\rho) \Delta(x)^\top \nabla_x \phi(x, \hat{\theta}), \tag{3.4}$$

$$\dot{\rho} = \frac{\nu(\rho)}{\nabla \nu(\rho)} \frac{1}{\phi(x, \hat{\theta}) + \eta} \left[-\sigma \rho + w_{\hat{\theta}}(x) \right], \tag{3.5}$$

$$w_{\hat{\theta}} = \begin{cases} 0 \text{ if } \nabla_{\hat{\theta}} \phi(x, \hat{\theta})^\top \dot{\hat{\theta}} \geq 0 \\ -\zeta \nabla_{\hat{\theta}} \phi(x, \hat{\theta})^\top [\gamma \Delta(x)^\top \nabla_x \phi(x, \hat{\theta})] \text{ otherwise} \end{cases} \tag{3.6}$$

where γ is an admissible adaptation gain, $\nu(\rho)$ is a scaling function such that $1 \leq \nu(\rho) \leq \zeta < \infty$ for $\zeta > 1$ and also $\nabla \nu(\rho) > 0$. Also, $\eta, \sigma, \zeta \in \mathbb{R}_{>0}$ are design parameters. We assume that θ belongs to a bounded set Θ. For many systems, we can easily provide a guess for Θ based on domain knowledge [57].

Theorem 3.3.1. *Define the set* $\mathcal{I}^r_{\hat{\theta}} = \{x \in \mathcal{X}, \hat{\theta} \in \Theta \mid \phi(x, \hat{\theta}) \geq \frac{1}{2\gamma} \tilde{\theta}^\top \tilde{\theta}\}$ *where* $\tilde{\theta}$ *is the maximum possible estimation error.* $\tilde{\theta} = \bar{\theta} - \underline{\theta}$*, where* $\theta \in [\underline{\theta} : \bar{\theta}]$*. Then, an raCBF must satisfy the following constraints to ensure invariance of* $\mathcal{I}^r_{\hat{\theta}}$*:*

$$\sup_{u \in \mathcal{U}} \left\{ \nabla_x \phi(x, \hat{\theta})(f - \Delta \hat{\theta} + gu) \right\} \geq -\alpha \left(\phi(x, \hat{\theta}) - \tfrac{1}{2\gamma} \tilde{\theta}^\top \tilde{\theta} \right) \tag{3.7}$$

$$\phi(x, \hat{\theta}) \geq 0 \implies b_i(x) \geq 0, \forall i = 1, \dots, t. \tag{3.8}$$

Proof. Eq. (3.7) by Thm. 1 in [74] and Eq. (3.8) by defintion of safety. □

Eq. (3.7) says that the raCBF must be a function that defines a robust invariant set under the true dynamics. Specifically, the robust set $\mathcal{I}^r_{\hat{\theta}}$ is invariant iff there always exists an admissible input to repel the state from the set boundary. This is implied by Eq. (3.7), which requires $\sup_{u \in \mathcal{U}} \dot{\phi} \geq 0$ (repulsion) when $\phi = \frac{1}{2\gamma} \tilde{\theta}^\top \tilde{\theta}$ (state at the boundary). Now, typically, the invariant set is considered $\{x | \phi \geq 0\}$. The robust set is shrunk from this typical set by a margin proportional to the maximum

estimation error. Intuitively, this triggers safe control earlier than is typical, which mitigates the impact of estimation error.

Additionally, the raCBF must obey the safety specification. Eq. (3.8) says that the raCBF may define an invariant set that is no larger than $\mathcal{X}_{\text{safe}}$. Often, $\mathcal{X}_{\text{safe}}$ itself cannot be an invariant set. There will be some states within $\mathcal{X}_{\text{safe}}$ that will leave the set for any admissible control input.

Given a valid raCBF, a controller design is immediate. We simply apply the raCBF as an optimization-based safety-filter on top of a nominal control policy. Since any control input satisfying Eq. (3.7) will keep the system safe, this filter simply projects the nominal control signal to the space of u satisfying Eq. (3.7).

Definition 3.3.3 (raCBF quadratic program (raCBF-QP))**.** Given nominal controller $u_{\text{ref}} = \pi(x)$,

$$\min_{u \in \mathcal{U}} \|u_{\text{ref}} - u\|_2^2 \tag{3.9}$$

$$\nabla_x \phi(x, \hat{\theta})(f - \Delta\hat{\theta} + gu) \geq -\alpha \left(\phi(x, \hat{\theta}) - \tfrac{1}{2\gamma} \tilde{\theta}^\top \tilde{\theta} \right). \tag{3.10}$$

Notice that the filtered input is computed using the *estimated* dynamics. This is why we require the raCBF, and the filtered input, by consequence, to be robust to estimation error.

3.4 Preliminaries: SOSP

We use sum-of-squares programming (SOSP) as the optimization framework for verification and synthesis. In the following section, we briefly summarize key definitions and theorems for SOSP. In general, checking polynomial non-negativity is NP-hard. However, checking that some $p(x) \in \mathcal{R}_n$ (the set of polynomials in n variables) is an SOS polynomial (Def. 3.4.1) is completely tractable [75].

Definition 3.4.1 (Sum-of-squares polynomial)**.** A polynomial $p(x)$ is SOS if $p(x) = \sum_i q_i(x)^2$ for some polynomials q_i. Clearly, this implies $p(x) \geq 0$. We denote the set of SOSP as Σ.

Hence, a popular kind of optimization problem is the SOSP, which has constraints of this form (enforcing a polynomial to be sum-of-squares).

Definition 3.4.2 (Sum-of-squares program)**.** An SOSP is a convex optimization problem of the following form for a given cost vector $c \in \mathbb{R}^{p \times q}$

$$\min_{w} \; c^\top w \tag{3.11}$$

$$a_0^i(x) + \sum_{\forall j \in 1:p} w_j^i a_j^i(x) \in \Sigma \;\; , \; \forall i = 1, \ldots, q \tag{3.12}$$

where $a_j^i(x) \in \mathcal{R}_n$.

The definition above says that an SOSP constrains q polynomials to be sum-of-squares. These polynomials may have variable coefficients (w_j^i) and the objective is linear in these coefficients. Next, we introduce the cornerstone theorem, the Positivstellensatz (P-satz), which allows us to

transform all different kinds of constraints (e.g., logical implications of polynomial inequalities) into the standard SOSP constraint form of Eq. (3.12).

Theorem 3.4.1 (Positivstellensatz [76]). *Given polynomials* $\{f_1, \ldots, f_r\}$, $\{g_1, \ldots, g_t\}$, *and* $\{h_1, \ldots, h_u\}$ *in* $\mathcal{R}_n$, *the following are equivalent:*
(a) The set

$$\left\{ x \in \mathbb{R}^n \,\middle|\, \begin{array}{l} f_1(x) \geq 0, \ldots, f_r(x) \geq 0 \\ g_1(x) \neq 0, \ldots, g_t(x) \neq 0 \\ h_1(x) = 0, \ldots, h_u(x) = 0 \end{array} \right\}$$

is empty.
(b) There exist polynomials $f \in \mathcal{P}(f_1, \ldots, f_r), g \in \mathcal{M}(g_1, \ldots, g_t), h \in \mathcal{I}(h_1, \ldots, h_u)$ *such that*

$$f + g^2 + h = 0 \tag{3.13}$$

where $\mathcal{M}(g_1, \ldots, g_t)$ *is the multiplicative monoid generated by the* g_i*'s and consists of all their finite products;* $\mathcal{P}(f_1, \ldots, f_r)$ *is the cone generated by the* f_i*'s:*

$$\mathcal{P}(f_1, \ldots, f_r) = \left\{ s_0 + \sum_{i=1}^{l} s_i b_i \,\middle|\, \begin{array}{l} l \in \mathbb{Z}_+, s_i \in \Sigma_n, \\ b_i \in \mathcal{M}(f_1, \ldots, f_r) \end{array} \right\}$$

and finally, $\mathcal{I}(h_1, \ldots, h_u)$ *is the ideal generated by the* h_i*'s:*

$$\mathcal{I}(h_1, \ldots, h_u) = \left\{ \sum h_k p_k \middle| p_k \in \mathcal{R}_n \right\}. \tag{3.14}$$

The P-satz allows us to convert statements in the form of (a) to the form of (b), which can then straightforwardly be turned into an SOSP-type constraint. However, it is often inconvenient to apply P-satz directly, since it generates complex constraints (e.g., the cone adds 2^r terms, each with a polynomial multiplier). Instead, we can use a simplification of P-satz called the S-procedure:

Definition 3.4.3 (S-procedure [75]). A sufficient condition for proving $p(x) \geq 0$ on the semi-algebraic set $K = \{x \in \mathbb{R}^n | b_1(x) \geq 0, \ldots, b_m(x) \geq 0\}$ is the existence of polynomials $r_i(x), i = 0, \ldots, m$ that satisfy

$$(1 + r_0(x))p(x) - \sum_{i=1}^{m} r_i(x) b_i(x) \in \Sigma \tag{3.15}$$

$$r_i(x) \in \Sigma, i = 0, \ldots, m. \tag{3.16}$$

Frequently, we simplify this condition further by taking $r_0(x) = 0$. Together, the concepts from this section allow us to write the constraints which define valid raCBFs in the form of SOSP constraints, which makes them amenable to verification and synthesis.

3.5 Methodology

In the following sections, we show how a valid raCBF can be characterized using several SOSP constraints. With this, we can pose raCBF verification as an SOSP; this allows us to efficiently certify whether a given raCBF is valid. Additionally, we propose an algorithm for synthesizing an raCBF, given the partially unknown system and the safety definition. The algorithm is a sequence of SOSPs that are iteratively solved until convergence.

3.5.1 raCBF verification

In this section, we derive an SOSP that allows us to verify if a given raCBF satisfies Theorem 3.3.1. Our goal is to convert Eq. (3.7), (3.8) into the form of Eq. (3.12).

Looking first at Eq. (3.7), we see that it requires us to show that for all $x \in \mathcal{X}, \theta \in \Theta$, there exists some $u \in \mathcal{U}$ satisfying the constraint. The "exists" quantifier complicates the conversion, as theorems like P-satz and S-procedure can only support "for all" quantifiers. However, it turns out that for any x, we can identify which $u \in \mathcal{U}$ achieves the supremum in Eq. (3.7) - call it $u^*(x)$. Therefore, we only need to check whether Eq. (3.7) is satisfied for $u^*(x)$ for all x.

We derive $u^*(x)$ by regarding Eq. (3.7) as a constrained optimization over the input u. In fact, it is a special kind of optimization problem with an affine objective and a compact, convex constraint set. Hence, the maximizer always occurs at a vertex of the constraint set [77], v^j where $v_i^j \in \{\underline{u}_i, \bar{u}_i\}$. For each x, we can further identify which vertex is the maximizer. First, we write Eq. (3.7) more clearly in control-affine form. Let $c(x,\hat{\theta}) := \nabla_x \phi(x,\hat{\theta})(f - \Delta\hat{\theta}) + \alpha(\phi(x,\hat{\theta}) - \frac{1}{2\gamma}\tilde{\theta}^\top\tilde{\theta})$ and $d(x,\hat{\theta}) := \nabla_x \phi(x,\hat{\theta})g(x)$, so that Eq. (3.7) can be rewritten as $\sup_{u\in\mathcal{U}} c(x,\hat{\theta}) + d(x,\hat{\theta})u$. For $i = 1,\ldots,m$, if $d(x,\theta)_i \geq 0$, i.e., the i^{th} entry of vector $d(x,\theta)$ is nonnegative, then $u_i^*(x) = \bar{u}_i$ and vice versa. There are then 2^m groups of x with distinct sign patterns on $d(x,\theta)$ that have distinct maximizers $u^*(x)$. Hence, we can write 2^m constraints for all the groups of x.

We give an example for $m = 1$, although the general case is an immediate extension. We have two constraints:

$$\begin{aligned}
&\forall x \in \mathcal{X}, \theta \in \Theta, \ \text{ s.t. } \ \phi(x,\theta) \geq \beta^- : \\
&d(x,\theta) \geq 0 \implies c(x,\theta) + d(x,\theta)\bar{u} \geq 0, && (3.17)\\
&d(x,\theta) \leq 0 \implies c(x,\theta) + d(x,\theta)\underline{u} \geq 0. && (3.18)
\end{aligned}$$

Applying the S-procedure gives us the following SOSP constraints:

$$c + d\bar{u} - s_1 d - s_2(\phi - \beta^-) \in \Sigma, \ \ s_1, s_2 \in \Sigma, \tag{3.19}$$

$$c + d\underline{u} + s_3 d - s_4(\phi - \beta^-) \in \Sigma, \ \ s_3, s_4 \in \Sigma. \tag{3.20}$$

Likewise, we can also apply the S-procedure to (3.8) to yield:

$$\phi(x,\theta) - r_i b_i \in \Sigma, \ \ \forall i = 1,\ldots,t \ , \ r_i \in \Sigma. \tag{3.21}$$

Theorem 3.5.1 (raCBF verification)**.** *For the case where $m = 1$[1], a polynomial $\phi(x, \hat{\theta})$ is a valid raCBF, providing set invariance and finite-time convergence in $\{x|\phi \geq \beta^-\}$, if there exists $s_{1:4}, r_i$ such that the convex constraints in Eq. (3.7) and (3.8) are satisfied. This can be determined by seeing if the SOSP consisting of constraints (3.19), (3.20), (3.21) is feasible.*

Proof. If there exists such $s_{1:4}, r_i$, then by Def. 3.4.3, Thm. 3.3.1 is satisfied. □

Algorithm 1 Synthesis of raCBFs via a sequence of three SOSPs

Start with $\phi^{(0)}, \theta_l^{(0)}, \theta_u^{(0)}, c^{(0)} = \infty, i = 0$, converged=False. Denote the objective value as $c^{(i)}$.
while not converged **do**
  Ellipsoid $\mathcal{E}_\delta^{(i)} \leftarrow$ **P1**$(\phi^{(i)}, \theta_l^{(i)}, \theta_u^{(i)})$
  Multipliers $s_i(x,\theta), r_i(x,\theta) \leftarrow$ **P2**$(\phi^{(i)}, \theta_l^{(i)}, \theta_u^{(i)}, \mathcal{E}_\delta^{(i)})$
  $\phi^{(i+1)}, \theta_l^{(i+1)}, \theta_u^{(i+1)}, c^{(i+1)} \leftarrow$ **P3**$(s_i(x,\theta), r_i(x,\theta), \mathcal{E}_\delta^{(i)})$
  if $\left|c^{(i+1)} - c^{(i)}\right| \leq \epsilon$ **then**
    converged $\leftarrow$ True
  end if
end while=0

3.5.2 raCBF synthesis

In this section, we provide an algorithm for synthesizing an raCBF given the partially known system (Eq. 3.1), the safety definition (Eq. 3.3), and a feasible initialization for the raCBF. We would like to find $\phi(x, \hat{\theta})$ satisfying Theorem 3.3.1, where ϕ is now a polynomial of fixed degree over x, θ with variable coefficients. However, with ϕ containing optimization variables, Eq. (3.19), (3.20) are no longer convex, but bilinear. This requires us to apply bilinear alternation on these constraints, which form the basis of our Alg. 1. We optimize the multipliers $s_i(x,\theta), r_i(x,\theta)$ in Phase 2 and the raCBF $\phi(x, \hat{\theta})$ in Phase 3 and continue back and forth.

Phase 2 (3.22)

$$\min_{s_i, r_i \in \Sigma, t \in \mathbb{R}} t$$

$$c + d\bar{u} - s_1 d - s_2(\phi - \beta^-) \in \Sigma \qquad \text{Ctrl fltr feas. constrs}$$

$$c + d\underline{u} + s_3 d - s_4(\phi - \beta^-) \in \Sigma$$

$$\phi - r_i b_i \in \Sigma \ , \ \forall i = 1, \ldots, t \qquad \text{Subset constraints}$$

$$t - \phi - s_5(\delta - x^\top P x) \in \Sigma \qquad \text{Ellipsoid constraint}$$

$$1 - \phi(0, \theta) \in \Sigma \qquad \text{Anchor constraint}$$

[1]The case for general m is an immediate extension, but omitted for notational brevity.

Phase 3 (3.23)

$$\begin{aligned}
&\min_{\phi,t,\theta_l,\theta_u} \check{\mathrm{N}}t + \theta_l - \theta_u \\
&c + d\bar{u} - s_1 d - s_2(\phi - \beta^-) \in \Sigma && \text{Ctrl fltr feas. constrs} \\
&c + d\underline{u} + s_3 d - s_4(\phi - \beta^-) \in \Sigma \\
&\phi - r_i b_i \in \Sigma \ , \ \forall i = 1, \ldots, t && \text{Subset constraints} \\
&t - \phi - s_5(\delta - x^\top P x) \in \Sigma && \text{Ellipsoid constraint} \\
&1 - \phi(0, \theta) \in \Sigma && \text{Anchor constraint}
\end{aligned}$$

While bilinear alternation alone would yield a *valid* raCBF, we'd also like the raCBF to capture a large invariant set. A larger invariant set means the system is less restricted, providing the nominal controller more flexibility. Thus, our optimization needs to also maximize the size of the invariant set $\mathcal{I}^r_{\hat{\theta}}$. To do this, we follow [24] (Fig.). To increase its size, we perform several steps in each iteration. First, we find the largest ellipsoid $\mathcal{E}_\delta := \{x \mid x^\top P x \leq \delta\}$ contained within $\mathcal{I}^r_{\hat{\theta}}$ (solve Phase 1 for the maximum ellipsoid radius δ). Then, we optimize the raCBF so that the margin between $\mathcal{E}_\delta$ and $\mathcal{I}^r_{\hat{\theta}}$ is as large as possible (Phase 2). This effectively increases the size of $\mathcal{I}^r_{\hat{\theta}}$ over the iterations. We also add an "anchor constraint", which limits the maximum value of the raCBF. This helps us avoid a trivially optimal raCBF with large values but a small invariant set.

Phase 1 (3.24)

$$\begin{aligned}
&\max_{\delta \in \mathbb{R}, s_0 \in \Sigma} r \\
&\text{s.t. } \ \delta - x^\top P x - s_0 \phi \in \Sigma
\end{aligned}$$

Another challenge posed by synthesis is the requirement for a feasible initialization. That is, we need to supply a raCBF already satisfying Eq. (3.7), (3.8) when using bilinear alternation. Generally, this initialization is something that can be computed straightforwardly, but has a trivially small invariant set. Then, the idea is for synthesis to enlarge the invariant set, while maintaining validity, until it has a reasonable size. We can easily provide such an initialization if the robust range Θ is small. In that case, we can use the control Lyapunov function (CLF) corresponding to a linear quadratic regulator (LQR) which stabilizes to an equilibrium point in $\mathcal{X}_{\text{safe}}$. This initialization will intrinsically have some robustness to perturbation [78]. However, in the general case, the robust range Θ will have moderate size.

To handle this, we consider a small robust range at initialization, so that we may use our LQR-based CLF initialization. Then, we maximize this robust range over the iterations (Phase 3). In our experiments, the ultimate robust range produced by our algorithm is always equal to the desired range. However, we have found, for example, that the algorithm is unable to increase the robust range to double the desired size. This is not a significant issue though, since robust control is

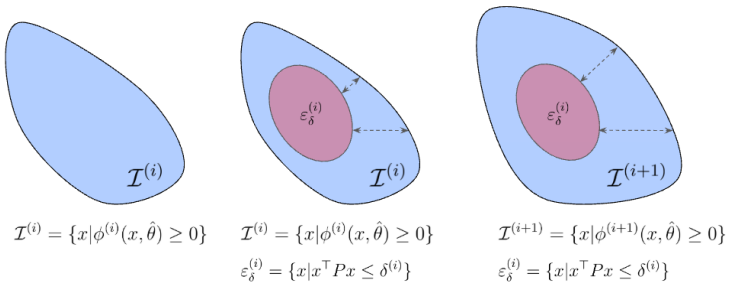


Figure 3.3: This figure illustrates the invariant set size maximization process.

generally not recommended when the desired range is large [57].

Table 3.1: Results for feedback simulations. We use 200 random trials for each test. For the performance test, the cartpole task metric is time-to-reach x_{goal} and the quadrotor's task metric is the average trajectory tracking error.

	Toy 2D	Cartpole	Quadrotor
Safety Test (% safe trajectories)	100%	100%	100%
Performance Test (% improvement of raCBF over rCBF on task metric)	–	7.014%	54.953%

3.6 Results

In this section, we synthesize raCBFs for three different systems: toy 2D, cartpole, and quadrotor. We show that our algorithm is quite general, producing valid raCBFs for a variety of dynamics and safety specifications. We also compare closed-loop performance in simulation for our robust-adaptive controller and a baseline robust controller [25]. We find that our robust-adaptive safety filter ensures 100% safety *while also* interfering less with the nominal controller. This means it achieves significantly better performance on combined safety-stabilization and safety-tracking tasks. Finally, we also show that our method scales better than the baseline, handling systems of 7D as opposed to 2D.

In particular, we run two sets of experiments with the simulated closed-loop system:

1. **Safety test**: ensure the system does not leave the invariant set under an adversarial nominal controller, which tries to drive the system out. Trials vary the initial state and the true parameter(s) value(s).

2. **Performance test**: measure stabilizing/tracking performance on combined safety-(stabilizing/tracking) task. Compare with baseline, a robust CBF [25]. Trials vary the true parameter(s) value(s). The cartpole's task metric is time to reach the goal and the quadrotor's task metric is the average trajectory tracking error.

We use 200 random trials for each test. Results are listed in Table 3.1 and analyzed further in the following subsections. All the experimental hyperparameters can be found in the Appendix (Sec. **??**). To solve the SOSPs, we use SOSTOOLS [79] with Mosek [80] on a 3.00GHz Intel(R) Core(TM) i9-9980XE CPU with 16 cores and 126GB of RAM. For all examples, the initial raCBF is derived from an LQR-based CLF that stabilizes the system to the origin. Specifically, we use the solution S of the algebraic Riccati equation of the system linearized about the origin to define the CLF $V(x) = x^T S x$ and the raCBF $\phi^{(0)}(x) = \epsilon - V(x)$, where $\epsilon \in \mathbb{R}_{>0}$ is chosen experimentally.

