2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Reachable Predictive Control: A Novel Control Algorithm for Nonlinear Systems with Unknown Dynamics and its Practical Applications*

Taha Shafa, Yiming Meng, and Melkior Ornik

Abstract— **This paper proposes an algorithm capable of driving a system to follow a piecewise linear trajectory without prior knowledge of the system dynamics. Motivated by a critical failure scenario in which a system can experience an abrupt change in its dynamics, we demonstrate that it is possible to follow a set of waypoints comprised of states analytically proven to be reachable despite not knowing the system dynamics. The proposed algorithm first applies small perturbations to locally learn the system dynamics around the current state, then computes the set of states that are provably reachable using the locally learned dynamics and their corresponding maximum growth-rate bounds, and finally synthesizes a control action that navigates the system to a guaranteed reachable state.**

I. Introduction

Many real-world systems must operate in environments where their exact dynamics are unknown, yet reliable motion toward safety-critical or mission-critical goals must still be achieved. In such settings, even modest uncertainty in the dynamics can render conventional model-based planning or tracking strategies ineffective. In this paper, we aim to simulate scenarios where the existing system model is ineffective due to an abrupt changes in its dynamics. This could be due to physical failures or environmental disturbances.

We address this challenge by proposing *Reachable Predictive Control* (RPC), a resilient framework for safe navigation. RPC proceeds iteratively: (1) update local dynamics through active learning, (2) compute a Guaranteed Reachable Set (GRS) with limited information and assign a feasible waypoint, (3) synthesize a controller to reach that waypoint, and (4) repeat to steer the system toward a provably reachable region. Unlike classical robust safety control, which relies on nominal models perturbed by mismatches or disturbances, RPC assumes only a local Lipschitz bound on the dynamics, encompassing a broader class of systems.

We review some crucial results that are pertinent to the work presented in this paper. In the realm of reachability-based robust safety control, frameworks developed around Control Barrier Functions (CBFs) and Model Predictive Control (MPC) represent the most widely adopted approaches.

CBFs are widely used to enforce safety in dynamical systems, often as filters that minimally adjust control inputs to prevent constraint violations. While elegant in theory, constructing valid CBFs is difficult, though integration with data-driven controllers is attractive. Recent work has explored neural-network-based barrier learning through policy evaluation and Hamilton–Jacobi reachability [1]–[3], extending to parametric uncertainties and dynamic environments. These methods, however, face challenges in dynamics assumptions, verification complexity, and interpretability. The approach in [4] eases reliance on precise dynamics and proposes runtime CBF approximations, but without provable safety guarantees.

MPC enforces safety by embedding state and input constraints into a finite-horizon problem solved in receding-horizon form [5], [6]. Robust variants such as tube-based [7] and min-max MPC [8] handle uncertainty when a nominal model is available. Extensions combine MPC with CBFs [9] or reachability-based safety certification [10]. Recent work leverages meta-learning for system identification and Koopman-based models to adapt to nonlinear systems with parametric uncertainties [11]–[13]. Despite these advances, MPC still suffers from finite-horizon approximations, the need for long horizons, and heavy computational demands in nonlinear or high-dimensional settings.

Learning-based Safety Control, particularly reinforcement learning (RL), has shown success in generating robust control policies for high-dimensional dynamics, sensory feedback, and complex tasks [14]–[16], including agility, robustness, and terrain traversal [17]–[19]. However, many RL approaches emphasize performance over safety—whether through motion planner integration or collision penalties [20]–[22], and thus inherit model-based drawbacks, restrict mobility, and lack formal safety guarantees. the resulting policies often exhibit degraded safety in practice, especially under distribution shifts away from the training environment.

Unlike prior work, the proposed RPC framework uses reachability analysis with provable guarantees, requiring neither a nominal model nor pre-collected data, enabling real-time response to a broader class of uncertainties.

A. Contributions

We formally present the RPC algorithm, an algorithm capable of solving reach-avoid and trajectory-tracking problems without the use of a nominal model. Previous work estimated the GRS using proxy systems [23]–[25], later incorporating myopic control [26] and developing synthesis methods [27], but these pipelines could not handle practical, underactuated systems. We generalize results to incorporate underactuated systems, provide a new algorithm capable of optimal path-following under constraints, and demonstrate its operation in an unknown environment via simulation.

*This work was supported in part by NASA grant 80NSSC22M0070 and the Air Force Office of Scientific Research under award number FA9550-23-1-0131.

Taha Shafa, Yiming Meng, and Melkior Ornik are with the Department of Aerospace Engineering and the Coordinated Science Laboratory, University of Illinois Urbana-Champaign, Urbana, USA. tahaas2, ymmeng, mornik@illinois.edu

II. PRELIMINARIES

In this section, we first introduce the dynamical systems under consideration and detail, through assumptions, what information can be utilized in place of nominal system dynamics. We then state the robust planning problem with the definition of the GRS. As a means of computing the GRS and synthesizing the corresponding control of the reachable path, we summarize the resilient reachability problem and the reachable predictive control framework, which together provide a solution to the stated reachability problem.

A. Unknown Nonlinear System Structure

We begin by considering a general autonomous, underactuated, nonlinear control-affine dynamical system of the abstract form described as

$$\dot{x}(t) = f(x(t)) + G(x(t))u(t), \quad x(0) = x_0, \tag{1}$$

which can be decomposed as

$$\dot{\tilde{x}}(t) = \tilde{f}(\tilde{x}(t)) + \tilde{G}(\tilde{x}(t))u, \qquad \tilde{x}(0) = \tilde{x}_0 \tag{2a}$$

$$\dot{\overline{x}}(t) = \overline{f}(x(t)) = F(x(t))\tilde{x}(t), \qquad \overline{x}(0) = \overline{x}_0 \tag{2b}$$

such that $x(t) = \begin{bmatrix}\tilde{x}(t) & \overline{x}(t)\end{bmatrix}^\mathsf{T}$ where all $t \geq 0$, $x(t) \in \mathcal{X} \subseteq \mathbb{R}^n$, $\tilde{x}(t) \in \mathbb{R}^m$, $\overline{x}(t) \in \mathbb{R}^{n-m}$. The admissible control inputs satisfy $u(t) \in \mathcal{U} = \mathbb{B}^m(0;1)$. The mappings $f : \mathbb{R}^n \to \mathbb{R}^n$ and $G : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ that are differentiable almost everywhere and local Lipschitz on a neighborhood $\mathcal{N}$ of any $x_0 \in \mathbb{R}^n$, with local Lipschitz constants $L_f^{\mathcal{N}}$ and $L_G^{\mathcal{N}}$. This implies that $\tilde{f}$ and $\overline{f}$ are locally Lipschitz with constant $L_f^{\mathcal{N}}$ as well. In practice, $L_f^{\mathcal{N}}$ can be proved to be a valid Lipschitz bound for $F(x(t))$ as well; additional analysis and discussion can be found Lemma 2 of Appendix I. A solution to (1) with control input u and initial value x_0 is denoted by $\phi_u(\cdot, x_0)$. Notice that subsystem (2a) is not dependent on subsystem (2b), which is consistent with many practical system models for systems like unmanned drones and ground vehicles [28], [29]. The decoupled structure of (2a) is required for the results presented in Section III; generalizing the structure to all control-affine systems remains for future work.

B. Problem Formulation

We begin by formalizing the strict control design constraints for the nonlinear control of system (1).

Condition 1: There is no known nominal model for system (1) and its corresponding subsystem (2). □

While we may not have access to a viable dynamics model, work in [26], [27] demonstrates how piecewise-linear control affine inputs can be used to determine $f(x_0)$ and $G(x_0)$ using trajectory data from a single system run. Since we are limited with no model or a priori trajectory data, our control-oriented learning method is constrained to the following condition.

Condition 2: There is only one system run, starting from a predetermined initial state x_0. Controller synthesis must be performed solely using trajectory information available during that run. □

Under the conditions above, our aim is to present a solution to the following problem.

Problem 1 (Reach-avoid problem): Let $x_0 \in \mathcal{X}$ and $T \geq 0$. Consider the target se $\mathcal{T} \subseteq \mathcal{X}$ and the unsafe (blocking) set $\mathcal{B} \subseteq \mathcal{X}$, respectively. Find, if it exists, a control signal $u^* : [0, T] \to \mathcal{U}$ such that the following holds:

i) $\phi_{u^*}(t, x_0) \notin \mathcal{B}$ for all $t \in [0, T]$;
ii) there exists $0 \leq T'_u \leq T$ such that $\phi_{u^*}(t, x_0) \in \mathcal{T}$ for all $t \in [T'_{u^*}, T]$;
iii) T'_u from ii) is the minimal such value for all the control laws $u : [0, T] \to \mathcal{U}$ which satisfy i) and ii). □

A previously developed myopic control algorithm [26] aimed to solve Problem 1, however myopic control lacks guarantees. Specifically, it can synthesize control action aimed at reaching an unattainable state, which could cause system (1) to inadvertently follow a trajectory which intersects $\mathcal{B}$, thus violating i) in Problem 1. To avoid this possibility, present the following additional problem and propose a solution in subsequent sections.