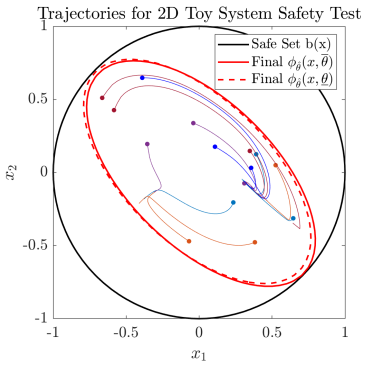


Figure 3.4: Random trajectories generated by the safety test. Observe that all stay within the invariant set (in red).

3.6.1 Toy 2D

We use the 2D toy system from [81]:

$$\dot{x} = \begin{bmatrix} x_2 - x_1^3 \\ 0 \end{bmatrix} - \begin{bmatrix} -x_1^2 \\ 0 \end{bmatrix} \theta + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \tag{3.25}$$

with $\Theta \in [0.8, 1.5]$ and $u \in [-2, 2]$. The safety definition is $\mathcal{X}_{\text{safe}} = \{x \mid b(x) := 1 - (x_1^2 + x_2^2) \geq 0\}$.

Our synthesized raCBF attains 100% safety on this system. Observe in Fig. 3.4 that the trajectories under the safety-filtered adversarial controller never leave the system despite the parametric uncertainty.

3.6.2 Cartpole

For the cartpole system (Fig. 3.1), we transform the standard trigonometric dynamics [82] to polynomial form by redefining the states as $[\dot{x}, x, \dot{\psi}, \sin(\psi), \cos(\psi) + 1]$. For these new states, we require $\sin^2(\psi) + \cos^2(\psi) = 1$, which we incorporate into the SOSP using the S-procedure [83]. The unknown parameter is the friction coefficient at the joint. We let $\Theta \in [0.28, 0.31]$, following [84] and $u \in [-5, 5]$ N. The safety definition confines the pole angle to be small: $\mathcal{X}_{\text{safe}} := \{\sin^2(\frac{\pi}{3}) - \sin^2(\psi) \geq 0\}$.

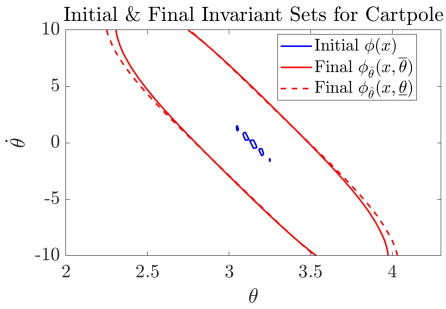


Figure 3.5: Note the significant increase in size from the initial to final invariant sets. Also, observe that the final raCBF depends on $\hat{\theta}$ in an intuitive way.

In Fig. 3.5, we plot the invariant set along $(\theta, \dot{\theta})$. First, observe that the shape of the set is sensible: it is an ellipsoid with primary axis $\dot{\theta} = -\theta$, meaning it considers the pole swinging to the vertical as safe and vice versa. Secondly, note that the synthesized raCBF depends meaningfully on $\hat{\theta}$. Comparing the invariant sets for $\phi(x, \underline{\theta})$ (low joint friction) and $\phi(x, \bar{\theta})$ (high joint friction), we can see that the former set is larger. This makes sense, since for low joint friction, the pole is more responsive and easily controlled. Finally, notice that the initial invariant is almost negligibly small, but the final set is quite large. We conclude that our algorithm is very effective at growing the invariant set.

As for the closed-loop simulations, raCBF attains 100% safety, even with an adversarially unsafe nominal controller. We also synthesize a robust CBF according to [25]. In the performance test, we implement a nominal controller which controls the cart to reach position x_{goal}. Then, the nominal controller is either layered with an raCBF-based or an rCBF-based safety filter, both of which prevent the pole from tipping.

Overall, the raCBF controller reaches the goal 7.014% faster. One reason for this is that the raCBF has a larger invariant set (Fig. 3.6), which allows larger pole angles during safe operation (Fig. 3.7) and hence larger cart acceleration. The difference in invariant set size can be explained as follows: while both raCBF and rCBF are robust to maximum estimation error, the raCBF is able to account for this error in a more moderate way, since it assumes parameter estimation is possible.

The second reason raCBF reaches the goal faster is its use of parameter estimation. The parameter adaptation law renders the raCBF *selectively conservative*, whereas the rCBF is always conservative (always operating under worst-case assumptions). To see this, observe Fig. 3.8. The law adapts the parameters toward values that increase conservatism when the state becomes increasingly unsafe. This happens at times $\approx 3, 7, 9$ seconds, when the pole changes direction as the cart oscillates around the goal. This pivot causes the parameter estimate to grow to $\bar{\theta}$, which corresponds to the smallest and thus most conservative invariant set ($\mathcal{I}^r_{\bar{\theta}}$) among the family of invariant sets $\{\mathcal{I}^r_{\hat{\theta}}, \hat{\theta} \in \Theta\}$. On the other hand, when the state is relatively safe, minimal conservatism is applied. This is the case when the pole is static. At those points, the parameter estimate is $\underline{\theta} = 0$, corresponding to the largest invariant set in the family of invariant sets.

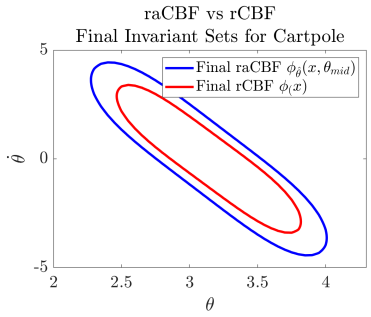


Figure 3.6: Notice that raCBF's invariant set is much larger than the baseline's, which accounts for the difference in the performance test.

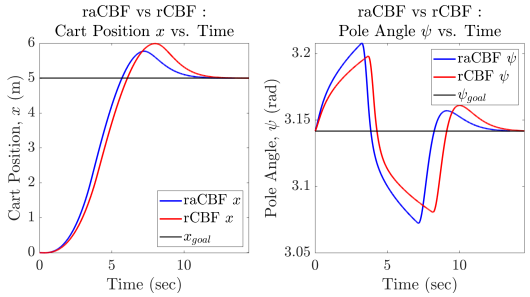


Figure 3.7: Detailed analysis of one trial from cartpole's performance test.

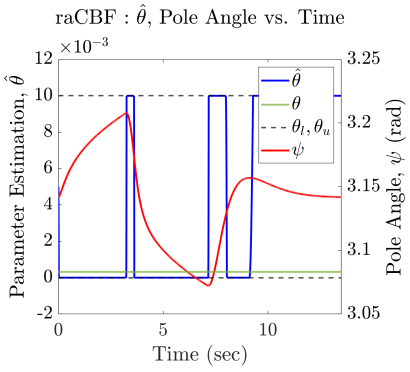


Figure 3.8: Analysis of parameter estimation for cartpole's performance test.

3.6.3 Quadrotor

This system is a planar quadrotor (Fig. 3.1) that we transform from the typical trigonometric dynamics [85] redefining the states as $[\dot{x}, \dot{y}, \dot{\psi}, x, y, \sin(\psi), \cos(\psi) - 1]$. We handle the transformation in much the same way as the cartpole (Sec. 3.6.2). The system has two unknown parameters, which are the drag coefficients in x and y that range from $[0, 0.01]$, roughly following the physical values from [86]. The propeller thrusts are limited to $[-4, 4]$ mg. The safety definition enforces the system to avoid walls on the left and right: $\mathcal{X}_{safe} = \{x \mid x \in [-1, 1]\}$.

Our synthesis is able to handle a system of this moderate size (7D), producing an raCBF that ensures safety in 100% of the randomized trials of the safety test (Table 3.1). The raCBF also has a large invariant set, which is evidenced by how it can track trajectories closely to the wall (Fig. 3.9). On the other hand, the robust CBF from [25] cannot be applied to a system of this size. However, in order to produce some point of comparison, we transfer the key parts of their formulation into our SOSP algorithm. Essentially, this just modifies the control filter feasibility constraints in phases 2 and 3 of our algorithm. This allows us to produce a result after six hours of optimization, which we use in the following performance test.

In the performance test, both controllers attempt to track a trajectory that crosses the walls. The nominal tracking controller is gain-scheduled LQR. As we can see, our raCBF offers an 54.95% improvement over the robust controller. In Fig. 3.9, we note that while our raCBF allows the quadrotor to approach the wall, the robust CBF, which has a much smaller invariant set in the x dimension, restricts the quadrotor to around $x = 0$. We can conclude that raCBF performs better and currently scales better than rCBF. More work will be required to get rCBF to scale moderately.

3.7 Conclusion

In this chapter, we proposed the first verification and synthesis methods for adaptive-type safe controllers for uncertain systems. We proved that the constraints characterizing a valid raCBF can be written as an SOS program, and showed we can devise a multi-phase algorithm to synthesize an raCBF. Then, in our experiments, we illustrated that our algorithm is generic and scalable with three examples with varied dynamics, dimensions (up to 7D), and safety specifications. We empirically confirmed our theoretical safety guarantees by showing that 100% of trajectories under the safe controller are safe. Finally, we showed that our raCBF provides up to $\approx 55\%$ task performance improvement over the baseline. In the future, we are interested in extending synthesis to systems with time-varying unknown parameters.

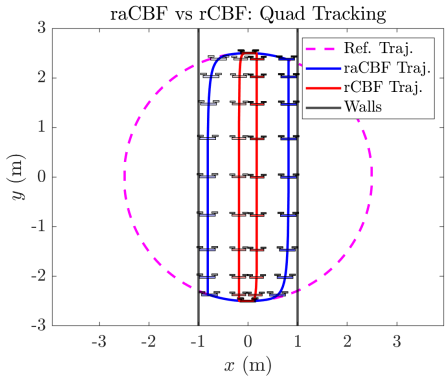


Figure 3.9: raCBF enables closer trajectory tracking than rCBF and is therefore more performant.

4 Adversarial Training of Neural CBF for High-dimensional Systems

4.1 Introduction

In theory, control barrier functions are an appealing tool for safe control. However, it is difficult to make the derived controllers respect input limits, which reduces their usage in practice. CBFs target the *set invariance* class of safety problems, in which safety is defined as keeping a system's state to a prescribed region. A large part of their appeal is that they offer mathematical guarantees of safety. Such assurances are essential for safety-critical robotics applications, like collision-free drone flight [87, 88], manipulators that work safely around humans [89], and stable bipedal walking [90]. However, these safety guarantees break down when input saturation occurs, since that means the system cannot exert the force required for an evasive maneuver. The system then becomes endangered (or dangerous), with the possibility of expensive equipment failure or people getting harmed. It is therefore imperative that we account for input limits when designing CBFs.

CBFs are *energy functions* which map states to an energy value, with safe states having lower energy. In *energy function methods*, an energy function is found and then a controller is crafted that dissipates the energy. The core problem of these methods is constructing the energy function. A valid energy function has to meet complex constraints that depend on the input limits, system dynamics, and safety specification. So far, this problem of designing CBFs around input limits has been studied to a limited degree, mostly for small or simple systems. To our knowledge, nothing has been proposed which handles the nonlinear and high-dimensional systems that are more realistic in robotics. Currently, many state-of-the-art approaches rely on hand-designing CBFs. This works well for certain simple systems, like the kinematic bicycle system [10, 11, 12, 13, 14, 15, 16, 17]. Some of these hand-design methods are more systematic, deriving non-saturating CBF for special *classes* of systems, like polynomial systems [18, 19]. Yet another variation of hand-design is to hand-select a parametric function for the CBF and optimize the parameters to avoid saturation [8, 9, 10]. The problem of designing a non-saturating CBF is also equivalent to computing a *control invariant set*, a well-known problem in the controls community [91]. This area has a long history and has been studied under different viewpoints and names, including viability kernel computation [92] and

infinite-time reachable set computation [93]. For a condensed survey, see [94]. The most generic framework for computing control invariant sets is HJ Reachability [1]. Although it can handle nonlinear systems and gives formal guarantees against saturation, it cannot typically scale past systems of 6 or 7 dimensions in the state. This paper takes a different approach from HJ Reachability and abandons formal guarantees for better scalability.

For the most part, existing approaches for synthesizing non-saturating CBF apply to narrow classes of systems and can involve considerable human effort. In contrast, we envision a framework for *automating* CBF synthesis that has a *wide range of applicability*. To achieve this, we borrow ideas from machine learning (ML). We take inspiration from the related field of Lyapunov function (LF) synthesis, which has incorporated ML with success. LFs certify *stabilization*, rather than safety. However, finding a non-saturating CBF is essentially finding a function that satisfies a constraint on a set of inputs, which is the same problem as in LF synthesis. Recent works in LF synthesis represent the LF as a NN and then train it to satisfy the function constraints [95, 96, 97, 98]. We borrow this paradigm for the unique problem of input saturation of CBFs. For additional related work, please see Appendix Sec. A2.1.

There are several advantages to framing the problem as NN training. Firstly, it allows us to handle synthesis for nonlinear systems. Good non-saturating CBFs for nonlinear systems tend to be nonlinear functions, and NN are a richly expressive class of nonlinear functions. Secondly, it allows us to handle synthesis for systems with large state dimensions ($\geq 10D$). Neural networks can be trained quickly for inputs (here, the system state) of this size.

We synthesize CBFs that respect input limits by posing this as a problem of training a neural function to satisfy limit-related constraints. Our contributions are as follows:

1. A novel way to frame the synthesis of non-saturating CBF.
2. The design of a training framework, including a neural CBF design, loss function and training algorithm design.
3. Experimental validation on a 10D state, 4D input nonlinear system.

The rest of this paper is laid out as follows: Sec. 4.2 explains how CBFs work and carefully define the input saturation problem. Then, Sec. 4.3 details our approach, including the design of the neural CBF, training losses, and training algorithm. Finally, Sec. 5.6 describes how we test our method on a challenging quadcopter-pendulum system against several baselines.

4.2 Preliminaries

In this section, we provide a review of CBFs, mathematically define a non-saturating CBF, and explain the premise of CBF synthesis. First, some notation: for a function $c : \mathbb{R}^n \to \mathbb{R}$, let $\mathcal{C} \triangleq \{c\}_{\leq 0}$ be its zero-sublevel set, $\partial\mathcal{C} = \{c\}_{=0}$ the boundary of this set, and $\text{Int}(\mathcal{C}) = \{c\}_{<0}$ the interior. Now, we assume the following is given: (1) a control-affine system $\dot{x} = f(x) + g(x)u$, where $x \in \mathcal{D} \subset \mathbb{R}^n, u \in \mathcal{U} \subseteq \mathbb{R}^m$ and $f : \mathbb{R}^n \to \mathbb{R}^n$, $g : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ are locally Lipschitz continuous on $\mathbb{R}^n$, (2) input set $\mathcal{U}$, a bounded convex polytope, and (3) a safety specification $\rho : \mathbb{R}^n \to \mathbb{R}$, which

implicitly defines the *allowable set* as $\mathcal{A} \triangleq \{\rho\}_{\leq 0}$. Further, assume $\rho(x)$ is continuous and smooth. Given $\rho(x)$, we can define $r \in \mathbb{Z}^+$ as the *relative degree* from $\rho(x)$ to u (i.e. the first derivative of $\rho(x)$ where u appears).

We now walk through the process of producing a safe controller via CBF methods. In safe control, the goal is to keep some subset of the allowable set *forward invariant*, which means keeping any trajectory starting within the subset inside of it for all time. We call this subset the *safe set* and it will be defined by a function, the control barrier function, which we design. The CBF will also be used to define the safe controller that ensures forward invariance.

In the absence of input limits, we would just form a CBF as a known function of the safety specification $\rho(x)$:

$$\phi = \left[\prod_{i=1}^{r-1}\left(1 + c_i \frac{\partial}{\partial t}\right)\right]\rho \qquad \text{(limit-blind CBF)}$$

where $c_i < 0$. See [11, 99] for an explanation. The safe set $\mathcal{X}_{safe} \subseteq \mathcal{A}$ defined by this limit-blind CBF is elaborated in Appendix Sec. A2.2. In the presence of input limits, we will have to modify this design, but we will come back to this. Next, to define a safe controller using a CBF is straightforward. A safe controller simply needs to repel the system back into the safe set whenever it reaches the set boundary. The following theorem formalizes this idea:

Theorem 4.2.1 (Taken from [11])**.** *Given a CBF ϕ and safe set $\mathcal{X}_{safe}$, any feedback controller $k(x) : \mathbb{R}^n \to \mathbb{R}^m$ satisfying*

$$\dot{\phi}(x, k(x)) \triangleq \underbrace{\nabla\phi(x)^\top f(x)}_{L_f\phi(x)} + \underbrace{\nabla\phi(x)^\top g(x)}_{L_g\phi(x)} k(x) \leq 0 \quad \forall x \in \partial\mathcal{X}_{safe} \tag{4.1}$$

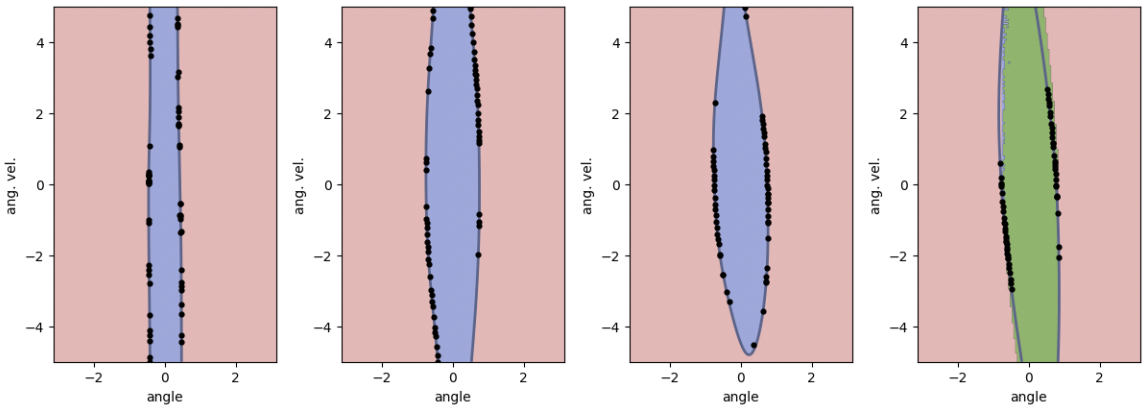
renders the system forward invariant over $\mathcal{X}_{safe}$.

Note that you can also require a CBF to satisfy a stricter inequality $\dot{\phi}(x) \leq -\alpha(\phi(x))$ for all $x \in \mathcal{D}$, where $\alpha(\cdot)$ is a class-κ function. We elaborate on this alternative in Appendix Sec. A2.2. The theorem above requires the controller to repel the system (decrease ϕ) from the boundary ($x \in \partial\mathcal{X}_{safe}$). We are allowed to use any nominal controller, k_{nom}, as long as we modify its inputs to satisfy Eqn. 4.1 at the boundary. Thus, a CBF-based safe controller simply filters (modifies) a nominal controller online by applying the QP below at every step of control execution:

$$k(x) = \underset{u \in \mathcal{U}}{\operatorname{argmin}} \ \frac{1}{2}\left\|u - k_{nom}(x)\right\|_2^2 \qquad \text{(CBF-QP)}$$

$$\text{s.t.} \qquad L_f\phi(x) + L_g\phi(x)u \leq \begin{cases} 0 & \text{if } x \in \partial\mathcal{X}_{safe} \\ \infty & \text{o.w.} \end{cases} \tag{4.2}$$

The issue with using a limit-blind CBF for this controller is that it can cause controller saturation. Specifically, saturation occurs when no $u \in \mathcal{U}$ exists that satisfies Eqn. 4.1 (constraint 4.2), causing



(a) Iter. 0 **(b)** Iter. 70 **(c)** Iter. 140 **(d)** Iter. 750

Figure 4.1: Learned safe sets for the toy cartpole problem at four iterations during training. Candidate counterexamples are marked in black. It is observed that the critic correctly identifies that the states with severe saturation are those with angle and angular velocity of the same sign (angular velocity swinging the pendulum out from the vertical). In (d), green denotes the *largest* non-saturating safe set, computed using MPC. Note that our volume enlargement is so effective that the learned safe set attains 95% of the largest volume.

the loss of safety guarantees.[1] What we need is to synthesize a *non-saturating CBF*, which is mathematically defined as:

Definition 4.2.1 (*Non-saturating CBF*)**.** A function $\phi : \mathbb{R}^n \to \mathbb{R}$ is a non-saturating CBF over a set $\mathcal{X}_{safe}$ if for all $x \in \partial\mathcal{X}_{safe}$:

$$\inf_{u \in \mathcal{U}} \dot{\phi}(x, u) \leq 0 \tag{4.3}$$

Intuitively, Def. 1 just requires that there exist a feasible control input to decrease ϕ (push the system to the interior of $\mathcal{X}_{safe}$) at every state on the boundary, $\partial\mathcal{X}_{safe}$. We approach this problem by modifying the limit-blind CBF:

$$\phi^* = \left[\prod_{i=1}^{r-1}\left(1 + c_i \frac{\partial}{\partial t}\right)\right]\rho - \rho + \rho^* \qquad \text{(modified CBF)}$$

where $\rho^*(x) : \mathbb{R}^n \to \mathbb{R}$ such that $\{\rho^*(x)\}_{\leq 0} \subseteq \{\rho(x)\}_{\leq 0}$. This modification acts to shrink the associated safe set from $\mathcal{X}_{safe}$ to $\mathcal{S}^*$. With a proper choice of function $\rho^*(x)$, we can exclude irrecoverable states (states where no feeasible input preserves safety) from the safe set boundary. In the rest of the paper, we focus on learning the function $\rho^*(x)$ to produce a non-saturating CBF.