Problem 2 (Trajectory Tracking): Let $\hat{\phi}_u(\cdot, x_0) : [0, \infty) \to \mathbb{R}^n$ be a piecewise linear trajectory with its time derivatives bounded and let $r > 0$. Design a controller such that $\|\phi_u(t, x_0) - \hat{\phi}_u(t, x_0)\| < r$ for every t. □

We detail through the following assumption what information we can utilize to solve Problems 1 and 2 under Conditions 1 and 2.

Assumption 1: The bounds $L_f^{\mathcal{N}}$ and $L_G^{\mathcal{N}}$ are known for some compact neighborhood $\mathcal{N}$ of any $x \in \mathbb{R}^n$ as well as values $f(x_0)$ and $G(x_0)$ such that $G(x_0) \neq 0$. □

This mild assumption is reasonable in practice, as it aligns with the physical limits of the controlled object and captures its inherently bounded response to changes in state and control inputs. For information on gathering the assumed knowledge, we refer the reader to related literature [26], [30]. With this knowledge, we want to determine the states we can reach by underapproximating the *guaranteed reachable set*.

Definition 1 (Guaranteed Reachable Set): Let us denote $\mathcal{D}_{\text{con}}$ as the set of all pairs (f, G) such that $\|f(x) - f(x_0)\| \leq L_f^{\mathcal{N}}\|x - x_0\|$ and $\|G(x) - G(x_0)\| \leq L_G^{\mathcal{N}}\|x - x_0\|$. The Guaranteed Reachable Set (GRS) is the set of all states that can be reached by every dynamic pair $(f, G) \in \mathcal{D}_{\text{con}}$ for system (1) at time T. □

The rest of the paper introduces an algorithm that solves the trajectory-tracking and reach-avoid problems for a nonlinear control-affine system, requiring knowledge of the dynamics solely through the information provided in Assumption 1. Specifically, to form a key step of the RPC framework and to extract, to the largest extent, the information provided by Assumption 1, we need to construct a proxy system whose reachable sets are contained within that of the original system (1). The next section derives this proxy system and presents novel results that generalize our previous analyses to underactuated systems.

C. Vehicle Dynamics

We aim to demonstrate the capabilities of the proposed approach using an unmanned vehicle and now present the dynamics model employed in the simulation. We emphasize that the vehicle dynamics are solely used to mirror realistic

conditions and physical constraints in simulation, while our proposed algorithm follows a desired trajectory without knowledge of the described model.

We implement a kinematic bicycle model, frequently used in the state-of-the-art [31]–[33], written as:

$$\dot{x} = v\cos(\theta),\ \dot{y} = v\sin(\theta) \tag{3a}$$

$$\dot{\theta} = \frac{v}{l_r}\tan(u_2),\ \dot{v} = \min(u_1, \mu g) - r_c g \tag{3b}$$

to simulate vehicle motion. In (3), $(x, y) \in \mathbb{R}^2$ are the position coordinates of the center of mass (CoM), θ is the yaw angle, v is the linear speed of the vehicle, l_f and l_r are the distance from the CoM to the front and rear axles respectively, r_c is the rolling resistance coefficient, μ is the coefficient of surface friction, g is the gravitational constant, u_1 is the acceleration of the CoM, u_2 is the front wheel steering angle. Notice that system (3) is not control-affine and thus seemingly violates our assumptions. In practice, this is not an issue, as a locally reduced version of the original system can be found that satisfies the assumed dynamics. Its GRS is contained within that of the original system, providing a conservative but acceptable estimate that keeps the synthesized control paths within a safe region. Further discussion is given in Section V.

III. Proxy System Dynamics

To perform trajectory tracking and obstacle avoidance for an unknown nonlinear system, we first take the knowledge from Assumption 1 to derive proxy systems whose reachable sets are contained in the GRS. Then, following the pipeline described in the introduction, we use the proxy system to conservatively estimate the reachable range, identify a relatively safe waypoint within the GRS based on that estimate, and generate reachable paths to be followed. Controller synthesis designed to follow these paths is presented in the next section, but first, for the convenience of the reader, we revisit the underapproximated proxy control system for the GRS evaluation [23, Theorem 1], which we can use to provide a set of reachable states for subsystem (2a).