Problem scope: to review, our proposed method applies to nonlinear, control-affine systems and safety problems that can be described by a smooth safety specification function $\rho(x)$. We also assume the system is deterministic and fully known. The method also applies to systems of high relative degree, since we based the modified CBF off of a higher-order CBF.

[1] In practice, to avoid an unsolvable QP when saturation occurs, we add a slack variable to Eqn. 4.2. However, we will still violate safety guarantees.

4.3 Methodology

In this section, we present our neural CBF design (Sec. 4.3.1) and then discuss the training framework which optimizes it with respect to control limits (Sec. 4.3.2, 4.3.3). Learning is formulated as a min-max optimization problem of the following form, where θ denotes the parameters of ϕ^*:

$$\min_{\theta} \max_{x \in \partial \mathcal{S}^*} \mathcal{L}(\theta, x) \tag{4.4}$$

We call this loss $\mathcal{L}(\theta, x)$ the *saturation risk*. This min-max problem is solved using a learner-critic algorithm (Alg. 2), where the critic and learner repeatedly find where the worst saturation occurs and then update the CBF to reduce saturation there. We also propose some strategies for training stably, boosting critic efficiency, and for increasing the volume of the safe set. For visualization, we plot the critic's counterexamples for a toy cartpole problem and intuitively justify their correctness (Fig. 4.1).

4.3.1 Design of CBF with Neural Residual

Building upon the previous work discussed in Sec. 4.2, to design a non-saturating CBF, we only need to consider the design of the function $\rho^*(x)$ from the modified CBF. Let $\text{nn} : \mathbb{R}^n \to \mathbb{R}$ be a multilayer perceptron with tanh activations. Then, we define

$$\rho^*(x) = (\text{nn}(x) - \text{nn}(x_e))^2 + \rho(x) \tag{4.5}$$

where x_e is a state, identified by the user, that should belong to any reasonable learned safe set. Specifically, x_e should belong to the allowable set $\mathcal{A}$ and should satisfy $\rho^{(i)}(x_e) \leq 0$ for $i \in [0, r-1]$ (this constrains any possible higher-order components in x_e; for example, velocities should be directed away from unsafe zones). For safety problems that limit the system's distance from an equilibrium, the equilibrium itself should lie in any reasonable safe set. For anti-collision problems, x_e can be a point far from the ego robot. For additional recommendations, see the Appendix. We have designed ρ^* to obey three constraints: *Constraint 1*: ρ^* is smooth. This is required to preserve the smoothness of ϕ^*, which allows us to make existence and uniqueness arguments for the closed-loop system. Our ρ^* satisfies this because nn has smooth tanh activations and also ρ is assumed smooth. *Constraint 2*: The 0-sublevel set of ρ^* is contained within the 0-sublevel set of ρ: $\{\rho^*\}_{\leq 0} \subseteq \{\rho\}_{\leq 0}$. The allowable set can be shrunk but not enlarged; otherwise, dangerous states may be incorporated. Our design meets this criterion because $\rho^* \geq \rho$. *Constraint 3*: $\mathcal{S}^*$ is nonempty, where $\mathcal{S}^*$ is defined by ρ^*. With our design, $x_e \in \mathcal{S}^*$ by the definition of $\mathcal{S}^*$ (see Appendix) and our assumptions on x_e.

4.3.2 Adversarial Training Framework

In the following sections, we present a loss function that encourages satisfaction of Eqn. 4.1 and the algorithm for solving the min-max problem on this loss. First, we define θ as the parameters of

Algorithm 2 Learning non-saturating CBF

function LEARN($\mathbb{X}_{ce}, \theta$)
  Set learning rate α_l
  $\theta \leftarrow \theta - \alpha_l \cdot \nabla_\theta \left[\sum_{x \in \mathbb{X}_{ce}} \text{softmax}(\mathcal{L}(\theta, x)) + \mathcal{R}(\theta)\right]$ {From Eqn. 4.7, 4.6}
  return θ
end function
function COMPUTECE(θ)
  Set learning rate α_c, number of gradient steps N
  $\mathbb{X} \leftarrow$ uniformly sample a set on the boundary {See Alg. 4}
  for i **in** $[0, \ldots, N]$ **do**
    $\mathbb{G} \leftarrow \nabla_{\mathbb{X}} \mathcal{L}(\theta, \mathbb{X})$ {Batch gradient}
    $\mathbb{P} \leftarrow$ project $\mathbb{G}$ along boundary
    $\mathbb{X} \leftarrow \mathbb{X} + \alpha_c \cdot \mathbb{P}$ {Batch update}
    $\mathbb{X} \leftarrow$ project $\mathbb{X}$ to boundary {See Alg. 5}
  end for
  $\mathbb{X}_{ce} \leftarrow$ worst saturating states in $\mathbb{X}$
  return $\mathbb{X}_{ce}$
end function
function MAIN( )
  Input: dynamical system $\dot{x}$, safety specification ρ
  Randomly initialize neural CBF parameters θ
  Repeat:
  $\mathbb{X}_{ce} \leftarrow$ COMPUTECE(θ){$\mathbb{X}_{ce}$ is a set of counterexamples}
  $\theta \leftarrow$ LEARN($\mathbb{X}_{ce}, \theta$)
  Until convergence
  return θ
end function=0

ϕ^*, which include the weights of nn and the c_i coefficients. Our loss function, called *saturation risk*, is defined as:

$$\mathcal{L}(\theta, x) \triangleq \inf_{u \in \mathcal{U}} \dot{\phi}^*_\theta(x, u) \qquad \text{(saturation risk)}$$

It is a measure of the best-case saturation at a given state x. When $\mathcal{L}(\theta, x) \leq 0$, then no saturation occurs at x; when $\mathcal{L}(\theta, x) > 0$, it measures how severe the saturation is. Thus, our min-max problem (Eqn. 4.4) is to minimize the *worst best-case saturation* over the boundary. When the worst best-case is negative, i.e.

$$\max_{x \in \partial \mathcal{S}^*} \mathcal{L}(\theta, x) \leq 0 \qquad \text{(training goal)}$$

then we have successfully found a non-saturating CBF.

To solve the min-max problem on $\mathcal{L}(\theta, x)$ (Eqn. 4.4), we propose a learner-critic algorithm (Alg. 2). Essentially, the algorithm alternates between the critic computing counterexamples (maximization with respect to x) and the learner updating the CBF (minimization with respect to θ). The critic uses projected gradient descent to produce an approximate maximizer, $\hat{x}^*$, and then the learner uses gradient descent to minimize the saturation loss at $\hat{x}^*$. Since both learner and critic perform gradient descent on $\mathcal{L}(\theta, x)$, it is useful to re-express it as an analytic function, rather than a

continuous minimization. To find the analytic expression, observe that this $\mathcal{L}(\theta, x)$ is a minimization over u where the objective is affine (from Eqn. 4.2.1) and the constraint set is a convex polyhedron $\mathcal{U}$, by assumption. This means the minimizing u^* is one of the vertices $v \in \mathcal{V}(\mathcal{U})$ of the constraint set. Thus, $\mathcal{L}(\theta, x)$ can be computed as a discrete minimization, which is analytic:

$$\mathcal{L}(\theta, x) \triangleq \min_{v \in \mathcal{V}(\mathcal{U})} \dot{\phi}^*_\theta(x, v) \qquad \text{(analytic risk)}$$

4.3.3 Encouraging CBF Optimality and Practical Training Methods

Design of objective term for enlarging the safe set: in this section, we introduce a regularization term that we add to the training objective to help enlarge the safe set. One measure of quality for safe sets is size. Since CBFs will allow a system to move freely inside a safe set, a larger safe set provides greater freedom towards accomplishing control objectives. Thus, a larger safe set enables better task performance. To this end, we add a regularization term to the training objective to encourage a large safe set. For context, recall that our CBF ϕ^*_θ defines a safe set $\mathcal{S}^*$. Specifically, there is a function ss^* of ϕ^*_θ that implicitly defines $\mathcal{S}^*$ as its 0-sublevel set: $\mathcal{S}^* = \{x | \text{ss}^*(x) \leq 0\}$); $\text{ss}^*(x)$ is defined in the Appendix. To clearly indicate its dependence on the CBF and its learned parameters θ, we write it as ss^*_θ here. Next, we evaluate the regularization term at some sampled states $\mathbb{X}_{reg}$. We have:

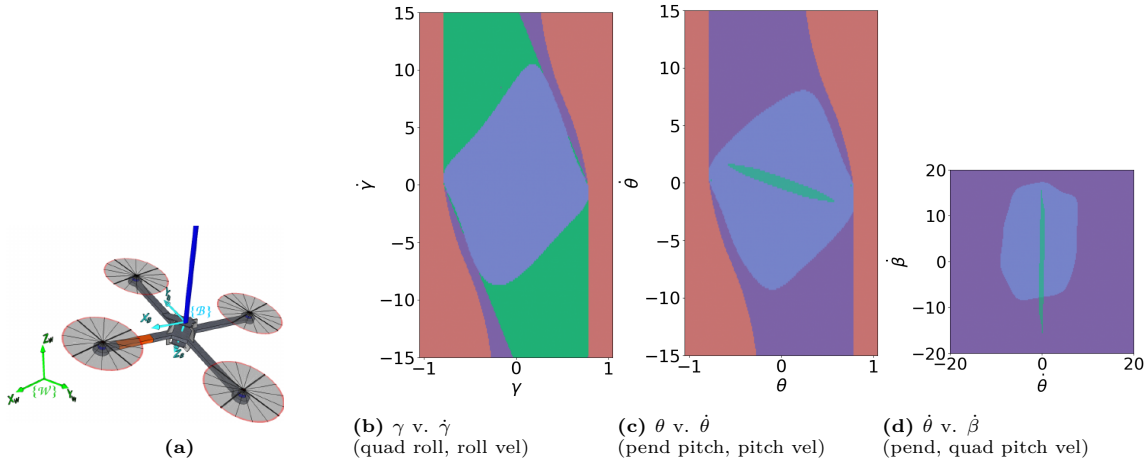
$$\mathcal{R}(\theta) \triangleq \sum_{x \in \mathbb{X}_{reg}} \text{sigmoid}(\text{ss}^*_\theta(x)) \tag{4.6}$$

The idea behind the sigmoid is to encourage states near the boundary ($\text{ss}(\phi^*_\theta(x)) \approx 0$) to move (further) inside the safe set ($\text{ss}(\phi^*_\theta(x)) << 0$). Sigmoid gives a gradient which encourages values near zero to become (more) negative. We demonstrate in the Appendix that including this term can increase the volume by several factors.

Batch optimization: in practice, training works better if both learner and critic use a *batch* of counterexamples, rather than just a single one. This means the critic optimizes a set of differently initialized counterexamples at once and the learner takes a weighted loss over a subset of the best counterexamples:

$$\theta = \theta - \alpha \cdot \nabla_\theta \left[\sum_{x \in \mathbb{X}_{ce}} \text{softmax}(\mathcal{L}(\theta, x)) \right] \tag{4.7}$$

This has the advantage of stabilizing convergence without adding much overhead. It stabilizes convergence by preventing (1) deadlock and (2) inaccurate gradients throughout training. Deadlock is when improvement at one counterexample causes saturation at another; averaging the learner's loss on a group of counterexamples avoids this by encouraging progress on many counterexamples at once. Inaccurate gradients refers to the fact that the learner should be using the gradient at the optimal counterexample x^* ($\nabla_\theta \mathcal{L}(\theta, x^*)$), but since the critic is suboptimal, it uses a different gradient, $\nabla_\theta \mathcal{L}(\theta, \hat{x}^*)$. Clearly, this could derail training. However, we find that with batching, the



(a)

(b) γ v. $\dot{\gamma}$ (quad roll, roll vel)

(c) θ v. $\dot{\theta}$ (pend pitch, pitch vel)

(d) $\dot{\theta}$ v. $\dot{\beta}$ (pend, quad pitch vel)

Figure 4.2: (Left) quadcopter-pendulum system (image from [103]). (Others) Axis-aligned 2D slices of the 10D safe set (blue is ours, purple is hand-designed CBF, green is safe MPC). For each slice, the unvisualized states have been set to 0.

critic produces better counterexamples, giving us a closer estimate of the true gradient ($\nabla_\theta \mathcal{L}(\theta, \hat{x}^*) \approx \nabla_\theta \mathcal{L}(\theta, x^*)$). Plus, with averaging, the learner averages out the effect of inaccurate gradients. We show in the Appendix that the algorithm requires large batches of counterexamples to perform well (Table A2.1). All in all, these techniques are a cheap and effective way to avoid the kind of training instability found in other counterexample-based methods, like adversarial training for image classifiers [100, 101, 102].

4.4 Experiments

In this section, we train a neural CBF on a challenging nonlinear, high-dimensional, and input-limited robotic system. We pose two experimental questions:

Q1. How well do we achieve our training objective?

Q2. Does the CBF-based safe controller ensure FI?

Our system is a pendulum on top of a quadcopter (Fig. 4.2). This is a coupled system, with the dynamics of both components found by the Euler-Lagrangian method [104]. The states are quadcopter position and roll-pitch-yaw orientation (x, y, z and γ, α, β) and roll-pitch pendulum orientation (ϕ_p, θ_p), as well as the first derivatives of these states. The inputs are thrust and torque ($F, \tau_\gamma, \tau_\beta, \tau_\alpha$), which are limited to a bounded convex polytope set. See the Appendix for the system dynamics. The safety specification is to prevent the pendulum from tipping and the quadcopter from rolling:

$$\rho = \gamma^2 + \beta^2 + \delta_p^2 - (\pi 4)^2 \tag{4.8}$$

where $\delta_p = \arccos(\cos(\phi_p)\cos(\theta_p))$ is the pendulum's angle from the vertical. Since the quadcopter position does not impact safety and the position dynamics can be decoupled, we exclude position

	Saturation at the boundary			Safety of rollouts w/ diff k_{nom}			Safe set volume
	% non-sat. states	mean, std dev of sat.	worst sat.	k_0	k_{lqr}	$k_{lqr-agg}$	(as fraction of ours)
Ours	99.00	1.75 ± 2.40	15.19	99.62	99.62	99.02	1.00
Hand-designed CBF	78.68	4.99 ± 3.78	29.50	78.68	80.28	80.28	53.87
Safe MPC	-	-	-	99.06	99.54	99.44	0.08

Table 4.1: Comparison of our method against baselines. The "mean, std dev of sat." is $\mathbb{E}[\mathcal{L}(\theta, x)] \pm \sigma[\mathcal{L}(\theta, x)]$ and "worst sat." is $\mathcal{L}(\theta, x^*)$.

and consider the resulting 10D state, 4D input system. Finally, in the design of $\rho^*(x)$, we let $x_e = \vec{0}$, which is the system's equilibrium.

Next, we propose metrics to answer each of our experimental questions:

Q1 metrics. *% of non-saturating states on $\partial\mathcal{S}^*$*: we measure how well we satisfy Eqn. 4.3 by uniformly sampling states on $\partial\mathcal{S}^*$ and calculating what percentage of them are non-saturating. We also use these samples to compute the *mean and variance of the severity of saturation*, $\mathbb{E}[\mathcal{L}(\theta, x)]$ and $\sigma[\mathcal{L}(\theta, x)]$. To approximate the *severity of the worst saturation*, $\mathcal{L}(\theta, x^*)$, we apply our critic and allow it to use more samples and computation time than during CBF training.

Q2 metrics. Q2 considers the in-the-loop control performance of our learned CBF. We measure *% of simulated rollouts that are FI* by initializing rollouts randomly inside $\mathcal{S}^*$ and simulating their trajectories under the safe controller (CBF-QP) until just after they reach the boundary. Then, we record whether the system exited or remained inside $\mathcal{S}^*$ after arriving at the boundary. The value of this metric depends on the choice of nominal controller, k_{nom}. We try $k_0(x) = 0$, which yields an unactuated system, and k_{lqr} and $k_{lqr-agg}$, which are linear quadratic regulator (LQR) stabilizing controllers, with $k_{lqr-agg}$ tuned to be more aggressive.

We also choose two well-known alternatives as our baselines:

Hand-designed CBF: for CBFs, this is a typical alternative. We hand-pick a parametric CBF and then optimize the parameters for non-saturation.

Safe MPC: MPC is often used for safe planning and control and it can take input limits into account. The safety specification ($\rho \leq 0$) becomes a nonlinear state constraint and we also have to set the terminal constraint to be a smaller, known invariant set for the MPC solution to be forward invariant.

For details on any of these baselines, see the Appendix Sec A2.4. Note that safe MPC defines its safe set *implicitly*. This means the boundary of the safe set is not defined by a function. Hence, it would be too time-consuming to sample states on the boundary and compute the metrics for Q1. For safe MPC, we only report the results for Q2.

Code. Our code can be found at `https://github.com/sliu2019/input_limit_cbf`

Training details. The learning framework and metrics were implemented using Python and PyTorch [105]. Training took about 2 hours on a single NVIDIA GeForce RTX 2080 Ti GPU.

Discussion. As we can see in Table 4.1, for our learned CBF, almost 100% of the boundary states are non-saturating and almost 100% of the rollouts are FI across the nominal policies. This shows that our neural CBF and learning framework are an effective combination and capable of handling systems of high complexity. We are not able to attain 100% non-saturating states and FI rollouts, which is either due to suboptimality of training or limitations of the function class,

as NNs are only universal approximators when their size is taken to the infinite. For our learned CBF, the severity of saturation at the saturating states is not negligible, but it is still low, and the worst saturation is large, but rarely encountered. The hand-designed CBF does poorly across all metrics (only 80% of the boundary states are non-saturating and around 80% of rollouts are FI). Safe MPC is equally good as the learned CBF at ensuring safety (almost 100% rollout safety)[2]. However, its significant disadvantage is that its safe sets are just a fraction of the size of our own (8.4% respectively). Size is an important measure of the quality of a safe set; these small sizes imply that this baseline can only ensure safety from relatively few states. The root of the issue is that safe MPC constructs a safe set by effectively expanding a smaller, "seed" invariant set provided by the user. The size of the safe set therefore depends greatly on the size of the seed invariant set. As we have already established, it can be quite difficult for users to design invariant sets (equivalently, non-saturating CBFs) of any reasonable size.

Visualizing slices of the safe set can provide deeper insight into the results of training (Fig. 4.2). Recall that a safe set contains only states that are *recoverable* from danger, given our input limits. We observe that our CBF has diamond-shaped safe sets in slices B and C, which makes sense because small angles can still be recoverable at larger speeds. In slice B, safe MPC's set largely captures states where the signs of $\gamma, \dot{\gamma}$ differ. This makes sense too, since these are states where the angular velocity acts to return the angle to 0. Safe MPC's set in slice B is also larger than ours. Since it is a non-saturating safe set, just like ours, we have to conclude that our algorithm could have found a larger non-saturating safe set. The reason for this suboptimality is that our volume regularization strategy is imperfect. It encourages expansion only at scattered points on the boundary, which means that some areas of the boundary may be unaffected. However, our regularization strategy seems to work well overall, since the volume of our safe set in 10D is much larger than that of safe MPC. In slices C and D, we get a glimpse of why that is. In both of these slices, safe MPC's set is tiny. It very tightly constraints the pendulum's angular velocity. While it makes sense for safe MPC to be more conservative towards the pendulum than the quadcopter (the pendulum is not directly actuated, while the quadcopter is), it is unnecessarily conservative.

Next, we analyze the shapes of the sets in slice D. Our safe set indicates that there should be a maximum safe angular speed for quadcopter and pendulum. This makes sense, since higher angular velocities are certainly harder to pull back from. We also observe a larger range for the quadcopter's pitch velocity than the pendulum's, which makes sense because again, the quadcopter is directly actuated and the pendulum is not. On the other hand, the hand-designed CBF does not restrict $\dot{\theta}$ or $\dot{\beta}$ in slice D. Due to the functional form of the hand-designed CBF (see Appendix Section A2.4), $\dot{\theta}$ and $\dot{\beta}$ are only restricted when $\dot{\theta}\theta > 0$ or $\dot{\beta}\beta > 0$ (that is, when angular velocity is strictly acting to destabilize). We've assumed $\theta, \beta = 0$ in slice D. This safety criterion is clearly too lenient, since large angular velocities that are currently swinging the system to the origin can cause overshooting and toppling shortly after. Overall, we can see that the non-neural CBF does not have the right function form. In general, it can be hard to guess what the right form might be for a system of this

[2]In theory, MPC should attain perfect safety. However, nonlinear MPC solvers are not "complete" in the sense that even if there exists a solution to their nonlinear program, they may not find it. Hence, sometimes they may falsely consider the safety problem infeasible and provide an unsafe input.

complexity. However, our algorithmic approach, combined with the NN functional representation, removes the need for guessing.

Limitations and future work: in the future, we intend to loosen our assumptions (see last paragraph of Sec. 4.2), particularly the assumption of a known and deterministic model. We would also like to test this method on more high-dimensional, nonlinear systems. One limitation of our work is that we had to perform a change of variables on the states input to the neural CBF before it would train successfully (see Appendix for details). While the dynamics on the new states were still nonlinear, this indicates that a simple feed-forward network might not be the best neural function class.

4.5 Conclusion

In summary, we proposed a framework for facilitating CBF synthesis under input limits. Thanks to our neural CBF representation and our effective and efficient learning framework, our method scales to higher-dimensional nonlinear systems. We learned a virtually non-saturating CBF on one such system, the quadcopter-pendulum. We hope that this safe control tool will makes CBFs more accessible and of practical value to roboticists.

II

Motion Planning for Contact-Rich Manipulation

Introduction to Part II

Part I argued that reachable sets are a unifying computational primitive for safety and performance in agile systems. In Part II, we shift that lens to performance-driven planning for contact-rich manipulation, where the challenge is not avoiding imminent violations but finding high-quality, feasible motion through hybrid state spaces. Contact-rich tasks inherently couple discrete mode changes (which contacts are active) with continuous motion (how the object and arms move), yielding a search space that is high-dimensional, nonsmooth, and riddled with local minima. Standard approaches split along two extremes: local trajectory optimizers that are fast but myopic and highly initialization-sensitive, and global methods (sampling or MIPs) that explore more broadly but often return circuitous plans or fail to scale. This section develops a middle path that retains global structure without sacrificing solution quality.

Our key idea is to make reachable sets the unit of abstraction for global planning. Concretely, we introduce Mutual Reachable Sets (MRS)—the set of object poses in SE(2) that are both forward- and backward-reachable in finite time from a seed grasp under a competent local contact-rich planner (CQDC-MPC). Because modern local planners can traverse multiple contact modes in one shot, each MRS compactly summarizes a large swath of feasible, multi-contact behaviors. We then (i) build an offline graph of MRS that covers the object space, (ii) plan online in object space by finding near-optimal paths through this graph via the Graph of Convex Sets (GCS) relaxation (yielding bounded suboptimality), and (iii) lift the object path back to full robot inputs by tracking with the same local planner and inserting regrasps only where sets intersect. On a demanding bimanual KUKA iiwa reorientation problem in SE(2), a small cover (e.g., 36 sets) suffices to collapse the hybrid search to a short path on a sparse graph, delivering seconds-level query times, high success rates, and substantially lower execution cost than a state-of-the-art sampling baseline—all while preserving the expressiveness needed for fine, multi-contact maneuvers.

5 Global Planning for Bimanual SE(2) Manipulation on a Graph of Reachable Sets

5.1 Introduction

Robots today are not yet leveraging their full physical potential. Most manipulation relies solely on end effectors, but without engaging other parts of the manipulator, tasks involving large, heavy objects or small, delicate ones become nearly impossible. To effectively handle these long-tail objects, we often need contact-rich interactions with the manipulator or environment. For example, for large, heavy objects, bracing can allow greater wrench application and more efficient, expressive motion. For small objects, multi-point support against the manipulator can provide stability during fine, precise actions. In order to fully exploit these manipulator capabilities, we must reason explicitly about contact.

Planning for contact-rich manipulation remains an open challenge, as it involves planning for systems that are both hybrid and underactuated. The discrete portion is a combinatorial search over the sequence of contact states or modes. Even for SE(2) tasks, this search branching factor can be significant: between 4^{10} and 4^{20} contact modes for the system we study here. The continuous portion, which considers motion within a fixed contact mode, is a nonconvex optimization problem. The joint hybrid search space is high-dimensional, nonsmooth, and nonconvex, making it difficult to find high-quality trajectories. Namely, local planners will often get trapped in local minima, and global planners tend to produce circuitous trajectories.

We also consider a challenging manipulation system: a bimanual KUKA iiwa robot (Fig. 5.1) that reorients objects in the plane. This system has high dimensionality and requires reasoning about multi-point contact involving complex geometries, as well as kinematic feasibility. It is significantly more complex than the systems typically studied in model-based planning, like those involving point contacts or parallel-jaw grippers.

We introduce a hierarchical, multi-query planner for full-body manipulation in SE(2). In our offline phase, we build a graph of mutual reachable sets in object space. A *mutual reachable set* comprises object states forward and backward reachable in finite time from a starting grasp. Since

current local planners [45] can generate far-reaching trajectories that span several contact modes, their reachable sets are a much more tractable discrete unit than contact modes themselves (Fig. 5.2). Relatively few of such sets can cover key regions of the search space, enabling efficient search over chains of local policies.

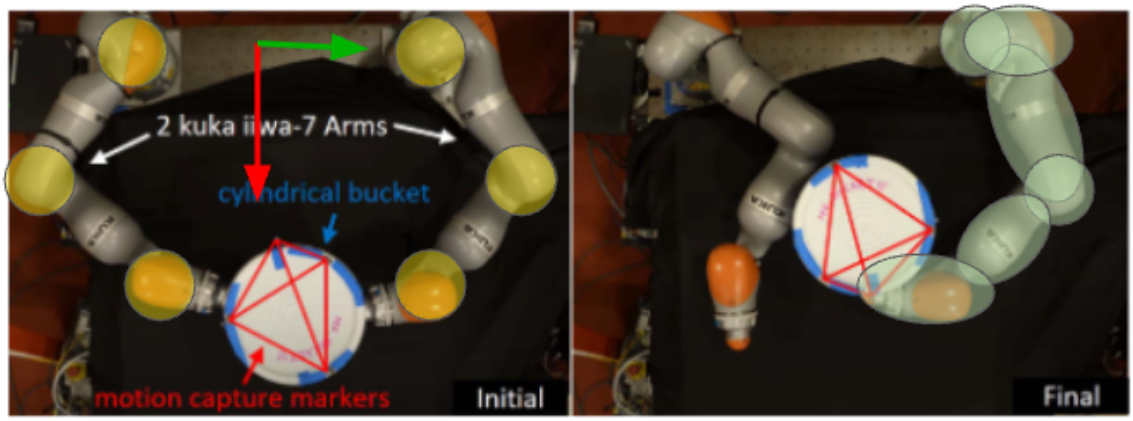


Figure 5.1: Left: Bimanual KUKA iiwa-7 setup with the three actuated joints on each arm (shoulder–elbow–wrist) highlighted. Right: collision model used for planning, which 29 spheres per arm.

Besides reducing the complexity of the discrete search space, our approach also reduces the dimensionality of the continuous search space. At runtime, our planner operates in two layers: (1) a top level that generates an object-space trajectory through the graph of reachable sets and (2) a lower level that computes the corresponding trajectory in the full manipulator and object configuration space. At the top level, which is the most time-consuming, we plan in the low-dimensional object subspace. We design the framework so that object space contains all the required information to plan - including information that ensures kinematic and dynamic feasibility at the lower-level planner.

Taken together, our design choices drastically shrink the search space and allow us to apply bounded-suboptimal search techniques to find high-quality solutions. This framework provides several advantages:

- Bounded suboptimality, up to the object-space cover
- Trajectories have orders of magnitude lower objective cost than state-of-the-art planners
- Online solutions in seconds
- High planning success across broad distribution of queries

5.2 Related Works

Reinforcement learning (RL) methods have shown early success in dexterous manipulation, but they tend to be highly sample-inefficient and require extensive reward shaping [26, 27, 28, 29, 30,

31, 32]. Additionally, while behavioral cloning (BC) has been effective for general real-world manipulation [106, 107, 108], it has been limited to all but the simplest contact-rich tasks, like end-effector pushing [33, 34]. This is because it is impractical to collect the necessary teleoperation data - the embodiment gap makes intricate contact-rich tasks hard to complete via teleoperation. Initial works have instead begun to investigate generating such data using planners like ours [109, 110]. One of their key findings is that the success of learning depends strongly on the quality of the planner-generated data, which is an issue we address in this work.

Model-based approaches can be divided into local and global methods. We first review local methods, which perform optimization within the hybrid space by treating it as piecewise nonlinear. One main type of local method employs contact-implicit models, which frame contact dynamics as constrained optimization problems [35, 36, 37, 38, 39]. These models can be incorporated into gradient-based trajectory optimizers [39, 45]. The second major body of local methods leverages differentiable simulators, which approximate dynamics via Taylor expansions [40, 41, 42, 43, 44, 45]. These can be used within gradient-free trajectory optimizers [46, 47]. Overall, though local methods tend to be fast and scalable, the feasibility and quality of their solutions depends heavily on having a "good" initial guess.

Next, we summarize global methods, which fall into three main categories. The first is sampling-based approaches [45, 48, 49, 50, 51, 111], which alternate between discrete search and continuous optimization. Recent works have been able to tackle complex problems, like bimanual manipulation or dexterous reorientation [45]. However, these methods generally lack asymptotic optimality guarantees and also perform poorly in practice, as their random growth strategy leads to inefficient trajectories. Post-processing with shortcutting or trajectory optimization provides only marginal improvements, as it does not address the underlying exploration bias. Some sampling-based works also incorporate local reachable sets, using them to guide sampling [45, 48, 112, 113], but this only improves efficiency, not optimality.

The second category is mixed-integer programming (MIP) methods [52, 53, 54], which encode contact modes as binary variables and search over mode sequences. These methods scale poorly, with runtime growing exponentially in the number of modes, and are typically limited to simple 2D manipulation tasks (e.g., gripper-and-cube or push-T). Solving convex relaxations of the MIP can improve scalability [54], but only somewhat.

The third class consists of hierarchical planners, which aim to reduce the combinatorial complexity of discrete search through abstraction. The top layer of the hierarchy can range from low-level contact mode enumeration [51, 114, 115, 116] to high-level, hand-designed motion primitives [117, 118, 119]. We argue that the structure of the hierarchy is critical. Too low-level, and planning becomes slow: for example, [51, 116] do not scale past point contact and gripper systems. On the other hand, if the hierarchy is too high-level, then planner becomes rigid, incapable of producing fine-grained motions [119]. To our knowledge, no prior work strikes the right balance between efficiency and expressiveness.

Finally, in feedback control, there are works that construct global feedback policies by chaining locally stabilizing controllers [120, 121], but these primarily focus on smooth systems.

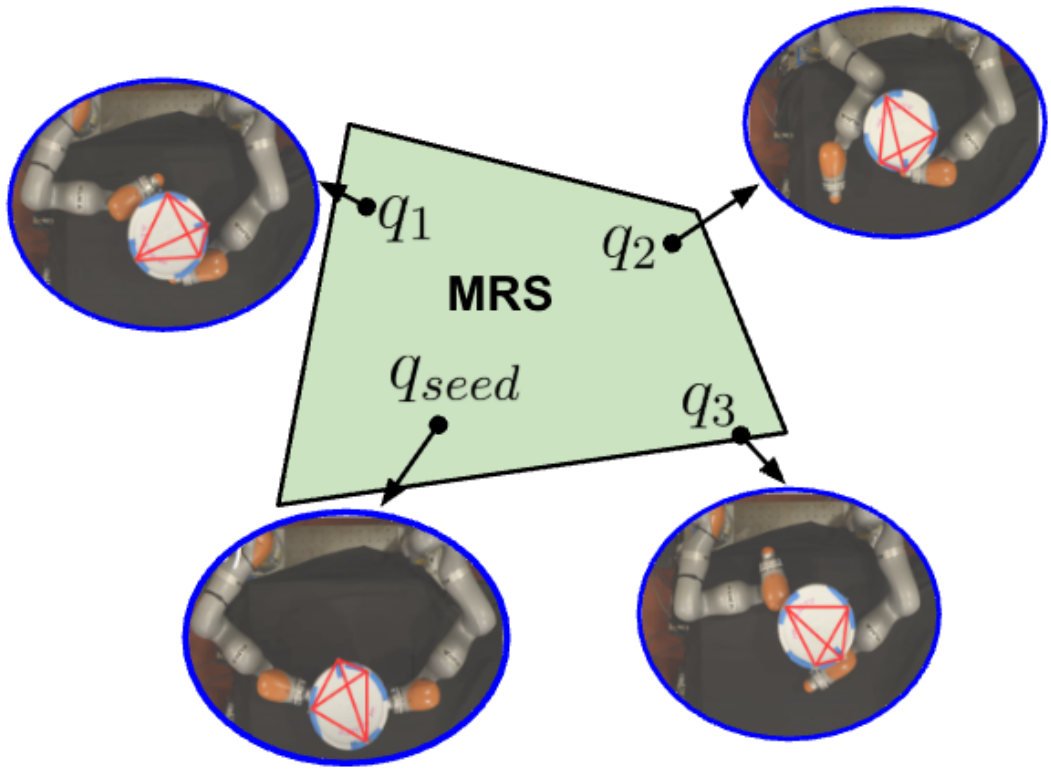


Figure 5.2: Example of MRS showing different configurations (q_1, q_2, q_3) reachable from an initial grasp q_{seed} via a local planner. Note especially how the MRS encapsulates multiple different contact states/modes (q_1, q_2, q_{seed} all have distinct active contact pairs). This is what makes MRS a useful discrete unit in planning - it abstracts away some of the combinatorial complexity of contact states/modes.

5.3 Preliminaries

We begin by specifying our problem setting, defining mutually reachable sets, and explaining two tools that we leverage: CQDC-MPC, a local planner for contact-rich problems, and GCS, a framework that can be used to find shortest paths on graphs of sets.

5.3.1 Problem Formulation

We consider a bimanual KUKA iiwa system (Fig. 5.1), where each arm has 7 joints but only 3 (shoulder, elbow, wrist) are actuated to constrain motion in the xy-plane. The task is to move a cylindrical object to a desired pose. The system has 3 unactuated DOFs, 6 actuated DOFs, and 29 collision geometries. We assume known geometric models of both the robots and the object.

We describe the system using configurations $q \in \mathcal{Q} \subseteq \mathbb{R}^6 \times \mathrm{SE}(2)$. We omit velocities due the quasidynamic assumption of CQDC, our planning model (Sec. 5.3.3). The configurations are divided into the manipulator's actuated configurations $q^a \in \mathcal{Q}^a \subseteq \mathbb{R}^6$ and unactuated object configurations $q^o \coloneqq (q^o_x, q^o_y, q^o_\theta) \in \mathcal{Q}^o \subseteq \mathrm{SE}(2)$. The actions are position commands $u \in \mathcal{U} \subseteq \mathbb{R}^6$ that are tracked with a stiffness controller with diagonal gain matrix $\mathbf{K}_a \in \mathbb{R}^{6\times 6}$. We model our system as a discrete-

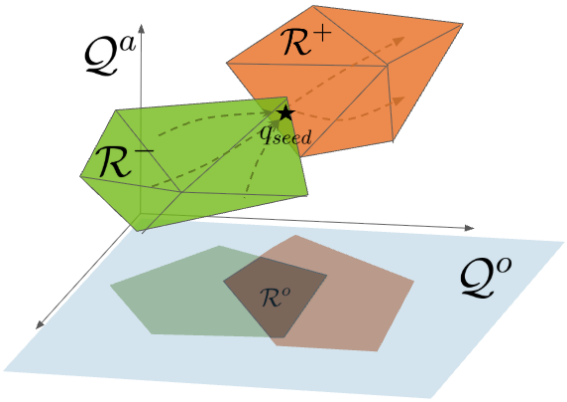


Figure 5.3: Figure showing the relationship between FRS $\mathcal{R}^+ \in \mathcal{Q}$, BRS $\mathcal{R}^- \in \mathcal{Q}$, and MRS FRS $\mathcal{R}^o \in \mathcal{Q}^o$.

time dynamical system:

$$q_{t+1} = f(q_t, u) \tag{5.1}$$

For convenience, we define $f_T : \mathcal{Q} \times \mathcal{U}^T \to \mathcal{Q}$, which recursively applies dynamics for a length-T control trajectory $u_{0:T-1}$ from some initial state q.

We use two distance functions. To describe differences in robot configuration, we use the standard ℓ^2 norm. For differences in object configuration, we use $d_{\mathrm{SE}(2)}$, which denotes a weighed Euclidean norm:

$$d_{\mathrm{SE}(2)}(q_1^o, q_2^o) = \sqrt{\|p_1 - p_2\|^2 + w((q_{1,\theta}^o - q_{2,\theta}^o) \bmod 2\pi)^2} \tag{5.2}$$

where $p_i = (q_{i,x}^o, q_{i,y}^o)$ for $i = 1, 2$. We set $w = 1$.

Now we are ready to describe our problem. Given an initial system configuration q_{start}, goal object configuration q_{goal}^o, and tolerance $r \in \mathbb{R}^+$, our goal is to produce an action sequence $u_{0:T-1}$ which, when applied to q_{start}, produces a configuration sequence $q_{0:T}$ with $d_{\mathrm{SE}(2)}(q_T^o, q_{\mathrm{goal}}^o) \leq r$. We would like this action sequence to *minimize* the following cost function

$$c\left(q_{0:T}, u_{0:T-1}\right) = \sum_{t=0}^{T-1} d\left(q_t, q_{t+1}\right) \tag{5.3}$$

where $d(q_t, q_{t+1}) = \|q_t^a - q_{t+1}^a\|$.

5.3.2 Mutual Reachable Sets

We define mutual reachable sets (MRS), the basis of our framework. An MRS consists of all object configurations that are forward and backward reachable from a *seed grasp* (a configuration $q_{seed} \in \mathcal{Q}$ where the manipulator and object are in contact) under a local planner. Each set can be thought of as the "coverage region" of the local planner. The core innovation of our framework is that we consider our MRS as the discrete elements, rather than contact modes or higher-level skills

(e.g. rotating, pivoting). This choice of discretization reduces combinatorial complexity while still remaining flexible enough to solve any reorientation queries. Later, in the methodology section, we will describe how we can cover the object configuration space $\mathcal{Q}^o$ with a collection of MRS and link them into a graph to plan through.

First, we begin by defining *forward* and *backward reachable sets* (FRS, BRS). Given a seed configuration q_{seed}, a maximum horizon T, a local planner which produces a control trajectory $\pi(q^o_{start}, q^o_{goal}) \to u_{0:T-1}$, the finite-time FRS is the set of all configurations that are reachable from q_{seed} under π:

$$\mathcal{R}^{+}_{\pi,T}(q_{seed}) = \{q \mid q = f_T(q_{seed}, \pi(q^o_{seed}, q^o))\} \subseteq \mathcal{Q}. \tag{5.4}$$

Analogously, the finite-time BRS is all configurations that can reach to q_{seed} under π:

$$\mathcal{R}^{-}_{\pi,T}(q_{seed}) = \{q \mid f_T(q, \pi(q^o, q^o_{seed})) = q_{seed}\} \subseteq \mathcal{Q}. \tag{5.5}$$

Though the FRS and BRS are defined on full configuration space $\mathcal{Q}$, we are interested in their coordinate projections onto $\mathcal{Q}^o$, which we denote by a superscripted o:

$$\mathcal{R}^{o,+/-}_{\pi,T}(q_{seed}) := \mathbf{proj}_{\mathrm{SE}(2)}\left(\mathcal{R}^{+/-}_{\pi,T}(q_{seed})\right). \tag{5.6}$$

We are now ready to define the mutually reachable set.

Definition 1. **Mutually reachable set (MRS)** - given a seed configuration q_{seed}, planner π, and maximum horizon T, the MRS is all object configurations that simultaneously can be reached from and can reach back to the seed in $2T$.

$$\mathcal{R}^{o}_{\pi,T}(q_{seed}) := \mathcal{R}^{o,+}_{\pi,T}(q_{seed}) \cap \mathcal{R}^{o,-}_{\pi,T}(q_{seed}) \subseteq \mathcal{Q}^o \tag{5.7}$$

See also Fig. 5.3 for a visualization. For brevity, we refer to these sets from here on out as $\mathcal{R}^o, \mathcal{R}^{o,+/-}, \mathcal{R}^{+/-}$, dropping the dependency on q_{seed}, π, T.

Lemma 5.3.1. *Any two object states q^o_1, q^o_2 in the MRS are mutually reachable: q^o_1 is reachable from q^o_2 and vice versa.*

We will use this lemma later to show that our object plans are feasible by construction, under some assumptions. Note that because we focus on prehensile manipulation, the FRS and BRS will tend to overlap significantly, giving us sizable MRS.

5.3.3 Local Planner: CQDC-MPC

Our framework is agnostic to the choice of local planner $\pi(q^o_{start}, q^o_{goal}) \to u_{0:T-1}$, so we just choose one with useful traits. As mentioned in the prior section, we will use π to construct the MRS offline. We will also use it online at the second stage of our hierarchical planner to produce manipulator control inputs given an object plan.

Our local planner consists of a contact-implicit dynamics model that is incorporated into a trajectory optimization program. This program is solved and applied MPC-style. The specific dynamics model we chose is called the Convex Quasidynamic Differentiable Contact (CQDC) model. It assumes a quasidynamic, implicit time-stepping model [45].

When used for control, this model is locally smoothed and then linearized. This linearized model is also constrained to a trust region [122]. Then, the one-step forward dynamics are computed as the solution of a convex program. The CQDC model offers several benefits. For one, the smoothing that precedes linearization makes the model numerically robust when used with gradient-based trajectory optimizers. This smoothing also enables contact discovery through gradients, though non-convexity prevents gradient-based optimization from effectively searching through all modes. For another, the quasidynamic assumption leads to more efficient trajectory optimization, since it predicts long-term behavior.

$$\text{CQDC-TrajOpt}(q^o_{start}, q^o_{goal}, q^{ref}_{0:T}) = \delta u^*_{0:T-1}, \text{where} \tag{5.8a}$$

$$\min_{\delta q_{0:T}, \delta u_{0:T-1}} \quad c(q_{0:T}, q^{ref}_{0:T}, q^o_{goal}) \tag{5.8b}$$

$$\text{s.t.} \quad \delta q_0 = 0 \tag{5.8c}$$

$$\text{and for} \quad t = 0, \dots, T: \tag{5.8d}$$

$$q_t = q^{ref}_t + \delta q_t \tag{5.8e}$$

$$u_t = q^{ref}_t + \delta u_t \tag{5.8f}$$

$$\mathbf{A}_{\kappa,t} := \partial f_\kappa / \partial q(q^{ref}_t, u^{ref}_t) \tag{5.8g}$$

$$\mathbf{B}_{\kappa,t} := \partial f_\kappa / \partial u(q^{ref}_t, u^{ref}_t) \tag{5.8h}$$

$$\delta q_{t+1} = \mathbf{A}_{\kappa,t} \delta q_t + \mathbf{B}_{\kappa,t} \delta u_t \tag{5.8i}$$

$$(\delta q_t, \delta u_t) \in \tilde{\mathcal{S}}_{\mathbf{\Sigma},\kappa}(q^{ref}_t, u^{ref}_t) \tag{5.8j}$$

To generate trajectories, we apply CQDC-TrajOpt in an MPC fashion (CQDC-MPC). We find empirically that CQDC-MPC is more likely to find feasible solutions if we use a batch of different reference trajectories, $\{q^{ref,i}_{0:T} \mid \forall i \in \mathbb{I}\}$ with $\mathbb{I}$ an index set. Following [123], we design the reference trajectories as two-segment paths with varying midpoints.