Theorem 1: Let $\mathcal{U}$, $L_f^{\mathcal{N}}$, and $L_G^{\mathcal{N}}$ be given. Then, there exists a $\hat{u} \in \mathcal{U}$ satisfying

$$ku = \tilde{f}(\tilde{x}) - \tilde{f}(\tilde{x}_0) + \tilde{G}(\tilde{x})\hat{u} \tag{4}$$

for all $\tilde{x} \in \mathbb{B} := \{\tilde{x} : \|\tilde{x}\| \leq \|\tilde{G}^\dagger(\tilde{x}_0)\|^{-1}/(L_f^{\mathcal{N}} + L_G^{\mathcal{N}})\}$, any $u \in \mathbb{B}^m(0; 1)$, and any k such that $|k| \leq \|\tilde{G}^\dagger(\tilde{x}_0)\|^{-1} - (L_f^{\mathcal{N}} + L_G^{\mathcal{N}})\|\tilde{x}\|$. □

Consequently, by [23, Theorem 3], an underapproximation of the GRS of (2a) can be calculated based on a proxy equation of the form

$$\dot{\tilde{x}}_p(t) = a + (b - c\|\tilde{x}_p(t)\|)u(t),\ \ \tilde{x}_p(0) = \tilde{x}_0, \tag{5}$$

on the domain $\mathbb{B}$, where $a = \tilde{f}(\tilde{x}_0)$, $b := \|\tilde{G}^\dagger(\tilde{x}_0)\|^{-1}$, and $c := L_f^{\mathcal{N}} + L_G^{\mathcal{N}}$. To identify a complete underapproximation of the GRS of system (1), we need to extend these results to include the states $\overline{x}(t)$ of (2b). We begin by presenting a generalization of [23, Theorem 1] to systems with input sets $\mathcal{U}$ such that $\mathcal{U} = \mathbb{B}^m(a; b(t))$.

Theorem 2: Let $\mathcal{U} = \mathbb{B}^m(a; b(t))$ where $a \in \mathbb{R}^m$ such that $b(t) \leq \|a\|$ for all t and $c(t) = \|a\| - b(t)$. Let $L_f^{\mathcal{N}}$, and $L_G^{\mathcal{N}}$ be defined as above. Let $x \in \mathbb{R}^n$ satisfy $(L_f^{\mathcal{N}} + L_G^{\mathcal{N}})\|x\| < \|G^\dagger(x_0)\|^{-1}$. Define $\overline{\mathcal{V}}_x^{\mathcal{G}} = \mathbb{B}^n\left(f(x_0); c(t)\left(\|G^\dagger(x_0)\|^{-1} - (L_f^{\mathcal{N}} + L_G^{\mathcal{N}})\|x\|\right)\right) \cap \mathrm{Im}(G(x_0))$. Then, $\overline{\mathcal{V}}_x^{\mathcal{G}} \subseteq \mathcal{V}_x^{\mathcal{G}}$. □

[1] Leveraging this fact, we next present an underestimate of the reachable set for the actuated part of the system, building upon its connection to another system whose reachable set is a ball contained within the reachable set of (5).

Lemma 1: Consider control systems $\dot{x}(t) = a + (b - c\|x(t) - x(0)\|)u$ and $\dot{z}(t) = (b - c\|x(t) - x(0)\| - \|a\|)u$, both defined on some ball $\mathbb{B} \subset \mathbb{R}^m$ with $a, x \in \mathbb{R}^m$, $b, c \in \mathbb{R}$, and $u \in \mathcal{U}$. Let $\mathcal{R}_x(T, x_0)$ and $\mathcal{R}_z(T, x_0)$ denote their reachable sets, respectively. Suppose the same assumption holds as in Theorem 2. Then, $\mathcal{R}_z(T, x_0) \subseteq \mathcal{R}_x(T, x_0)$. Additionally, $\mathcal{R}_z(T, x_0)$ is a ball in $\mathbb{B} \subset \mathbb{R}^m$. □

By immediate virtue of Lemma 1, and using notation consistent with system (5), we identify the proxy system

$$\dot{\tilde{x}}_p(t) = (\tilde{b} - c\|\tilde{x}_p(t)\|)u(t),\ \ \tilde{x}_p(0) = \tilde{x}_0, \tag{6}$$

where $\tilde{b} = b - \|a\|$, whose reachable set forms a ball contained within the GRS of (2a). Let $\mathbb{B}^m(\tilde{x}_0, b(t))$ be the reachable set of (6). By following similar steps as above, a straightforward application of Theorem 2 using $\mathbb{B}^m(\tilde{x}_0, b(t))$ yields the following result.