5.3.4 Shortest Paths on Graph of Convex Sets

Using our local planner, we cover the object space with MRS and then connect them into a graph. At query-time, our hierarchical planner first searches for the shortest path through the MRS graph, yielding an object path, $q^o_{0:T}$. For this, we apply the Graph of Convex Sets (GCS) framework [124], an efficient approach to solving the Shortest Path Problem (SPP) in a graph of convex sets. It has primarily been used for collision-free motion planning [125], but has also been used for contact-rich manipulation planning [54] and skill composition [126].

We define the problem setting formally as follows: we have a directed graph $G = (\mathcal{V}, \mathcal{E})$, with

vertex set $\mathcal{V}$ and edge set $\mathcal{E} \subseteq \mathcal{V}^2$. Each vertex $v \in \mathcal{V}$ is associated with a state $x_v \in \mathcal{X}_v$, where $\mathcal{X}_v$ is a convex set. Each edge $e := (u, v) \in \mathcal{E}$ is associated with a nonnegative convex cost $\ell_e : \mathcal{X}_u \times \mathcal{X}_v \to \mathbb{R}^+$. Next, we define a path p as a sequence of distinct vertices that connect a source $s \in \mathcal{V}$ to a target $t \in \mathcal{V}$. Let $\mathcal{P}$ denote the family of all s to t paths in the graph, and let $\mathcal{E}_p$ denote the set of edges traversed by p. Since the SPP involves both discrete (choosing which sets to traverse) and continuous (designing a trajectory within each set) components, it is naturally transcribed into the following mixed-integer convex program (MICP):

$$\min \sum_{e:=(u,v)\in\mathcal{E}_p} \ell_e(x_u, x_v) \tag{5.9a}$$

$$\text{s.t. } p \in \mathcal{P} \tag{5.9b}$$

$$x_v \in \mathcal{X}_v, \forall v \in p \tag{5.9c}$$

While this problem can be solved to global optimality using MIP, the GCS framework introduces a tight convex relaxation that can be solved much faster. The solution to this relaxation is bounded-suboptimal to the solution of the original MIP, where the bound can easily be computed. We find this optimality gap to be easily computed (Sec. 5.6). This means that it is sensible to solve the faster relaxation rather than the MIP, and also that our object paths are nearly globally optimal up to our assumptions.

5.4 Methodology - Offline Graph Construction

As typical of multi-query planners, we begin by constructing a reusable graph offline. This section outlines that process: we first compute a convex approximation of the MRS from a seed grasp q_{seed}, then use this subroutine to cover object space with sets. We describe how these sets are connected into a planning graph—defining edges, costs, and constraints—and conclude with guarantees for object-space planning over this graph.

5.4.1 Computing a Convex-Approximated MRS

Given a seed grasp q_{seed}, we construct its mutual reachable set by intersecting the projected forward and backward reachable sets induced by CQDC-MPC. We discretize the object space into a grid with resolution Δ and represent the FRS and BRS as binary occupancy maps. To compute the projected FRS, $\mathcal{R}_\Delta^{o,+}$, we attempt to reach each grid cell from q_{seed}. Likewise, to compute the projected BRS, $\mathcal{R}_\Delta^{o,-}$, we attempt to reach q_{seed} from each grid cell. The MRS is then obtained by their intersection: $\mathcal{R}_\Delta^o = \mathcal{R}_\Delta^{o,+} \cap \mathcal{R}_\Delta^{o,-}$.

Since the MRS is generally non-convex, we need to approximate it with a convex set $\hat{\mathcal{R}}_\Delta^o$ to use it within the GCS framework. We apply the lightweight IRIS-ZO algorithm [127], which inflates a convex polytope inside an implicitly defined non-convex region. IRIS-ZO iteratively adds separating hyperplanes until the volume violating the exclusion region falls below a threshold. We can use

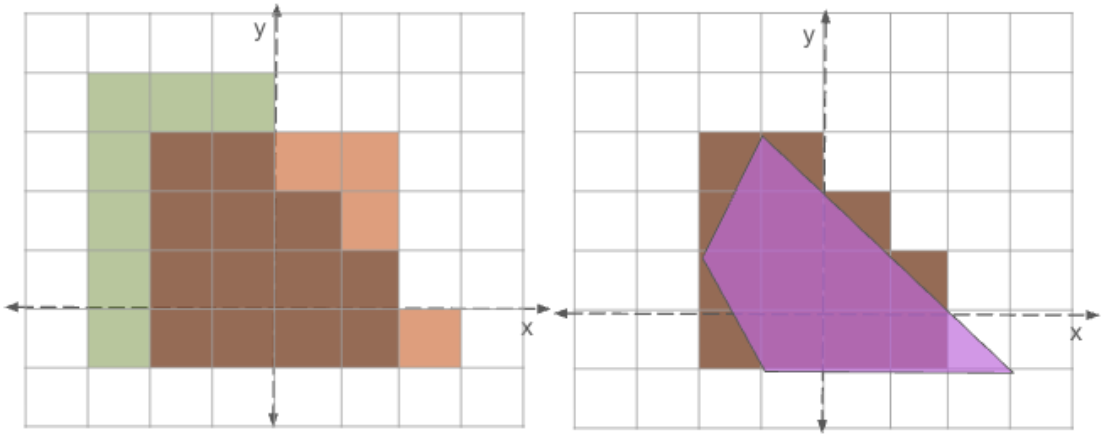


Figure 5.4: Illustration of computing a convex-approximated MRS in 2D. Left: shows how the discrete MRS (brown) $\mathcal{R}^o_\Delta$ is computed as the intersection of the projected FRS (orange) $\mathcal{R}^{o,+}_\Delta$ and BRS (green) $\mathcal{R}^{o,-}_\Delta$. Right: gives an example of a likely convex approximation $\hat{\mathcal{R}}^o_\Delta$ (violet) - mostly within $\mathcal{R}^o_\Delta$, but perhaps slightly exceeding it.

hypothesis testing to guarantee with high probability that this violating volume is bounded by an ϵ of choice. Overall, our estimated MRS $\hat{\mathcal{R}}^o_\Delta$ is guaranteed to have no more than $(\epsilon + \Delta^3)$ percent of its volume lying outside the true MRS $\mathcal{R}^o$.

5.4.2 Covering Object Space with MRS

Given the ability to compute an MRS from a single seed, we now seek to build an "α-approximate" (see Defn. A3) convex cover of the free object space $\mathcal{Q}^o_{free}$. This enables coverage of most arbitrary queries. We construct this cover by iteratively sampling the uncovered region $\mathcal{Q}^o_{free,uncovered} := \mathcal{Q}^o_{free} \setminus \bigcup_{i=1}^{N} \hat{\mathcal{R}}^{o,i}_\Delta$, seeding new sets from sampled points $q^o_j \sim \text{Uniform}(\mathcal{Q}^o_{free,uncovered})$. For each sample, we generate a stable two-point grasp by selecting symmetric contact points on the object and checking inverse kinematics (IK) for feasibility. Under mild assumptions, this procedure is probabilistically complete.

5.4.3 Linking MRS Into a Graph

Given the approximate cover, we construct a graph and pose a shortest path problem following Eqns 5.9.

Each convex region $\hat{\mathbb{R}}^{o,i}_\Delta$ contributes an intraset "move-object" edge $e^i = (x^i_{\text{in}}, x^i_{\text{out}})$ with cost $\ell^i = d(x^i_{in}, x^i_{out}, \cdot)$ (Eqn. (5.3)) and constraints $x^i_{in/out} \in \hat{\mathbb{R}}^{o,i}_\Delta$. If this cost depends on full configurations (e.g. penalizing actuation effort), we must approximate it as a function of the object configurations to use it in our object-space graph (Sec. 5.4.3).

To enable transitions between sets, we also add "regrasp" edges between intersecting MRS. Specifically, we assume a regrasp is possible between configurations q_i and q_j as long as they share the same object pose ($q^o_i = q^o_j$). In SE(2), this is typically valid due to planar support of the object. In practice, this assumption holds for $\geq 97.5\%$ of queries (Sec. 5.6). To find valid regrasp pairs (i, j), we identify intersecting sets $\hat{\mathbb{R}}^{o,i}_\Delta \cap \hat{\mathbb{R}}^{o,j}_\Delta \neq \emptyset$ via linear programs (LP). Each pair adds two directed

edges: (x^i_{out}, x^j_{in}) with constraint $x^i_{out} = x^j_{in}$ and (x^j_{out}, x^i_{in}) with constraint $x^j_{out} = x^i_{in}$. Edge costs (ℓ^{ij} and ℓ^{ji}, respectively) are handled as above.

Finally, given a query $(q^o_{start}, q^o_{goal})$, we first identify which convex sets contain these object poses using LP. Then we add zero-cost dummy edges (q^o_{start}, x^i_{in}) for all sets $i \in \mathbb{I}_{start}$ that contain q^o_{start} and likewise for q^o_{goal}.

Handling SE(2) in GCS: GCS assumes a Euclidean configuration space, but SE(2) includes a Riemannian angular component. To bridge this mismatch, we expand the angular domain from $[-\pi, \pi]$ to $[-2\pi, 2\pi]$ when constructing the set cover. This allows us to identify overlapping sets that are equivalent modulo 2π. For any such pair (q_i, q_j) where $q_i(\theta) = q_j(\theta) \pm 2\pi$, we connect their corresponding graph vertices with symmetric edges enforcing $x^i_{out} = x^j_{in} \pm 2\pi$.

Approximating Manipulator Configuration Costs in Object Space: GCS requires edge costs ℓ^i_c and ℓ^{ij}_r to be nonnegative convex functions of their arguments. Due to the complexity of analytically expressing many useful cost functions in such a manner (e.g., time taken for a robot to execute a particular maneuver), we instead approximate edge costs using a regression procedure.

Regression. Given a dataset of input-output pairs $\mathcal{D} = \{(x_k, y_k)\}_{k=1}^K$, least squares regression seeks a function $f_\theta : \mathbb{R}^n \to \mathbb{R}$ that best explains a data under a squared loss:

$$\theta^\star \;=\; \arg\min_{\theta\in\Theta} \sum_{k=1}^{K} \Big(f_\theta(x_k) - y_k \Big)^2 . \tag{5.10}$$

where Θ is the parameter space for the selected model class. Because GCS requires that cost functions be nonnegative convex functions, we consider two model classes: *nonnegative convex quadratics* and *constants*. We use quadratics to represent contact edge costs and constants to represent regrasp edge costs; these model classes are selected based on validation loss in a sample dataset.

Contact Edge Costs. In order to approximate contact costs of edge e^i_c, we first collect a dataset $\mathcal{D}^i_c$ consisting of pairs of start and end object configurations $({}^k v_{\text{start}}, {}^k v_{\text{end}}) \in V^i_{\text{in}} \times V^i_{\text{out}}$ and the corresponding ${}^k c = c(\pi({}^k v_{\text{start}}, {}^k v_{\text{end}}))$. This is done by sampling $({}^k v_{\text{start}}, {}^k v_{\text{end}})$ pairs from $V^i_{\text{in}} \times V^i_{\text{out}}$ uniformly at random, then evaluating $c(\pi({}^k v_{\text{start}}, {}^k v_{\text{end}}))$. Data from IV.A.1 can also be used. Now, let $x_k := \begin{bmatrix} {}^k v_{\text{start}} \\ {}^k v_{\text{end}} \end{bmatrix} \in \mathbb{R}^6$ and $y_k := {}^k c$. We seek a nonnegative convex quadratic $f_\theta(x) = x^\top A x + b^\top x + c$ that minimizes the squared error on $\mathcal{D}^i_c$:

$$\begin{aligned} \min_{A,b,c} \quad & \sum_{(x_k, y_k)\in\mathcal{D}^i_c} \Big(x_k^\top A\, x_k + b^\top x_k + c - y_k \Big)^2 \\ \text{s.t.} \quad & A \succeq 0, \;\; M \;=\; \begin{bmatrix} c & \frac{1}{2} b^\top \\ \frac{1}{2} b & A \end{bmatrix} \succeq 0, \end{aligned} \tag{5.11}$$

where the first constraint enforces convexity and the second constraint guarantees $f_\theta(x) \geq 0$ for all x. The optimal $f_{\theta^\star}$ defines the contact-edge cost $\ell^i_c(v_{\text{start}}, v_{\text{end}}) = f_{\theta^\star}\Big([v_{\text{start}}; v_{\text{end}}]\Big)$.

Regrasp Edge Costs. For each regrasp edge e^{ij}_r we assemble $\mathcal{D}^{ij}_r = \{({}^k v, {}^k c)\}$, where ${}^k v \in V^i_{\text{out}} \cap V^j_{\text{in}}$ is the unchanged object pose and ${}^k c = c(\pi_{\text{regrasp}}({}^k q^i, {}^k q^j))$ is the cost of the trajectory returned

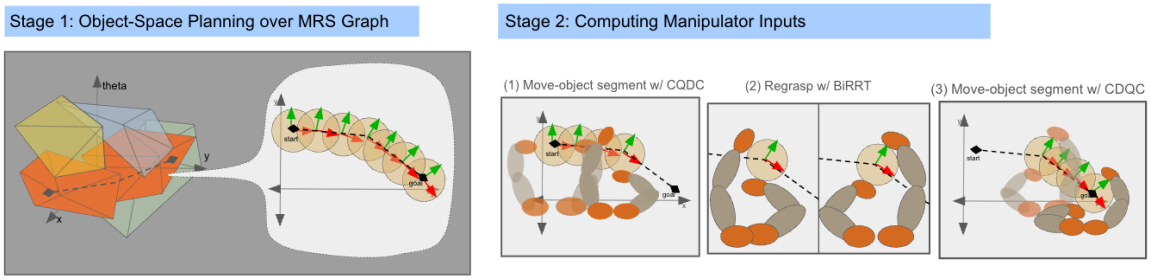


Figure 5.5: This figure illustrates the 2-stage hierarchical planning process. On the left, the first stage takes in $q^o_{start}, \mathcal{Q}^o_{goal} = (q^o_{goal}, r)$ and computes an object plan $q^{o,*}_{0:T}$. The next stage takes in this object plan and produces a manipulator input trajectory $u^*_{0:T-1}$.

by a regrasp policy π_{regrasp} from the system configuration ${}^k q^i$ to ${}^k q^j$; note that $\mathbf{proj}_{Q^o}({}^k q^i) = \mathbf{proj}_{Q^o}({}^k q^j) = {}^k v$. A constant model $f_\theta({}^k v) = \bar{c}$ with $\bar{c} = \frac{1}{|\mathcal{D}_r^{ij}|} \sum_k ({}^k c)$ minimizes (5.10) for this dataset and satisfies $\bar{c} \geq 0$ by construction.

5.4.4 Guarantees for Object-Space Planning

Taken altogether, the object space path produced by GCS will bounded suboptimal, up to the choice of set cover and trajectory parametrization. We verify in Sec. 5.6 that the suboptimality bound is negligibly small. This implies that our object path is effectively optimal given these assumptions. It is difficult to state guarantees for the corresponding configuration space path, since GCS optimizes a proxy to the true objective(Sec. 5.4.3). However, we find that our configuration space path are empirically several times better than a SOTA baseline.

5.5 Methodology - Online Hierarchical Planning

Given a query $(q^o_{start}, \mathcal{Q}^o_{goal})$, we plan in two stages:

1. compute an object-space plan $q^{o,*}_{0:T}$ over the MRS graph, and
2. recover the corresponding manipulator controls $u^*_{0:T-1}$

This hierarchical design offers some key advantages. Firstly, it offers *efficiency*. Searching low-dimensional object space (3D here) is much more efficient than searching high-dimensional full configuration space (9D here). Plus, much of the manipulator's configuration space is redundant, as multiple configurations may produce the same object motion. Besides object space search, our method only needs to track the object trajectory—a simple problem that is quickly solved with local planner π. This enables fast planning without sacrificing quality.

Secondly, it offers a *high planning success rate*. A well-known limitation of hierarchical planners is that the high-level plans may turn out to be kinematically or dynamically infeasible, which lowers success rates. In contrast, our object plans are built over reachable sets, making them feasible by design. As a result, the second stage can generally recover valid manipulator inputs and configurations.

Algorithm 3 Transcription of GCS plan

Require: Object trajectory $q^{o,*}_{0:N}$; sequence of sets $\{\hat{\mathcal{R}}^o_k\}_{k=0}^{N-1}$
Ensure: Control trajectory T_u
$q \leftarrow \textsc{GetGrasp}(\hat{\mathcal{R}}^o_0, q^{o,*}_0)$ {Initialize grasp}
$T_u \leftarrow [\,]$ {Initialize empty control list}
for $n \leftarrow 0$ **to** $N-2$ **do**
  $(q, u_{0:T^c_n}) \leftarrow \pi(q, q^o_{n+1})$ {Generate contact segment}
  $T_u \leftarrow \textsc{Extend}(T_u, u_{0:T^c_n})$
  $q' \leftarrow \textsc{GetGrasp}(\hat{\mathcal{R}}^o_{n+1}, q^o)$ {Grasp for next set at current object pose}
  $u_{0:T^r_n} \leftarrow \pi_{\text{regrasp}}(q, q')$ {Generate regrasp segment}
  $T_u \leftarrow \textsc{Extend}(T_u, u_{0:T^r_n})$
  $q \leftarrow q'$
end for
$(q, u_{0:T^c_N}) \leftarrow \pi(q, q^o_N)$ {Final contact segment}
$T_u \leftarrow \textsc{Extend}(T_u, u_{0:T^c_N})$
return T_u =0

Overall, our design provides a fast, dependable framework for solving global planning problems. We expand on its components below.

Stage 1: Object-Space Planning Over the MRS Graph

Given a query $(q^o_{start}, q^o_{goal})$, we add these states to the MRS graph (Sec. **??**) and solve for a low-cost object trajectory $q^{o,*}_{0:T}$ using GCS. This step is fast—typically under 10 seconds—due to the low dimensionality of object space.

By construction, any path through the MRS graph should be both kinematically and dynamically feasible: "move-object" transitions satisfy Lemma 5.3.1 and all regrasps are assumed feasible. In practice, however, approximations can lead to rare infeasibilities (discussed in Sec. 5.5).

Stage 2: Computing Manipulator Inputs

Next, we compute the manipulator inputs $u^*_{0:T-1}$ by closed-loop tracking of the object trajectory $q^{o,*}_{0:T}$ from Stage 1. We assume Stage 1 also provides the sequence of MRS indices $(I_1, \ldots, I_k)$ and regrasp times $(T_1, \ldots, T_{k-1})$. In this stage, we alternate between "translating" **move-object** and **regrasp** phases. For each move-object phase, we track an object path segment, $q^{o,*}_{T_i:T_{i+1}}$. For each regrasp phase, we hold the object static at $q^{o,*}_{T_i}$ and find a collision-free plan from the current grasp to the next. We elaborate below.

Move-object phase: We use CQDC to track $q^{o,*}_{T_i:T_{i+1}}$. Failures here typically occur when the convex-approximated MRS $\hat{\mathcal{R}}^o_\Delta$ includes states that are not truly reachable, making reaching the target $q^{o,*}_{T_{i+1}}$ dynamically infeasible.

Regrasp phase: At the ith regrasp, assume the current translated state is q_j, where $q^o_j = q^{o,*}_{T_i}$ (we have reached the most recent target object pose). We assume the object is static between regrasps, so all we have to do is calculate the new grasp and then compute a collision-free plan to it. We first have to calculate the next grasp, as Stage 1 only provides the object configuration at

the grasp, $q_{T_i}^{o,*}$, and the identity of the next set, I_{i+1}. To find the manipulator configuration in the grasp, we simply attempt to reach the object configuration from the full seed configuration, just like during MRS construction. That is, the next grasp can be computed as: $q'_{grasp} := \pi(q_{seed,I_{i+1}}, q_{T_i}^{o,*})$. After that, we simply use BiRRT ([**?**]) to find a collision-free path between q_j and q'_{grasp}.

Failures here occur when (1) the new grasp is unreachable (again due to MRS approximation error), or (2) no collision-free regrasp path exists. The latter is rare—our offline phase assumes regrasps are always possible when $q_i^o = q_k^o$, which is mostly valid in SE(2)—but kinematic infeasibility can still appear in $\leq 2.5\%$ of queries (e.g., when the object traps the manipulators against the base).

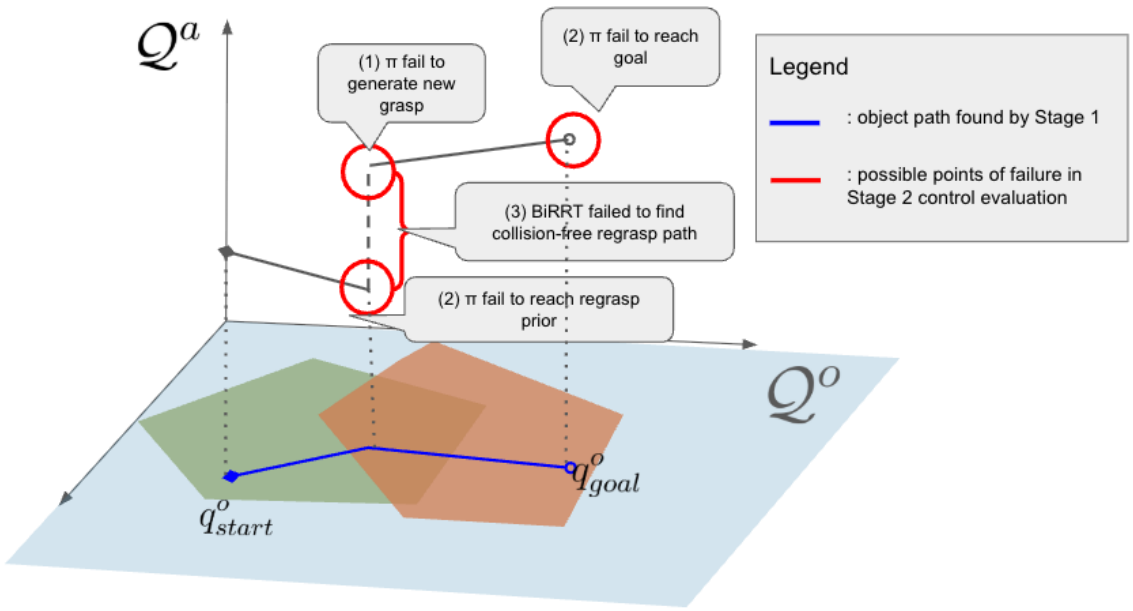


Figure 5.6: This figure illustrates the various points of failure that may occur in Stage 2 of hierarchical planning. The local planner π may fail to reach the goal of a "move-object" segment or to generate the regrasp target. BiRRT may also fail to find a collision-free path for executing regrasp.

Sampling Multiple Paths to Mitigate Infeasibility: Since these failures stem from the chosen object path, we increase robustness by sampling multiple candidates in Stage 1. GCS solves a convex relaxation of a MIP, yielding soft edge weights that define traversal probabilities. Rather than evaluating only the nominal lowest-cost path, we sample several candidates and select the lowest-cost *feasible* trajectory after validating them all in Stage 2. This boosts success rates without significant additional computation (Sec. 5.6).

5.6 Experiments

Task Setup We evaluate on planar reorientation of a cylindrical object using a bimanual KUKA iiwa-7 system. This is a challenging and representative contact-rich manipulation task: it requires

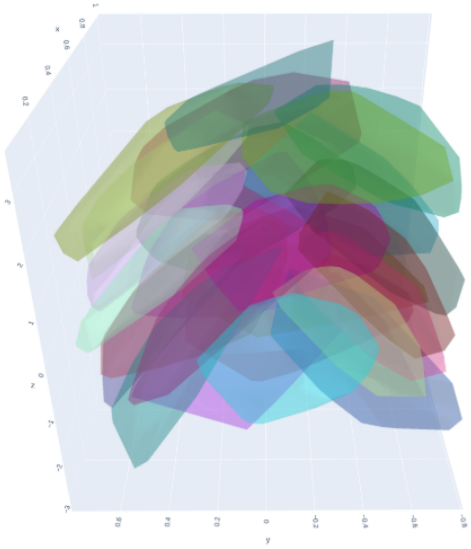


Figure 5.7: This figure illustrates the α-approximate set cover we find, which contains only 36 sets. By covering the object space with MRS, we reduce the combinatorial complexity of the planning problem to a minimum.

exploiting intrinsic dexterity to complete efficiently, while reasoning about multi-point contacts and complex arm geometries. Compared to the simpler end-effector or gripper systems typically studied in model-based planning, this setup is significantly higher dimensional and better tests the capabilities of our planner.

Experimental Questions Our experiments address two key questions:

1. Does our method provide higher-quality global plans?
2. Does it achieve this without sacrificing speed or planning success rate?

Baseline and Parameters We compare against the state-of-the-art sampling-based planner from [45], which uses the same local planner (CQDC-MPC) for its extend operation. This makes for a direct comparison, as the methods differ only in their global search strategy: sampling-based expansion versus graph search over MRS.

Key details for our method: the MRS graph we compute has 36 sets and is fully connected. When approximating manipulator configuration costs in object space, we select the convex function classes using a holdout dataset. Ultimately, we chose a quadratic function to approximate "move-object" cost and a constant for regrasp cost. We implement using Python.

Experimental design: we run 250 queries with random initial and goal object orientations, sampled uniformly from the feasible kinematic workspace.

	Objective cost	Query time (seconds)	Success rate
RRT	7.39 ± 5.36	52.7 ± 47.14	66.8%
Ours	2.55 ± 1.53	35.74 ± 19.48	92.4%

Table 5.1: Objective cost - total actual effort. Query time - time to plan online. Success rate - number of queries for which the algorithm successfully produced a manipulator sequence.

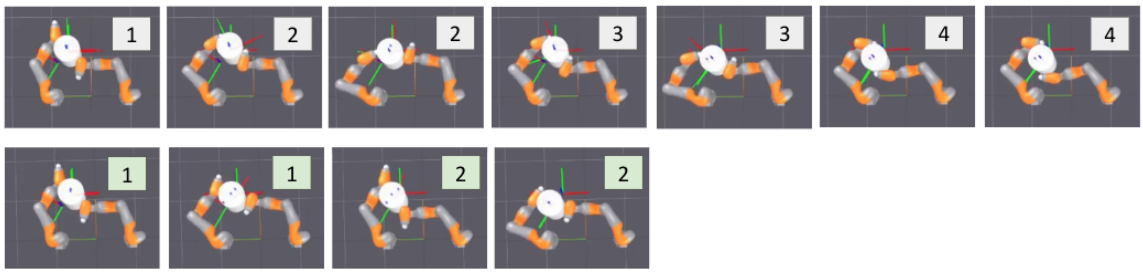


Figure 5.8: This figure shows keyframes for a single query: top row is RRT solution, bottom row is ours. The frames are numbered according to which "move-object" phase they are in. Hence, for example, between frames numbered 2 and 3, there is a regrasp. Note that the top and bottom row frames are not aligned according timestamp - the RRT path takes much longer to execute than ours (10s vs. 4s). Also note that the RRT solution requires 3 regrasps, whereas ours only uses 1.

Discussion on Path Quality

This task is especially illustrative because it requires contact-rich manipulation to complete efficiently. End-effector manipulation is not suitable for such large, unwieldy objects: it would require frequent regrasps for rotation and may be unable to generate sufficient force for heavy objects.

In keyframe comparisons (Fig. 5.8), we see that RRT often produces circuitous, jittery trajectories. As shown in Table 5.1, RRT struggles to minimize the objective (total manipulator effort). This aligns with intuition: the random growth of sampling-based planners tends to yield inefficient solutions.

By contrast, our method generates concise, direct paths—often resembling those a human might devise. The plans exploit intrinsic dexterity, e.g., reorienting the object by rolling it along manipulator surfaces, or selecting successive antipodal grasps that maximize rotation or translation.

We attribute this effectiveness to our choice of MRS as the discrete unit in this hybrid problem. The unit is high-level enough to keep the search space small, yet structured enough for efficient optimization.[1] With only 36 MRS we obtain an α-approximate cover of the kinematic object space, and the offline graph search completes in just a few seconds.

Discussion on Runtime

Online: Query time is only seconds, not minutes. This is competitive with other SOTA planners, which also take seconds even on simpler point-contact systems [51]. Breaking down the timing: ∼15% is spent on object planning, 25% on translating move-object phases with CQDC-MPC, and 60% on regrasp phases with BiRRT. The bottleneck is BiRRT, which we implemented in Python; a C++ implementation should significantly reduce runtime.

Offline: Building the MRS graph required 3.98M (s, a, s') samples (∼11 hours of real-time equivalent). For comparison, RL and BC require 4–7 robot-hours *per task* for dexterous hand systems [27, 128]. Our graph is task-agnostic: tasks like drawer opening or object reorientation can all be specified within the same framework. Thus, while the upfront cost is higher than task-specific data collection, it amortizes quickly across tasks and provides a reusable foundation for diverse manipulation problems.

[1] In practice, solved via GCS. We compare GCS and exact MIP, showing near-identical cost (optimality gap 0.02%) with GCS much faster (3.16s on average vs. 29.52s).

Discussion on Success Rate

We set RRT's termination condition (max nodes) so average query times matched ours. At comparable runtimes, our planner achieved a 92.4% success rate—about 40% higher than RRT. Most RRT failures came from not finding a path before termination.

Our high success rate again stems from MRS. First, MRS operates at a finer granularity than motion primitives (pushing, pulling, pivoting), allowing us to handle diverse queries requiring different rotations and translations. Second, planning in object space with reachable sets yields paths that are inherently (mostly) kinematically and dynamically feasible, since those are the states captured by the sets.

Failure Modes. Despite this, approximations introduce some infeasibility (Fig. 5.6): 18.2% of failures from start/goal states outside the approximate cover (can be reduced by longer offline computation), 45.5% from dynamic infeasibility in move-object phases due to convex over-approximation of reachable sets, 38.4% from BiRRT failing to find a collision-free regrasp (e.g., when the object traps manipulators against the base).

To address multiple failure modes at once, we sample multiple object paths from GCS, check feasibility downstream, and select the best. This strategy improves success rate by 8.6% at only a 2% increase in query time, and is highly parallelizable.

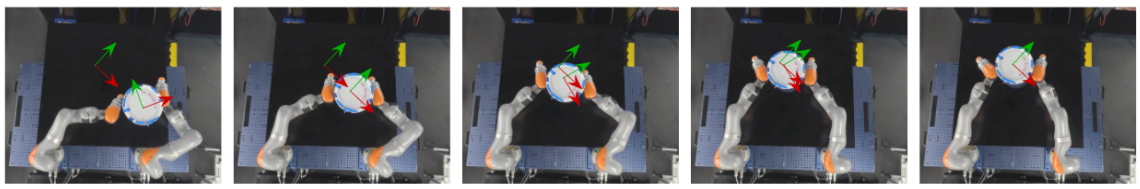


Figure 5.9: This figure shows keyframes for our plan executed on hardware. Notice that robot rotates the bucket using the full surfaces of the orange "hand", demonstrating fine-grained contact-rich behavior.

Hardware Results Finally, we present hardware rollouts in Fig. 5.9. Plans transfer, but execution is sensitive to model mismatch: our planner relies on a quasidynamic, discrete-time approximation, which accumulates error over long horizons. Future work could incorporate higher-fidelity local planners, e.g., shooting methods in physics simulation, to improve robustness.

6 Conclusion

This thesis addressed two seemingly different problems in robotics: enforcing safety for agile systems and planning high-quality motions for contact-rich manipulation. At the core of both lies a unifying idea: reachable sets. By capturing how a system evolves under dynamics and constraints, reachable sets provide a principled way to reason about safety, feasibility, and performance across domains.

On the safety side, we developed methods for synthesizing control barrier functions (CBFs) that extend beyond the standard assumption of perfectly known models. Using sum-of-squares programming, we introduced a robust-adaptive framework that guarantees safety under parametric uncertainty while reducing unnecessary conservatism. To address the scalability limits of such formal approaches, we proposed adversarially trained neural CBFs, which preserve near-perfect safety while scaling to higher-dimensional systems like drones. Together, these contributions show that function-based synthesis can achieve rigorous safety guarantees while remaining practical for real-world platforms.

On the manipulation side, we proposed a hierarchical planner that treats mutual reachable sets as motion primitives. This representation reduces the combinatorial complexity of planning by covering object space with a compact set of primitives and then searching over them as a graph. Experiments with a bimanual KUKA iiwa system demonstrated that this approach scales effectively to challenging reorientation tasks and produces significantly higher-quality trajectories than sampling-based baselines, with successful transfer to hardware execution.

Taken together, these results highlight the breadth and versatility of reachable sets as a computational primitive. They enable both reactive safety enforcement for agile systems and scalable, expressive planning for contact-rich manipulation.

There are several natural extensions of this work. On the safe control side, a key direction is to broaden neural control barrier function (CBF) synthesis beyond known models. Handling stochastic disturbances, time-varying dynamics, and partial observability would make these methods more directly applicable to real systems. One approach is to incorporate chance constraints or distributionally robust formulations into the adversarial training framework, so that controllers can maintain safety with quantified risk while remaining minimally conservative.

On the planning side, the mutual reachable set (MRS) abstraction used for bimanual SE(2)

reorientation tasks can be extended to more general domains. In particular, extending the graph-of-sets representation to SE(3) object manipulation would test its scalability while preserving its advantage of reducing combinatorial complexity. At the same time, the hardware results in this thesis show that model mismatch remains a limiting factor. Addressing this will likely require higher-fidelity local solvers, differentiable contact models, and data-driven corrections from tactile sensing.

These directions continue the central theme of the thesis: using reachable sets as a computational primitive that supports both safety and planning, and adapting them to more complex models, tasks, and hardware constraints.

In summary, this thesis demonstrates that by elevating reachable sets from a specialized analysis tool to a central design abstraction, we can develop algorithms that are safe, scalable, and performant across diverse robotic domains. This perspective not only advances the particular problems studied here but also points toward a broader design philosophy for robotics: leveraging general mathematical abstractions to unify solutions across seemingly disparate challenges.

A1 Appendix for Chapter 3

Table A1.1

Computation Lengths for Synthesis			
	2D Toy	Cartpole	Quadrotor
Total Time (sec)	23	655	68,934
Total Iterations	11	159	337
Avg. P1 Time (sec)	0.1	0.12	0.87
Avg. P2 Time (sec)	1	2.8	74
Avg. P3 Time (sec)	0.8	1.2	130

Table A1.2

Synthesis & Estimation Hyperparameters			
	2D Toy	Cartpole	Quadrotor
Deg. $\phi(x, \hat{\theta})$	3	3	3
β^-	-1.0	-0.5	-0.1
α	0.1	0.01	0.01
γ	10.0	10.0	10.0
$\nu(\rho)$	$\text{atan}(\rho) + 1$	$\text{atan}(\rho) + 1$	$\text{atan}(\rho) + 1$
η	100	100	1000
σ	1	1	1
ξ	1	1	1
ρ range	[0,10]	[0,10]	[0,10]

A1.1 System Parameters

- Cartpole: $m_c = 1\text{kg}$, $m_p = 0.1\text{kg}$, $l = 0.5\text{m}$
- Quadrotor: $m = 0.486\text{kg}$, $r = 0.25\text{m}$, $I = 0.00383\text{kg}\cdot\text{m}^2$

A1.2 Final raCBF

2D toy: $\phi(x,\hat{\theta}) = (0.3666)\hat{\theta}^2 + (0.07823)\hat{\theta}x_1 + (0.0007759)\hat{\theta}x_2$
$- (1.673)x_1^2 - (2.005)x_1x_2 - (1.726)x_2^2 - (0.8380)\hat{\theta} - (0.09071)x_1$
$- (0.0007912)x_2 + 1.098$

Cartpole: $\phi(x,\hat{\theta}) = (23.24)\hat{\theta}^2 + (0.0002120)\hat{\theta}z_1 - (22.73)\hat{\theta}z_2$
$- (0.04065)x_3^2 + (1.136)x_3z_1 - (10.06)z_1^2 - (8.015)z_2^2 - (12.98)\hat{\theta}$
$+ (10.27)z_2 + 2.785$

Quadrotor: $\phi(x,\hat{\theta}) = (-2.663)\hat{\theta}_1^2 - (0.06829)\hat{\theta}_1\hat{\theta}_2$
$+ (0.05703)\hat{\theta}_1z_2 + (0.3199)\hat{\theta}_2^2 - (0.08092)\hat{\theta}_2z_2 - (0.7239)x_1^2$
$+ (0.03167)x_1x_3 - (1.138)x_1x_4 + (3.074)x_1z_1 - (0.01136)x_3^2$
$+ (0.02167)x_3x_4 - (0.1209)x_3z_1 - (1.462)x_4^2 + (2.118)x_4z_1$
$- (5.706)z_1^2 - (5.308)z_2^2 + (0.06482)z_2 + 0.9881$

A2 Appendix for Chapter 4

A2.1 Extended related works

There are also many works on neural CBFs, but they target problems unrelated to input saturation. Some examples: learning an unknown safety criterion from safe expert trajectories [129, 130], jointly learning a safe policy and safety certificate via reinforcement learning [131, 132], optimizing the task performance of a CBF-based controller [133], and learning dynamics under model uncertainty [134].

A2.2 Appendix for preliminaries

Defining the safe sets: we define the safe sets corresponding to the limit-blind CBF and modified CBF. We need to first define the following functions, for all j in $[1, r-1]$ where r is the relative degree betweeen safety specification $\rho(x)$ and input u:

$$\phi_j = \left[\prod_{i=1}^{j}\left(1 + c_i \frac{\partial}{\partial t}\right)\right]\rho \quad , \forall j \in [1, r-1] \tag{A2.1}$$

and then from [11], we have

$$\mathcal{X}_{safe} = \mathcal{A} \cap \left[\bigcap_{j=1}^{r-1}\{\phi_j\}_{\leq 0}\right] \qquad \text{(limit-blind safe set)}$$

$$\mathcal{S}^* = \mathcal{X}_{safe} \cap \{\phi^*\}_{\leq 0} \qquad \text{(modified safe set)}$$

For this paper, it is also convenient to indicate the function $\text{ss}^*(x)$ that implicitly defines the safe set $\mathcal{S}^*$ as its 0-sublevel set.

$$\mathcal{S}^* = \{\text{ss}^*(x)\}_{\leq 0} \tag{A2.2}$$

$$\text{ss}^*(x) = \max_{\forall j}(\rho(x), \phi_j(x), \phi^*(x)) \tag{A2.3}$$

Comparison to the class-κ CBF formulation: there is a different CBF formulation that requires

$$\dot{\phi}(x) \leq -\alpha(\phi(x)) \ , \ \forall x \in \mathcal{D} \tag{A2.4}$$

for a class-κ function $\alpha : \mathbb{R} \to \mathbb{R}$ ($\alpha(0) = 0$, α is continuous and monotonically increasing). While this is a stricter constraint than ours (it constrains the inputs at all states, not just boundary states), the benefit is that it produces a smooth control signal. It could be possible to extend our method to CBFs of this variety. The only major change is that the critic would search for counterexamples in the domain $\mathcal{D}$, rather than just along the safe set boundary, $\partial\mathcal{S}^*$. One could learn the $\alpha(\cdot)$ function as well, parametrizing as a monotonic NN [135].

A2.3 Appendix for methodology

More recommendations for the design of $\rho^*(x)$
If it is very awkward to choose an x_e in Eqn. 4.5, then another design can be used. For example, $\rho^*(x)$ can be:

$$\rho^*(x) = \text{softplus}(\text{nn}(x)) + \rho(x) \tag{A2.5}$$

The disadvantage of such a design is that the safe set might be empty (Contraint 3 is violated). It might be fine to starting training with no safe set, since the volume regularization term in the objective may slowly create a safe set.

Helper functions for training algorithm

Algorithm 4 Sampling uniformly on a boundary (MSample from [136])

function SAMPLEBOUNDARY(θ, N_{samp}) {Note that θ defines the boundary, $\partial\mathcal{S}^*$ }
  Set error parameter $\epsilon \in (0, 1]$ to 0.01, boundary attribute τ to 0.25, n to state space dim.
  $\mathbb{X}_{samp} \leftarrow \{\}$
  $\sigma \leftarrow 2\left(\tau\sqrt{\epsilon}4(n + 2\ln(1\epsilon))\right)^2$ {Set hyperparam. to meet sampling guarantees}
  While size of $\mathbb{X}_{samp} < N_{samp}$:
  $p \leftarrow$ sample uniformly inside $\mathcal{S}^*$
  $q \leftarrow$ sample from Gaussian$(p, \sigma \cdot I_{n\times n})$
  $x \leftarrow$ attempt to intersect segment $\overline{pq}$ with boundary
  If x is not none:
  Add x to $\mathbb{X}_{samp}$
  return $\mathbb{X}_{samp}$
end function=0

Algorithm 5 Projecting to a boundary

0: **function** ProjToBoundary($\mathbb{X}, \theta$) {Project set $\mathbb{X}$ to boundary defined by θ}
0: Set learning rate $\gamma = 0.01$
0: $\text{ss}^*_\theta \leftarrow$ function that defines the boundary implicitly ($\partial\mathcal{S}^* \triangleq \{\text{ss}^*_\theta\}_{=0}$)
0: **Repeat:**
0: $\mathbb{X} \leftarrow \mathbb{X} - \gamma \cdot \nabla_{\mathbb{X}} |\text{ss}^*_\theta(\mathbb{X})|$ {Batch GD}
0: **Until** convergence
0: **return** $\mathbb{X}$
0: **end function**=0

For the critic, the first step to computing boundary counterexamples is sampling on the boundary. Alg. 3 from [136] provides a method to uniformly sample on manifolds with bounded absolute curvature and diameter. The algorithm finds points on the boundary by sampling line segments and checking if they intersect the boundary. An important trait of the algorithm is that it is efficient: the number of evaluations of the membership function ss^*_θ does not depend on the state space dimension, n. It only depends on the curvature of the boundary (captured by an inversely proportional "condition number", τ) and the error threshold, ϵ, which bounds the total variation distance between the sampling distribution and a true uniform distribution. We do not measure or estimate τ; for our purposes, it is enough to set it sufficiently small. Another essential helper routine (Alg. 5) projects states onto the boundary. The boundary $\partial\mathcal{S}^*$ is implicitly defined as the 0-level set of a function ss^*_θ. Thus, we can simply apply gradient descent to minimize $|\text{ss}^*_\theta(x)|$ toward 0, which is a "good enough" approximate projection scheme.