Theorem 3: Consider a control system of the form

$$\dot{\tilde{x}} = h(\|x\|)v, \quad x(0) = x_0 \tag{7}$$

on $\{x \mid \|x - x_0\| \leq \|F^\dagger(x_0)\|/L_f^{\mathcal{N}}, \|\tilde{x}_0\| \geq \|\tilde{x}_0 - \tilde{x}(t)\|\}$, with $v \in \mathbb{B}^m(\tilde{x}_0, b(t))$ and where $h(\|x\|) = (\|\tilde{x}_0\| - \|\tilde{x}_0 - \tilde{x}(t)\|)(\|F^\dagger(x_0)\| - L_f^{\mathcal{N}}\|x - x_0\|)$ if $\|x - x_0\| \leq L_f^{\mathcal{N}}/\|x - x_0\|$. The reachable set of (7) is contained in the GRS of (2b). □

We naturally arrive at the following conclusion.

Theorem 4: Let $\mathcal{R}^{\mathcal{G}}(T, x_0)$ be the GRS of system (1). If $\mathcal{R}_{\tilde{x}}(T, \tilde{x}_0)$ and $\mathcal{R}_{\overline{x}}(T, x_0)$ are the reachable sets of (5) and (7), and $\overline{\mathcal{R}}_x(T, x_0) = \mathcal{R}_{\tilde{x}}(T, \tilde{x}_0) \cup \mathcal{R}_{\overline{x}}(T, x_0)$, then $\overline{\mathcal{R}}_x(T, x_0) \subseteq \mathcal{R}^{\mathcal{G}}(T, x_0)$. □

Theorem 4 provides a complete set of reachable states for all states in system (1) using solely the information in Assumption 1. This is achieved by first computing a decomposed GRS estimation for (2a) under the given input bound, and then using this estimated GRS as the input bound for (2b), thereby obtaining the complete set of reachable states. In the next section, we detail how $\mathcal{R}_{\tilde{x}}(T, \tilde{x}_0)$ is used in controller synthesis. Then, in our practical demonstration, we will show how $\mathcal{R}_{\overline{x}}(T, x_0)$ informs high-level decision making, that is, informs which reachable paths to follow.

IV. Controller Synthesis

We now introduce an algorithm, originally presented in [27], that synthesizes control inputs designed to follow piecewise-linear trajectories for system (2a). In view of Theorem 1 and [23, Theorem 3], for any path generated by the proxy system (5), there exists a control signal u for (2a)

[1] For proofs of Theorem 2 and Lemma 1, we direct the reader to https://arxiv.org/abs/2510.02623

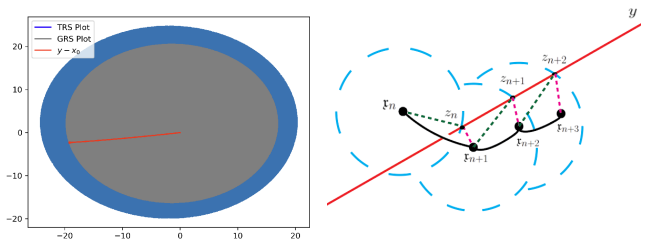

Fig. 1: Trajectory following for unknown systems where the line (red) represents the desired reachable linear path contained within the GRS (gray). The $\mathfrak{x}_n = \tilde{x}(\tau_n)$ denotes the begining point of each learning cycle, while z_n represents the sequence of points automatically generated by Algorithm 1, each located an r-distance apart and progressively approaching $y := \tilde{x}_f$, toward which the system state converges within each learning cycle.

that ensures the system follows the same path. We detail the design of control inputs for following such reachable paths.

A. Control Design

The results in [27] can be summarized as follows. For sampling period $\delta t > 0$, each learning cycle has length $\tau = (m+1)\delta t$ with start time $\tau_n := n\tau$. Within $[\tau_n + j\delta t, \tau_n + (j+1)\delta t)$ for each $j \in \{1, 2, \cdots, m\}$, piecewise constant controls (sample-and-hold) are applied as

$$u_{n,j} := u_{n,0} + \Delta u_j, \tag{8}$$

where $\Delta u_j := \pm \epsilon e_j$ generates local excitation along orthonormal unit vectors with a small amplitude $\epsilon > 0$ for dynamics learning. $\{u_{n,0}\}$ initializes each cycle and steers the state toward points $\{z_n\}$ on the reference path. Each z_n is designed and dynamically updated to be placed at an r-distance from each cycle's endpoint and to monotonically approaching $\tilde{x}_f$ along the reference path (see Fig. 1 for an illustration), where $r = r(k, \delta t) = \sup_u |\tilde{x}_p(k(m+1)\delta t)|$.