A2.4 Appendix for experiments

System model for quadcopter-pendulum

In our 10D state and 4D input model, the states are roll-pitch-yaw quadcopter orientation (γ, α, β) and roll-pitch pendulum orientation (ϕ_p, θ_p), as well as the first derivatives of these states. The inputs are thrust and torque $(F, \tau_\gamma, \tau_\beta, \tau_\alpha)$, which are limited to a bounded convex polytope set. The quadcopter dynamics are from [137], which models the inputs realistically, and the pendulum dynamics are from [104]:

$$\begin{bmatrix} \ddot{\gamma} \\ \ddot{\beta} \\ \ddot{\alpha} \end{bmatrix} = R(\gamma, \beta, \alpha) J^{-1} \begin{bmatrix} \tau_\gamma \\ \tau_\beta \\ \tau_\alpha \end{bmatrix} \tag{A2.6}$$

$$\begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{3}{2mL_p \cos\theta}(k_y(\gamma,\beta,\alpha)\cos\phi + k_z(\gamma,\beta,\alpha)\sin\phi) \\ \frac{3}{2mL_p}(-k_x(\gamma,\beta,\alpha)\cos\theta - k_y(\gamma,\beta,\alpha)\sin\phi\sin\theta + k_z(\gamma,\beta,\alpha)\cos\phi\sin\theta) \end{bmatrix} (F + mg) + \begin{bmatrix} 2\dot{\theta}\dot{\phi}\tan\theta \\ -\dot{\phi}^2 \sin\theta\cos\theta \end{bmatrix} \tag{A2.7}$$

where $R(\gamma, \beta, \alpha)$ rotates between the quadcopter and world frame and is computed as the composition of the rotations about the X, Y, Z axes of the world frame. Also, the variables $k_x(\gamma, \beta, \alpha), k_y(\gamma, \beta, \alpha), k_z(\gamma, \beta, \alpha)$ are defined as:

$$R(\gamma, \beta, \alpha) \triangleq R_z(\alpha) R_y(\beta) R_x(\gamma) \tag{A2.8}$$

$$k_x(\gamma, \beta, \alpha) \triangleq (\cos\alpha \sin\beta \cos\gamma + \sin\alpha \sin\gamma) \tag{A2.9}$$

$$k_y(\gamma, \beta, \alpha) \triangleq (\sin\alpha \sin\beta \cos\gamma - \cos\alpha \sin\gamma) \tag{A2.10}$$

$$k_z(\gamma, \beta, \alpha) \triangleq (\cos\beta \cos\gamma) \tag{A2.11}$$

and $J = \text{diag}(J_x, J_y, J_z) = \text{diag}(0.005, 0.005, 0.009)$ kg $\cdot$ m^2 contains the moments of inertia of the quadcopter, $m = 0.84$ kg is the mass of the combined system, $L_p = 0.03$ m is the length of the pendulum. The values of these physical parameters are taken from the default values in a high-fidelity quadcopter simulator, jMAVSim [138] and also extrapolated from the real-world experiments in [104]. Our control inputs are limited to a convex polyhedral set defined by:

$$\mathcal{U} \triangleq \{u \mid u + [mg, 0, 0, 0] = Mv, \text{for some } v \in [\vec{0}, \vec{1}]\} \tag{A2.12}$$

$$M \triangleq \begin{bmatrix} k_1 & k_1 & k_1 & k_1 \\ 0 & -\ell k_1 & 0 & \ell k_1 \\ \ell k_1 & 0 & -\ell k_1 & 0 \\ -k_2 & k_2 & -k_2 & k_2 \end{bmatrix} \tag{A2.13}$$

with $\ell = \frac{0.3}{2}, k_1 = 4.0, k_2 = 0.05$ from jMAVSim. The interpretation of this is that v contains the low-level motor command signals at each rotor, which are limited between 0 to 1, and we linearly transform them to thrusts and torques using the *mixer matrix* M, which is derived using first principles [137]. Finally, we perform a change of variables on the input by adding mg to the thrust so that the origin is an equilibrium ($\dot{x}|_{x=0} = 0$).

Baseline details

Hand-designed CBF: the system was not very intuitive to reason about, so we picked a simple and general function form from [8, 14]:

$$\rho^* = (\gamma^2 + \beta^2 + \delta_p^2)^{a_1} - (\pi 4)^{2 \cdot a_1} + a_2 \tag{A2.14}$$

where $a_1 > 0, a_2 \geq 0$ are the parameters. The parameters were optimized with an evolutionary algorithm, Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [139] using the same objective function from our method for fairness. After tuning the hyperparameters of CMA-ES, the best parametrization we found was:

$$\rho^* = (\gamma^2 + \beta^2 + \delta_p^2)^{3.76} - ((\pi 4)^2)^{3.76} \tag{A2.15}$$

$$\phi^* = \rho^* + 0.01 \cdot \dot{\rho}^* \tag{A2.16}$$

Safe MPC: the MPC formulation was kept similar to CBF-QP. The objective here is also to

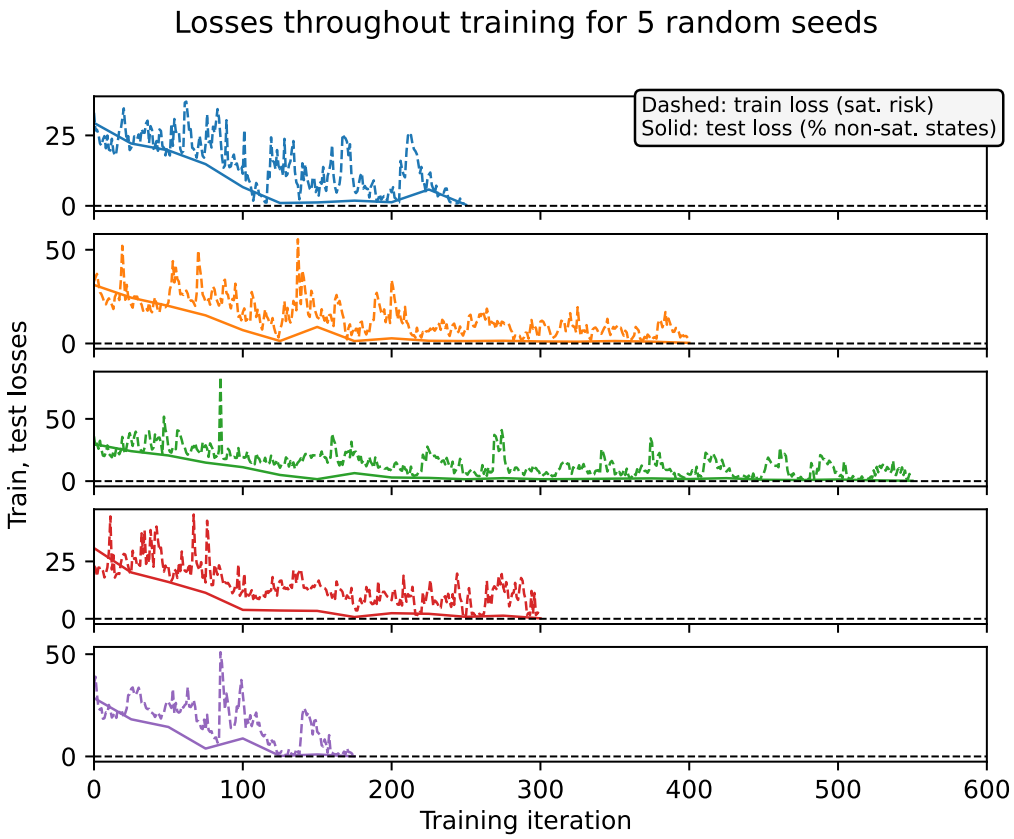


Figure A21: Plot of train and test loss over training for 5 runs with different random seeds. The black dashed line marks 0, the target loss. As we can see, the runs finish training in different lengths of time, but they all ultimately train successfully (reach ≈ 0 loss).

minimize modification to $k_{nom}(x)$ while keeping the trajectory safe and forward invariant.

$$\min_{u(t)\in\mathcal{U}} \int_0^T \|u(t) - k_{nom}(x(t))\|_2^2 \, \partial t \tag{A2.17}$$

$$x(0) = x_0 \tag{A2.18}$$

$$\dot{x}(t) = f(x(t)) + g(x(t))u(t) \tag{A2.19}$$

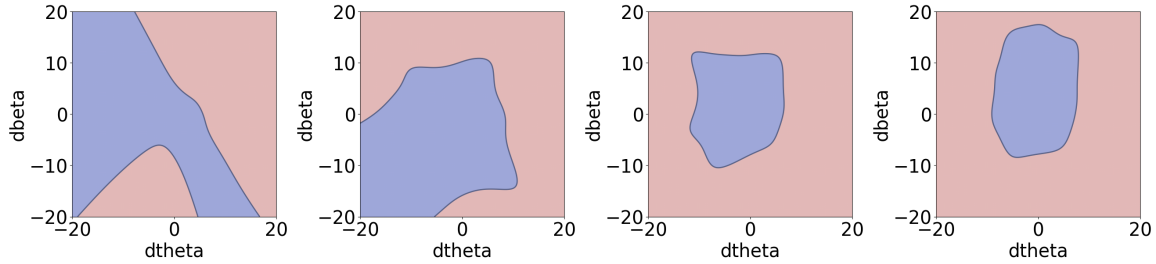
$$\rho(x(t)) \leq 0, \forall t \in [0, T] \tag{A2.20}$$

$$\rho_b(x(T)) \leq 0 \qquad \text{(terminal constraint)}$$

The terminal constraint ensures invariance (safety for all time) of the MPC solution by enforcing the last predicted state $x(T)$ to lie in an invariant set defined by $\rho_b(x)$. We set $\rho_b(x)$ to be the approximated region of attraction of an LQR stabilizing controller: $\rho_b(x) = \|x\|_2^2 - 0.1$.

Training details

Random seeds: We trained a neural CBF for the quadcopter-pendulum problem on 5 different random seeds. The random seed affects the neural CBF initialization, the critic's counterexamples, etc. The test loss (% non-saturating states) consistently reaches ≈ 0 across seeds; the seeds only



(a) Iter. 0 **(b)** Iter. 100 **(c)** Iter. 120 **(d)** Iter. 250

Figure A22: An axis-aligned 2D slice (depicting $\dot{\theta}$ (pendulum pitch velocity) vs. $\dot{\beta}$ (quadcopter pitch velocity) of the 10D safe set, at four points during training. The safe set being learned is in blue.

affect how long it takes to reach this loss (14 ± 4 hours, on average). For Table 4.1, we chose the run that yielded a CBF that balances performance and a large safe set volume.

Training hyperparameters: (1) Critic: takes 20 gradient steps with learning rate 1e−3 to optimize batch of size 500. To initialize the batch, uses 50% uniform random samples, 50% warmstarted from the previous critic call. (2) Neural CBF: nn : $\mathbb{R}^n \to \mathbb{R}$ is a multilayer perceptron with 2 hidden layers (sizes 64, 64) and $\tanh, \tanh,$ softplus activations. It uses the default Xavier random initialization [140]; the c_i coefficients are initialized uniformly in $[0, 0.01]$. (3) Regularization: we used regularization weight 150.0 and 250 state samples in $\mathcal{D}$ to compute the regularization term. (4) Learner: used learning rate 1e−3.

Ablation study: We conduct ablation to analyze the effect of two key design choices: (1) our regularization term and (2) batch computing counterexamples.

Reg weight	Volume (as % of domain volume)
0	5.72e−3
10	8.36e−3
50	1.34e−2
200	1.91e−2

Batch size	Training time (h)
1	-
10	2.50
50	1.88
100	1.88
500	11.00

Table A2.1: (Left) Demonstrating how increasing the regularization weight effectively increases the volume of the learned safe set. (Right) Demonstrating how using a medium-sized *batch* of counterexamples can provide significant speed gains. Batch size 1 didn't finish.

For the regularization term, we measure the impact of the regularization weight on the volume of the learned safe set. We varied the weight between 0 (no regularization) and 200 and chose learned safe sets that attained a similarly low loss (within 0.05 of each other). Next, we approximated the safe set volume by sampling: we took 2.5 million uniformly random samples in the state domain and checked whether they belonged to the safe set. We see in Table A2.1 that increasing the regularization weight effectively increases the volume.

For batch computing counterexamples, we measure the effect of batch size on the training time. To compute training time, we consider training finished when the loss drops below a certain threshold

(note that batch size 1 didn't finish). We might expect that for medium-sized batches improve the quality of the counterexamples (since there are more counterexample options in a batch), resulting in more efficient training. On the other hand, we also expect that for larger batch sizes, the overhead of creating the batch (mainly, sampling a large number of points on the boundary) exceeds any speed gains. In fact, this is what we observe in Table A2.1: as we increase the batch size from 10 to 50, the training speed improves by 25%. But a further increase from 100 to 500 sees the speed drop due to the aforementioned overhead.

Testing details

Implementing safe control in discrete time: Our CBF has a continuous-time formulation, and moreover, yields discontinuous control which is abruptly activated at the boundary. This means that in discrete time, where the system state is sampled at some frequency, the system might reach the boundary *in between* samples and the safe control may not kick in to prevent exiting from $\mathcal{S}^*$. In this case, we should have safe control kick in slightly before the boundary. This can be at a fixed distance from the boundary (at $\text{ss}^*(x) = -\epsilon$ for small $\epsilon > 0$) or we can leverage the known dynamics to apply safe control when the boundary would otherwise be crossed in the next time step. We use the latter approach when simulating rollouts for Table 4.1. Another way to address this could be to seek finite-time or asymptotic convergence guarantees, in addition to forward invariance guarantees. If we had them, the system would be returned quickly to $\mathcal{S}^*$ should it ever exit. This is acceptable in most cases, as it is only mandatory for the system to stay inside the user-specified allowable set $\mathcal{A}$, which contains $\mathcal{S}^*$. Generally, the system will exit $\mathcal{S}^*$ without exiting $\mathcal{A}$.

Testing hyperparameters: (1) For the metric "% of non-saturating states on $\partial\mathcal{S}^*$", we used 10K boundary samples. The critic used to compute the worst saturation used a batch of size 10K and took 50 gradient steps, during which its objective converged. (2) For the metric "% of simulated rollouts that are FI", we used 5K rollouts. To approximate the volume of the safe set, we calculated the percentage of samples in $\mathcal{D}$ falling within $\mathcal{S}^*$, for 1 million samples.

Details for k_{lqr}: We construct an LQR controller to stabilize the full 16D system to the origin in the typical way: we find the linearized system $\dot{x} = Ax + Bu$, let $Q = I_{16\times16}, R = I_{4\times4}$, and compute the linear feedback matrix K. For a nonlinear system such a quadcopter-pendulum, this stabilizing controller only has a small region of attraction about the origin. Thus, it may produce unsafe behavior when initialized further from the origin, so a CBF safeguard is useful.

Details on inverted pendulum volume comparison (from Fig. 4.1): For our toy inverted pendulum problem with a 2D state space and 1D input space, we are curious about how our learned, volume-regularized safe set compares to the largest possible safe set. The largest safe set is the set of all states from which a safe trajectory exists (that is, a trajectory keeping within allowable set $\mathcal{A}$). We can identify most of these states by checking, for every state x_{start} in the two-dimensional domain $\mathcal{D}$, if such a trajectory can be found. Specifically, we pose the following nonlinear program to the

do-mpc Python package [141]:

$$\min_{u(t)} \int_0^T \rho(x(t))\partial t \tag{A2.21}$$

$$\text{s.t. } u(t) \in \mathcal{U} \ , \ \forall t \in [0, T] \tag{A2.22}$$

$$x(0) = x_{start} \tag{A2.23}$$

$$\dot{x}(t) = f(x) + g(x)u(t) \tag{A2.24}$$

where recall that $\mathcal{A}$ is defined as $\{\rho\}_{\leq 0}$ and T is a sufficiently long time horizon. Besides MPC, another way to compute the largest safe set would be to use HJ reachability [1]. However, the problem of finding the largest safe set under input limits is NP-hard, so we can only compute this baseline for our toy inverted pendulum problem and not the higher-dimensional quadcopter-pendulum problem.

Testing robustness to model mismatch and stochastic dynamics:

Noise variance	% FI rollouts
1	99.42
2	99.11
5	93.47
10	85.84

Inertia off by a factor of...	% FI rollouts
0.75	99.66
1.00	99.62
1.25	99.20
1.5	97.51
2	89.18
5	53.33

Table A2.2: (Left) Rollout metrics computed for our learned CBF under stochastic dynamics (when the spread of the zero-mean, Gaussian noise is varied). (Right) Rollout metrics computed for our learned CBF under model mismatch (when the moments of inertia of the quadcopter are off by a factor).

We test whether our learned CBF still ensures safety when our assumption of a known, deterministic system is broken. First, we consider what happens if the model is unknown. Specifically, we consider the case where some model parameters (the quadcopter's moments of inertia) have been misidentified (are all off by a factor). We expect that if the inertia is greater than believed, our learned safe controller will probably intervene too late to save the higher-inertia system. In Table A2.2, we see this is true. Our safe controller becomes increasingly ineffective at preserving safety as the true inertia increases. On the other hand, when inertia is smaller than expected, the system will be easier to save than expected, which means the same level of safety is preserved (compare first and second rows of Table A2.2). Second, we consider what happens if the model is stochastic. We consider a system with additive white Gaussian noise: $\dot{x} = f(x) + g(x)u + w$, with $w \in \mathcal{N}(0, \sigma)$ (0-mean, σ-variance Gaussian). As the variance of the noise increases, the system departs further from its assumed dynamics, and our safe controller fails more and more to ensure safety. Note that we have run rollouts with k_{lqr} (a stabilizing LQR nominal controller).

Elaborating on limitations: We mentioned that we had to perform a change of variables on the states input to the neural CBF before it would train successfully. Specifically, we changed the pendulum's angular velocity to a linear velocity and the quadcopter's angular velocity to the linear

velocity of its vertical body axis. However, we note that after we made this adjustment, the rest of the synthesis required no human intervention.

A3 Appendix for Chapter 5

Definition 2. Following [127], an α**-approximate cover** of Q^o_{free} is a collection of convex sets $\hat{\mathcal{R}}^1_\Delta, \ldots, \hat{\mathcal{R}}^N_\Delta \subset Q^o_{free}$ whose union covers at least an α-fraction of its volume:

$$\text{vol}\left(\bigcup_{i=1}^{N} \hat{\mathcal{R}}^i_\Delta\right) \geq \alpha \cdot \text{vol}(Q^o_{free}) \tag{A3.1}$$

Bibliography

[1] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin, "Hamilton-jacobi reachability: A brief overview and recent advances," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, 2017, pp. 2242–2253.

[2] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler, "Spaceex: Scalable verification of hybrid systems," in *International Conference on Computer Aided Verification*. Springer, 2011, pp. 379–395.

[3] X. Chen, E. Ábrahám, and S. Sankaranarayanan, "Flow*: An analyzer for non-linear hybrid systems," in *International Conference on Computer Aided Verification*. Springer, 2013, pp. 258–263.

[4] M. Althoff, "An introduction to cora 2015," in *Proc. of the workshop on applied verification for continuous and hybrid systems*, 2015, pp. 120–151.

[5] P. S. Duggirala, S. Mitra, M. Viswanathan, and M. Potok, "C2e2: A verification tool for stateflow models," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2015, pp. 68–82.

[6] C. Fan, B. Qi, S. Mitra, M. Viswanathan, and P. S. Duggirala, "Automatic reachability analysis for nonlinear hybrid models with c2e2," in *International Conference on Computer Aided Verification*. Springer, 2016, pp. 531–538.

[7] S. Kong, S. Gao, W. Chen, and E. Clarke, "dreach: δ-reachability analysis for hybrid systems," in *International Conference on TOOLS and Algorithms for the Construction and Analysis of Systems*. Springer, 2015, pp. 200–205.

[8] T. Wei and C. Liu, "Safe control with neural network dynamic models," in *Learning for Dynamics and Control Conference*. PMLR, 2022, pp. 739–750.

[9] A. Clark, "Verification and synthesis of control barrier functions," in *2021 60th IEEE Conference on Decision and Control (CDC)*. IEEE, 2021, pp. 6105–6112.

[10] Y. Lyu, W. Luo, and J. M. Dolan, "Probabilistic safety-assured adaptive merging control for autonomous vehicles," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 10 764–10 770.

[11] C. Liu and M. Tomizuka, "Control in a safe set: Addressing safety in human-robot interactions," in *Dynamic Systems and Control Conference*, vol. 46209. American Society of Mechanical Engineers, 2014, p. V003T42A003.

[12] D. R. Agrawal and D. Panagou, "Safe control synthesis via input constrained control barrier functions," in *2021 60th IEEE Conference on Decision and Control (CDC)*. IEEE, 2021, pp. 6113–6118.

[13] G. Dalal, K. Dvijotham, M. Vecerik, T. Hester, C. Paduraru, and Y. Tassa, "Safe exploration in continuous action spaces," *arXiv preprint arXiv:1801.08757*, 2018.

[14] W. Zhao, T. He, and C. Liu, "Model-free safe control for zero-violation reinforcement learning," in *5th Annual Conference on Robot Learning*, 2021.

[15] A. D. Ames, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," in *53rd IEEE Conference on Decision and Control*. IEEE, 2014, pp. 6271–6278.

[16] K. Garg and D. Panagou, "Control-lyapunov and control-barrier functions based quadratic program for spatio-temporal specifications," in *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE, 2019, pp. 1422–1429.

[17] P. Liu, D. Tateo, H. B. Ammar, and J. Peters, "Robot reinforcement learning on the constraint manifold," in *Conference on Robot Learning*. PMLR, 2022, pp. 1357–1366.

[18] W. S. Cortez and D. V. Dimarogonas, "Safe-by-design control for euler-lagrange systems," *61st IEEE American Controls Conference*, pp. 950–955, 2020.

[19] W. S. Cortez, X. Tan, and D. V. Dimarogonas, "A robust, multiple control barrier function framework for input constrained systems," *IEEE Control Systems Letters*, vol. 6, pp. 1742–1747, 2021.

[20] S. Liu, C. Liu, and J. Dolan, "Safe control under input limits with neural control barrier functions," in *Conference on Robot Learning*. PMLR, 2023, pp. 1970–1980.

[21] T. Wei and C. Liu, "Safe control with neural network dynamic models," in *Learning for Dynamics and Control Conference*. PMLR, 2022, pp. 739–750.

[22] W. Zhao, T. He, T. Wei, S. Liu, and C. Liu, "Safety index synthesis via sum-of-squares programming," *arXiv preprint arXiv:2209.09134*, 2022.

[23] A. Clark, "A semi-algebraic framework for verification and synthesis of control barrier functions," *arXiv preprint arXiv:2209.00081*, 2022.

[24] H. Dai and F. Permenter, "Convex synthesis and verification of control-lyapunov and barrier functions with input constraints," in *2023 American Control Conference (ACC)*. IEEE, 2023, pp. 4116–4123.