In determining $u_{n,0}$, we choose $u_{0,0} = (1-\epsilon)\frac{\tilde{G}^\dagger(\tilde{x}_0)(\tilde{x}_f - \tilde{x}_0)}{\|\tilde{G}^\dagger(\tilde{x}_0)\||\tilde{x}_f - \tilde{x}_0|}$ which pushes the state $\tilde{x}$ directly toward $\tilde{x}_f$. For each $n \geq 1$, parameters k, ϵ, and δt are tuned so that the above described sequence above-described $\{z_n\}$ is generated automatically, and $u_{n,0}$ ensures $\dot{d}_{z_n}(\tilde{x}(t)) < 0$ for all $t \in [\tau_n, \tau_{n+1})$, where $d_{z_n}(\tilde{x}) = \|\tilde{x} - z_n\|^2$. The work [27] established that the mechanism holds under a sufficient condition

$$\mathcal{E}_n(\delta t, \epsilon) + \mathcal{E}_\mu(\delta t, \epsilon) \leq r(b - c\|\tilde{x}(\tau_n)\| - |a| - \mathcal{E}_r(\delta t, \epsilon) + \epsilon M_0), \tag{9}$$

where $-r(b - c\|\tilde{x}(\tau_n)\| - \|a\|)$ is the guaranteed minimal force of $\dot{d}_{z_n}$ that an optimal control can generate at τ_n, offset by perturbations (with the satisfaction of (9) ensuring $\dot{d}_{z_n}$ remains negative throughout each learning cycle). The perturbation terms have the following heuristic interpretation (see Appendix I for details):

- $\mathcal{E}_r(\delta t, \epsilon)$ (due to estimation error of r.h.s. of (2a)),
- $\mathcal{E}_n(\delta t, \epsilon)$ (due to estimation error of $\tilde{x}$),
- ϵM_0 (effect of myopic control), and
- $\mathcal{E}_\mu(\delta t, \epsilon)$ (deviation due to suboptimal control).

Particularly, the term $\mathcal{E}_\mu(\delta t, \epsilon)$ quantifies the performance gap between this suboptimal solution and the true optimal control. Since the learning–control process cannot achieve the exact optimal strategy for minimizing $\dot{d}_{z_n}(\tilde{x}(\tau_n))$ (whose minimal value is strictly less than the attraction force $-r(b - c\|\tilde{x}(\tau_n)\| - |a|)$), the controller instead seeks a near–optimal solution. Specifically, $u_{n+1,0} = (1-\epsilon)\,\mathrm{argmin}_\lambda \langle 2(\tilde{x}(\tau_{n+1}) - z_{n+1}), \sum_{j=0}^m \lambda_j(\tilde{x}_{n,j+1} - \tilde{x}_{n,j})\rangle$, where λ is such that $\sum_{j=0}^m \lambda_j = 1$ and $\lambda_j \geq 0$ for all j, and $\tilde{x}_{n,j} := \tilde{x}(\tau_n + j\delta t)$ is the shorthand notation of state $\tilde{x}$.

Note that all of the perturbation terms introduced above can be tuned to arbitrarily small values, thereby providing a theoretical guarantee for the automation of the control synthesis of each $u_{n,0}$, and hence for the entire control signal in the reachability problem. We summarize the algorithm below and demonstrate its application in the next section, and kindly refer the reader to [27] for further technical details.

Algorithm 1 Control Synthesis

Require: d, m, $\tilde{x}_{0,0} := \tilde{x}_0$, T, $\tilde{x}_f \in \partial\mathcal{R}_{\tilde{x}}(T, \tilde{x}_0)$, $\theta_0 = 0$, and $u_{0,0} = (1-\epsilon)\frac{\tilde{G}^\dagger(\tilde{x}_0)(\tilde{x}_f - \tilde{x}_0)}{\|\tilde{G}^\dagger(\tilde{x}_0)\||\tilde{x}_f - \tilde{x}_0|}$.
Require: δt, ϵ, k based on the condition (9), where $r = \sup_u |\tilde{x}_p(k(m+1)\delta t)|$
$n = 0$.
repeat
  $\tau_n = n(m+1)\delta t$.
  for j **from** 0 **to** m **do**
    $\mathbf{u}(t) \equiv u_{n,j}$ for all $t \in [\tau_n + j\delta t, \tau_n + (j+1)\delta t)$ based on Eq. (8);
    $\tilde{x}_{n,j+1} = \tilde{x}(\tau_n + (j+1)\delta t)$.
  end for
  $\mathfrak{x}_{n+1} = \tilde{x}_{n,m+1}$.
  Determine z_{n+1} by solving $\|z_{n+1} - \mathfrak{x}_{n+1}\|^2 = r^2$ with $z_{n+1} = \theta_{n+1}\tilde{x}_f$ and $\theta_{n+1} > \theta_n$.
  Let $u_{n+1,0} = (1-\epsilon)\,\mathrm{argmin}_\lambda \langle 2(\mathfrak{x}_{n+1} - z_{n+1}), \sum_{j=0}^m \lambda_j(\tilde{x}_{n,j+1} - \tilde{x}_{n,j})\rangle$, where $\sum_{j=0}^m \lambda_j = 1$ and $\lambda_j \geq 0$ for all j.
  $n := n + 1$.
until $\|z_n - \tilde{x}_f\| < r$.