[25] S. Kang, Y. Chen, H. Yang, and M. Pavone, "Verification and synthesis of robust control barrier functions: multilevel polynomial optimization and semidefinite relaxation," *arXiv preprint arXiv:2303.10081*, 2023.

[26] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, *et al.*, "Scalable deep reinforcement learning for vision-based robotic manipulation," in *Conference on robot learning*. PMLR, 2018, pp. 651–673.

[27] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine, "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations," *arXiv preprint arXiv:1709.10087*, 2017.

[28] A. Nagabandi, K. Konolige, S. Levine, and V. Kumar, "Deep dynamics models for learning dexterous manipulation," in *Conference on robot learning*. PMLR, 2020, pp. 1101–1112.

[29] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, *et al.*, "Learning dexterous in-hand manipulation," *The International Journal of Robotics Research*, vol. 39, no. 1, pp. 3–20, 2020.

[30] T. Chen, J. Xu, and P. Agrawal, "A system for general in-hand object re-orientation," in *Proceedings of the 5th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, A. Faust, D. Hsu, and G. Neumann, Eds., vol. 164. PMLR, 08–11 Nov 2022, pp. 297–307. [Online]. Available: https://proceedings.mlr.press/v164/chen22a.html

[31] W. Zhou, B. Jiang, F. Yang, C. Paxton, and D. Held, "Hacman: Learning hybrid actor-critic maps for 6d non-prehensile manipulation," *arXiv preprint arXiv:2305.03942*, 2023.

[32] A. Handa, A. Allshire, V. Makoviychuk, A. Petrenko, R. Singh, J. Liu, D. Makoviichuk, K. Van Wyk, A. Zhurkevich, B. Sundaralingam, *et al.*, "Dextreme: Transfer of agile in-hand manipulation from simulation to reality," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 5977–5984.

[33] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," *The International Journal of Robotics Research*, p. 02783649241273668, 2023.

[34] P. Florence, C. Lynch, A. Zeng, O. A. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, and J. Tompson, "Implicit behavioral cloning," in *Conference on robot learning*. PMLR, 2022, pp. 158–168.

[35] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 69–81, 2014.

[36] J.-P. Sleiman, J. Carius, R. Grandia, M. Wermelinger, and M. Hutter, "Contact-implicit trajectory optimization for dynamic object manipulation," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 6814–6821.

[37] A. Ö. Önol, P. Long, and T. Padır, "Contact-implicit trajectory optimization based on a variable smooth contact model and successive convexification," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 2447–2453.

[38] A. Aydinoglu and M. Posa, "Real-time multi-contact model predictive control via admm," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 3414–3421.

[39] S. Le Cleac'h, T. A. Howell, S. Yang, C.-Y. Lee, J. Zhang, A. Bishop, M. Schwager, and Z. Manchester, "Fast contact-implicit model predictive control," *IEEE Transactions on Robotics*, vol. 40, pp. 1617–1629, 2024.

[40] T. Pang and R. Tedrake, "A convex quasistatic time-stepping scheme for rigid multibody systems with contact and friction," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 6614–6620.

[41] H. J. T. Suh, T. Pang, and R. Tedrake, "Bundled gradients through contact via randomized smoothing," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4000–4007, 2022.

[42] V. Kurtz and H. Lin, "Contact-implicit trajectory optimization with hydroelastic contact and ilqr," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 8829–8834.

[43] T. A. Howell, S. L. Cleac'h, J. Brüdigam, J. Z. Kolter, M. Schwager, and Z. Manchester, "Dojo: A differentiable physics engine for robotics," *arXiv preprint arXiv:2203.00806*, 2022.

[44] T. A. Howell, S. Le Cleac'h, S. Singh, P. Florence, Z. Manchester, and V. Sindhwani, "Trajectory optimization with optimization-based dynamics," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 6750–6757, 2022.

[45] T. Pang, H. T. Suh, L. Yang, and R. Tedrake, "Global planning for contact-rich manipulation via local smoothing of quasi-dynamic contact models," *IEEE Transactions on robotics*, vol. 39, no. 6, pp. 4691–4711, 2023.

[46] T. Howell, N. Gileadi, S. Tunyasuvunakool, K. Zakka, T. Erez, and Y. Tassa, "Predictive sampling: Real-time behaviour synthesis with mujoco," *arXiv preprint arXiv:2212.00541*, 2022.

[47] A. H. Li, P. Culbertson, V. Kurtz, and A. D. Ames, "Drop: Dexterous reorientation via online planning," *arXiv preprint arXiv:2409.14562*, 2024.

[48] N. Chavan-Dafle, R. Holladay, and A. Rodriguez, "In-hand manipulation via motion cones," *arXiv preprint arXiv:1810.00219*, 2018.

[49] C. Chen, P. Culbertson, M. Lepert, M. Schwager, and J. Bohg, "Trajectotree: Trajectory optimization meets tree search for planning multi-contact dexterous manipulation," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 8262–8268.

[50] X. Cheng, E. Huang, Y. Hou, and M. T. Mason, "Contact mode guided sampling-based planning for quasistatic dexterous manipulation in 2d," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 6520–6526.

[51] ——, "Contact mode guided motion planning for quasidynamic dexterous manipulation in 3d," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 2730–2736.

[52] T. Marcucci, R. Deits, M. Gabiccini, A. Bicchi, and R. Tedrake, "Approximate hybrid model predictive control for multi-contact push recovery in complex environments," in *2017 IEEE-RAS 17th international conference on humanoid robotics (Humanoids)*. IEEE, 2017, pp. 31–38.

[53] B. Aceituno-Cabezas and A. Rodriguez, "A global quasi-dynamic model for contact-trajectory optimization in manipulation," 2020.

[54] B. P. Graesdal, S. Y. C. Chia, T. Marcucci, S. Morozov, A. Amice, P. A. Parrilo, and R. Tedrake, "Towards tight convex relaxations for contact-rich manipulation," *arXiv preprint arXiv:2402.10312*, 2024.

[55] A. Papachristodoulou and S. Prajna, "A tutorial on sum of squares techniques for systems analysis," in *Proceedings of the 2005, American Control Conference, 2005.* IEEE, 2005, pp. 2686–2700.

[56] Z. Jarvis-Wloszek, R. Feeley, W. Tan, K. Sun, and A. Packard, "Some controls applications of sum of squares programming," in *42nd IEEE international conference on decision and control (IEEE Cat. No. 03CH37475)*, vol. 5. IEEE, 2003, pp. 4676–4681.

[57] E. Lavretsky and K. A. W. Wise, *Robust and Adaptive Control*, ser. Advanced Textbooks in Control and Signal Processing. Springer London, 2012.

[58] X. Xu, P. Tabuada, J. W. Grizzle, and A. D. Ames, "Robustness of control barrier functions for safety critical control," *IFAC-PapersOnLine*, vol. 48, no. 27, pp. 54–61, 2015.

[59] M. Jankovic, "Robust control barrier functions for constrained stabilization of nonlinear systems," *Automatica*, vol. 96, pp. 359–367, 2018.

[60] J. Breeden and D. Panagou, "Robust control barrier functions under high relative degree and input constraints for satellite trajectories," *arXiv preprint arXiv:2107.04094*, 2021.

[61] A. J. Taylor and A. D. Ames, "Adaptive safety with control barrier functions," in *2020 American Control Conference (ACC)*. IEEE, 2020, pp. 1399–1405.

[62] M. Maghenem, A. J. Taylor, A. D. Ames, and R. G. Sanfelice, "Adaptive safety using control barrier functions and hybrid adaptation," in *2021 American Control Conference (ACC)*. IEEE, 2021, pp. 2418–2423.

[63] A. Isaly, O. S. Patil, R. G. Sanfelice, and W. E. Dixon, "Adaptive safety with multiple barrier functions using integral concurrent learning," in *2021 American Control Conference (ACC)*. IEEE, 2021, pp. 3719—3724.

[64] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *2019 18th European control conference (ECC)*. IEEE, 2019, pp. 3420–3431.

[65] T. Wei and C. Liu, "Safe control algorithms using energy functions: A uni ed framework, benchmark, and new directions," in *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE, 2019, pp. 238–243.

[66] T. Wei, S. Kang, W. Zhao, and C. Liu, "Persistently feasible robust safe control by safety index synthesis and convex semi-infinite programming," *IEEE Control Systems Letters*, vol. 7, pp. 1213–1218, 2022.

[67] V. Hamdipoor, N. Meskin, and C. G. Cassandras, "Safe control synthesis using environmentally robust control barrier functions," *European Journal of Control*, p. 100840, 2023.

[68] K. Long, V. Dhiman, M. Leok, J. Cortés, and N. Atanasov, "Safe control synthesis with uncertain dynamics and constraints," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7295–7302, 2022.

[69] W. Xiao, C. Belta, and C. G. Cassandras, "Adaptive control barrier functions," *IEEE Transactions on Automatic Control*, vol. 67, no. 5, pp. 2267–2281, 2021.

[70] H. Parwana and D. Panagou, "Recursive feasibility guided optimal parameter adaptation of differential convex optimization policies for safety-critical systems," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 6807–6813.

[71] Y. Lyu, W. Luo, and J. Dolan, "Probabilistic safety-assured adaptive merging control for autonomous vehicles," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021.

[72] M. Althoff, O. Stursberg, and M. Buss, "Computing reachable sets of hybrid systems using a combination of zonotopes and polytopes," *Nonlinear analysis: hybrid systems*, vol. 4, no. 2, pp. 233–249, 2010.

[73] M. Althoff, "Reachability analysis of nonlinear systems using conservative polynomialization and non-convex sets," in *Proceedings of the 16th international conference on Hybrid systems: computation and control*, 2013, pp. 173–182.

[74] B. T. Lopez and J.-. E. Slotine, "Unmatched control barrier functions: Certainty equivalence adaptive safety," in *2023 American Control Conference (ACC)*. IEEE, 2023, pp. 3662–3668.

[75] P. A. Parrilo, *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*. California Institute of Technology, 2000.

[76] G. Stengle, "A nullstellensatz and a positivstellensatz in semialgebraic geometry," *Mathematische Annalen*, vol. 207, pp. 87–97, 1974.

[77] G. B. Dantzig, "Linear programming," *Operations research*, vol. 50, no. 1, pp. 42–47, 2002.

[78] C. Chen, "On the robustness of the linear quadratic regulator via perturbation analysis of the riccati equation," Ph.D. dissertation, Dublin City University, 2015.

[79] A. Papachristodoulou, J. Anderson, G. Valmorbida, S. Prajna, P. Seiler, P. A. Parrilo, M. M. Peet, and D. Jagt, *SOSTOOLS: Sum of squares optimization toolbox for MATLAB*, `http://arxiv.org/abs/1310.4716`, 2021. [Online]. Available: https://github.com/oxfordcontrol/SOSTOOLS

[80] M. ApS, *The MOSEK optimization toolbox for MATLAB manual. Version 9.0.*, 2019. [Online]. Available: http://docs.mosek.com/9.0/toolbox/index.html

[81] S. Prajna, P. A. Parrilo, and A. Rantzer, "Nonlinear control synthesis by convex optimization," *IEEE Transactions on Automatic Control*, vol. 49, no. 2, pp. 310–314, 2004.

[82] C. C. Chung and J. Hauser, "Nonlinear control of a swinging pendulum," *Automatica*, vol. 31, no. 6, pp. 851–862, 1995.

[83] M. Posa, M. Tobenkin, and R. Tedrake, "Stability analysis and control of rigid-body systems with impacts and friction," *IEEE Transactions on Automatic Control*, vol. 61, no. 6, pp. 1423–1437, 2016.

[84] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-13, no. 5, pp. 834–846, 1983.

[85] T. Luukkonen, "Modelling and control of quadcopter," Ph.D. dissertation, School of Science at Aalto University, 2011.

[86] P. Ventura Diaz and S. Yoon, "High-fidelity computational aerodynamics of multi-rotor unmanned aerial vehicles," in *2018 AIAA Aerospace Sciences Meeting*, 2018, p. 1266.

[87] B. Xu and K. Sreenath, "Safe teleoperation of dynamic uavs through control barrier functions," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 7848–7855.

[88] A. Singletary, K. Klingebiel, J. Bourne, A. Browning, P. Tokumaru, and A. Ames, "Comparative analysis of control barrier functions and artificial potential fields for obstacle avoidance," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 8129–8136.

[89] R. Liu, R. Chen, Y. Sun, Y. Zhao, and C. Liu, "Jerk-bounded position controller with real-time task modification for interactive industrial robots," in *2022 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, 2022, pp. 1771–1778.

[90] A. D. Ames, E. A. Cousineau, and M. J. Powell, "Dynamically stable bipedal robotic walking with nao via human-inspired hybrid zero dynamics," in *Proceedings of the 15th ACM international conference on Hybrid Systems: Computation and Control*, 2012, pp. 135–144.

[91] F. Blanchini, "Set invariance in control," *Automatica*, vol. 35, no. 11, pp. 1747–1767, 1999.

[92] J.-P. Aubin, A. M. Bayen, and P. Saint-Pierre, *Viability theory: new directions*. Springer Science & Business Media, 2011.

[93] D. Bertsekas, "Infinite time reachability of state-space regions by using feedback control," *IEEE Transactions on Automatic Control*, vol. 17, no. 5, pp. 604–613, 1972.

[94] Y. Chen, M. Jankovic, M. Santillo, and A. D. Ames, "Backup control barrier functions: Formulation and comparative study," in *2021 60th IEEE Conference on Decision and Control (CDC)*. IEEE, 2021, pp. 6835–6841.

[95] S. M. Richards, F. Berkenkamp, and A. Krause, "The lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems," in *Conference on Robot Learning*. PMLR, 2018, pp. 466–476.

[96] H. Ravanbakhsh and S. Sankaranarayanan, "Learning control lyapunov functions from counterexamples and demonstrations," *Autonomous Robots*, vol. 43, no. 2, pp. 275–307, 2019.

[97] Y.-C. Chang, N. Roohi, and S. Gao, "Neural lyapunov control," *Advances in neural information processing systems*, vol. 32, 2019.

[98] C. Dawson, Z. Qin, S. Gao, and C. Fan, "Safe nonlinear control using robust neural lyapunov-barrier functions," in *Conference on Robot Learning*. PMLR, 2022, pp. 1724–1735.

[99] W. Xiao and C. Belta, "Control barrier functions for systems with high relative degree," in *2019 IEEE 58th conference on decision and control (CDC)*. IEEE, 2019, pp. 474–479.

[100] A. Shafahi, M. Najibi, M. A. Ghiasi, Z. Xu, J. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein, "Adversarial training for free!" *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[101] E. Wong, L. Rice, and J. Z. Kolter, "Fast is better than free: Revisiting adversarial training," *arXiv preprint arXiv:2001.03994*, 2020.

[102] H. Kim, W. Lee, and J. Lee, "Understanding catastrophic overfitting in single-step adversarial training," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 8119–8127.

[103] R. Figueroa, A. Faust, P. Cruz, L. Tapia, and R. Fierro, "Reinforcement learning for balancing a flying inverted pendulum," in *Proceeding of the 11th World Congress on Intelligent Control and Automation*. IEEE, 2014, pp. 1787–1793.

[104] M. Hehn and R. D'Andrea, "A flying inverted pendulum," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 763–770.

[105] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[106] O. M. Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu, *et al.*, "Octo: An open-source generalist robot policy," *arXiv preprint arXiv:2405.12213*, 2024.

[107] N. M. M. Shafiullah, A. Rai, H. Etukuru, Y. Liu, I. Misra, S. Chintala, and L. Pinto, "On bringing robots home," *arXiv preprint arXiv:2311.16098*, 2023.

[108] M. Janner, Y. Du, J. B. Tenenbaum, and S. Levine, "Planning with diffusion for flexible behavior synthesis," *arXiv preprint arXiv:2205.09991*, 2022.

[109] H. Zhu, T. Zhao, X. Ni, J. Wang, K. Fang, L. Righetti, and T. Pang, "Should we learn contact-rich manipulation policies from sampling-based planners?" *IEEE Robotics and Automation Letters*, vol. 10, no. 6, pp. 6248–6255, 2025.

[110] X. Li, T. Zhao, X. Zhu, J. Wang, T. Pang, and K. Fang, "Planning-guided diffusion policy learning for generalizable contact-rich bimanual manipulation," *arXiv preprint arXiv:2412.02676*, 2024.

[111] R. Natarajan, G. L. Johnston, N. Simaan, M. Likhachev, and H. Choset, "Torque-limited manipulation planning through contact by interleaving graph search and trajectory optimization," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 8148–8154.

[112] A. Shkolnik, M. Walter, and R. Tedrake, "Reachability-guided sampling for planning under differential constraints," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 2859–2865.

[113] A. Wu, S. Sadraddini, and R. Tedrake, "R3t: Rapidly-exploring random reachable set tree for optimal kinodynamic planning of nonlinear hybrid systems," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 4245–4251.

[114] G. Lee, T. Lozano-Pérez, and L. P. Kaelbling, "Hierarchical planning for multi-contact non-prehensile manipulation," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 264–271.

[115] B. Aceituno and A. Rodriguez, "A hierarchical framework for long horizon planning of object-contact trajectories," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 189–196.

[116] X. Cheng, S. Patil, Z. Temel, O. Kroemer, and M. T. Mason, "Enhancing dexterity in robotic manipulation via hierarchical contact exploration," *IEEE Robotics and Automation Letters*, vol. 9, no. 1, pp. 390–397, 2023.

[117] T. Lozano-Pérez and L. P. Kaelbling, "A constraint-based method for solving sequential manipulation planning problems," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 3684–3691.

[118] M. A. Toussaint, K. R. Allen, K. A. Smith, and J. B. Tenenbaum, "Differentiable physics and stable modes for tool-use and manipulation planning," 2018.

[119] H. Zhu, A. Meduri, and L. Righetti, "Efficient object manipulation planning with monte carlo tree search," in *2023 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2023, pp. 10 628–10 635.

[120] R. Tedrake, "Lqr-trees: Feedback motion planning on sparse randomized trees," 2009.

[121] A. Majumdar and R. Tedrake, "Funnel libraries for real-time robust feedback motion planning," *The International Journal of Robotics Research*, vol. 36, no. 8, pp. 947–982, 2017.

[122] H. Suh, T. Pang, T. Zhao, and R. Tedrake, "Dexterous contact-rich manipulation via the contact trust region," *arXiv preprint arXiv:2505.02291*, 2025.

[123] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *2009 IEEE international conference on robotics and automation*. IEEE, 2009, pp. 489–494.

[124] T. Marcucci, J. Umenberger, P. Parrilo, and R. Tedrake, "Shortest paths in graphs of convex sets," *SIAM Journal on Optimization*, vol. 34, no. 1, pp. 507–532, 2024.

[125] T. Marcucci, M. Petersen, D. von Wrangel, and R. Tedrake, "Motion planning around obstacles with convex optimization," *Science robotics*, vol. 8, no. 84, p. eadf7843, 2023.

[126] S. Morozov, T. Marcucci, B. P. Graesdal, A. Amice, P. A. Parrilo, and R. Tedrake, "Mixed discrete and continuous planning using shortest walks in graphs of convex sets," *arXiv preprint arXiv:2507.10878*, 2025.

[127] P. Werner, T. Cohn, R. H. Jiang, T. Seyde, M. Simchowitz, R. Tedrake, and D. Rus, "Faster algorithms for growing collision-free convex polytopes in robot configuration space," *arXiv preprint arXiv:2410.12649*, 2024.

[128] H. Zhu, A. Gupta, A. Rajeswaran, S. Levine, and V. Kumar, "Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 3651–3657.

[129] A. Robey, H. Hu, L. Lindemann, H. Zhang, D. V. Dimarogonas, S. Tu, and N. Matni, "Learning control barrier functions from expert demonstrations," in *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE, 2020, pp. 3717–3724.

[130] N. Boffi, S. Tu, N. Matni, J.-J. Slotine, and V. Sindhwani, "Learning stability certificates from data," in *Proceedings of the 2020 Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, J. Kober, F. Ramos, and C. Tomlin, Eds., vol. 155, 16–18 Nov 2021, pp. 1341–1350.

[131] Z. Qin, K. Zhang, Y. Chen, J. Chen, and C. Fan, "Learning safe multi-agent control with decentralized neural barrier certificates," *arXiv preprint arXiv:2101.05436*, 2021.

[132] Y. Meng, Z. Qin, and C. Fan, "Reactive and safe road user simulations using neural barrier certificates," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 6299–6306.

[133] W. Xiao, R. Hasani, X. Li, and D. Rus, "Barriernet: A safety-guaranteed layer for neural networks," *arXiv preprint arXiv:2111.11277*, 2021.

[134] C. Wang, Y. Meng, Y. Li, S. L. Smith, and J. Liu, "Learning control barrier functions with high relative degree for safety-critical control," in *2021 European Control Conference (ECC)*. IEEE, 2021, pp. 1459–1464.

[135] J. Sill, "Monotonic networks," *Advances in neural information processing systems*, vol. 10, 1997.

[136] H. Narayanan and P. Niyogi, "Sampling hypersurfaces through diffusion," in *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*. Springer, 2008, pp. 535–548.

[137] R. W. Beard, "Quadrotor dynamics and control," *Brigham Young University*, vol. 19, no. 3, pp. 46–56, 2008.

[138] P. Autopilot, "Px4 user guide: jmavsim with sitl," 2022. [Online]. Available: https://docs.px4.io/master/en/simulation/jmavsim.html

[139] N. Hansen, "The cma evolution strategy: A tutorial," *arXiv preprint arXiv:1604.00772*, 2016.

[140] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

[141] S. Lucia, A. Tătulea-Codrean, C. Schoppmeyer, and S. Engell, "Rapid development of modular and sustainable nonlinear model predictive control solutions," *Control Engineering Practice*, vol. 60, pp. 51–62, 2017.