V. Simulation of Reachability Predictive Control Framework

We now demonstrate RPC and its capability to solve the trajectory tracking and reach-avoid problem without knowledge of the system dynamics via simulation.

We consider a scenario in which an autonomous vehicle carrying a delicate payload must navigate an uncertain environment. To accomplish the mission, the vehicle must transport the payload between 2.2 and 2.8 meters per second to reach the target position without damaging the payload through high-frequency vibrations [34]. The vehicle will travel along various unknown surfaces, including dirt, gravel, and grass. We demonstrate that Algorithm 1 can be leveraged to solve Problems 1 and 2 under Conditions 1 and 2.

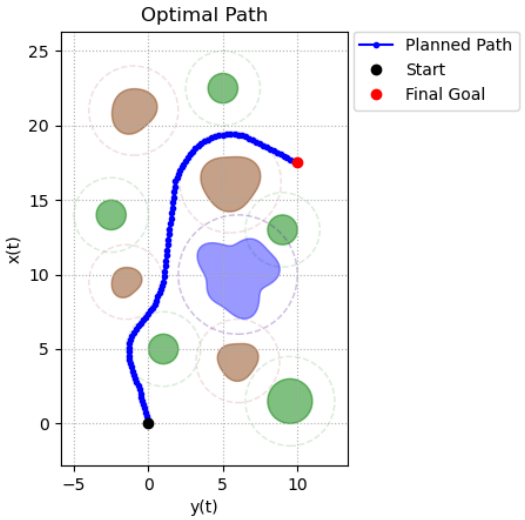

Fig. 2: An optimal path determined by solving (10). Path obstructing obstacles are shown in green, brown, and blue with the tolerance radius $\mathbb{B}^2(c_k, r_k + \Delta)$ for all k obstacles.

A. Control Design Requirements

We ultimately aim to navigate to a goal state while avoiding an unsafe set. Consider the states defined in (3) and notation in Problem 1; we define the unsafe set as $\mathcal{B} = \{(x, y, v) : v \notin [2.2, 2.8] \text{ and } (x, y) \in \mathcal{S}_{\text{blue}} \cup \mathcal{S}_{\text{brown}} \cup \mathcal{S}_{\text{green}}\}$ with the corresponding colored regions illustrated in Fig. 2. We formally present the control specification.

Objective 1: Synthesize a trajectory $\phi_u(\cdot, x_0)$ such that $\overline{x}(t) \to \overline{x}_f$ while $x(t) \notin \mathcal{B}$ for all t, where $\overline{x}$ represents the physical location of the vehicle.

We begin by determining an optimal path using sequential least squares programming (SLSQP) [35]. We solve the following optimization problem:

$$\begin{aligned} \min_{W} J(W) &= \sum_{i=1}^{N-1} \|(x_{i+1}, y_{i+1}) - (x_i, y_i)\|^2 \\ \text{s.t.} \quad & \|p_i^{\text{front}} - c_k\| \geq r_k + \Delta \quad \forall i, k \end{aligned} \tag{10}$$

where $W = \big[(x_1, y_1), \ldots, (x_N, y_N)\big] \in \mathbb{R}^{N\times 2}$, $p_i^{\text{front}} = (x_i, y_i) + \frac{L((x_i,y_i)-(x_{i-1},y_{i-1}))}{2\|(x_i,y_i)-(x_{i-1},y_{i-1})\|}$, c_k and r_k are the center and radius of the k-th obstacle, and Δ is the tolerance around each obstacle to avoid collisions. The optimal path we will follow and its surrounding environment are shown in Fig. 2. We now discuss how we can synthesize control action designed to follow the desired path for the vehicle presented in Section II-C.

B. Controller Synthesis for Vehicle Dynamics

As discussed in Section II-C, the vehicle dynamics in (3) are not control-affine. However, following the philosophy of GRS underestimation via proxy systems and employing myopic-type control for learning and control as in Section III and IV, our algorithm operates iteratively over short time horizons. For small angles, $\sin\theta \approx \theta$, $\cos\theta \approx 1$.

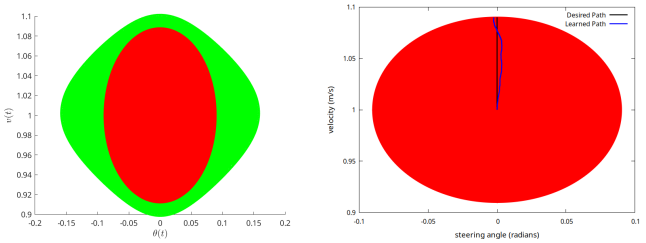

Fig. 3: **Left:** the true reachable set (green) of actuated states θ, v of system (3) calculated knowing the dynamics, and an underapproximation (red) calculated using solely the knowledge from Assumption 1 for system (11). **Right:** we apply Algorithm 1 to learn a path (blue) designed to follow a reachable path (black). The learned path (blue) is calculated without knowing the dynamics, but instead using $\tilde{x}_0 = \begin{bmatrix} 0 & 1 \end{bmatrix}^T$, $L_f^{\mathcal{N}} = 0$, $L_G^{\mathcal{N}} = 3$, $\tilde{f}(\tilde{x}_0) = 0$, $\tilde{G}(\tilde{x}_0) = I$.

Additionally, $|u_2| \leq |\tan(u_2)|$ for $|u_2| \leq 1$, and the reachable set of

$$\dot{x} = v, \ \dot{y} = v\theta, \ \dot{\theta} = \frac{v}{l_r} u_2, \ \dot{v} = u_1 \tag{11}$$

is contained in the reachable set of (3), as shown in Fig. 3. Note that (11) satisfies the assumed structure. This construction is valid for RPC purposes, as it provides an underapproximation of the GRS of (1), ensuring navigation toward a provably reachable state. Operating within this underapproximated GRS prevents the system from entering unproved regions of the state space. We can thus apply the procedures from [26], [30] to extract knowledge from Assumption 1 for the local system (11). The above setting ensures that our guarantees remain valid and enables controller synthesis, as illustrated in Fig. 3. We now detail the proposed control strategy to accomplish this task.

C. RPC Algorithm and Control Strategy

We begin by presenting the RPC algorithm, a high-level path-following algorithm that requires iteratively applying the low-level reachability control Algorithm 1.We use the notation $\hat{n} \in \mathcal{N}$, where $\hat{n}$ indexes the number of times Algorithm 1 has been applied, and $\mathcal{N}$ is the integer set of all iteration indices, starting at index 0.

To correctly apply Algorithm 2, we must reinitialize the initial conditions at the end of each iteration to the system's current state, update the estimates of $\mathcal{R}_{\tilde{x}}(T, \tilde{x}_0)$ and $\mathcal{R}_{\overline{x}}(T, x_0)$, and to choose $\tilde{x}_f^{\hat{n}+1}$, such that Algorithm 1 can be used to synthesize a controller that reaches $\tilde{x}_f^{\hat{n}+1}$. The selection of $\tilde{x}_f^{\hat{n}+1}$ is guided by the updated information of $\tilde{x}_f^{\hat{n}+1}$ and its relative position w.r.t. the desired path, which we look over multiple time horizons for high-level planning. Physically, this corresponds to tuning the velocity and steering angles so that the controlled state trajectory converges to the desired path. The following remark provides conditions under which the calculated reachable sets are simply translated, that is, situations where the calculated sets are affine transforms centered at different initial conditions.

Algorithm 2 Reachable Predictive Control

Require: d, m, $\tilde{x}_{0,0} := \tilde{x}_0$, T, $\tilde{x}_f^0 \in \partial\mathcal{R}_{\tilde{x}}(T, \tilde{x}_0)$, $\overline{x}_f$, $\theta_0 = 0$, and $u_{0,0} = (1-\epsilon)\frac{\tilde{G}^\dagger(\tilde{x}_0)(\tilde{x}_f - \tilde{x}_0)}{\|\tilde{G}^\dagger(\tilde{x}_0)\|\|\tilde{x}_f - x_0|}$, $\overline{r} = \sup_{\overline{x}\in\mathcal{R}_{\overline{x}}(T,x_0)} \|\overline{x}_0 - \overline{x}\|$, $\hat{n} = 0$.

Require: δt, ϵ, k based on the condition (9), where $r = \sup_u |\tilde{x}_p(k(m+1)\delta t)|$

1: **while** $\|\overline{x} - \overline{x}_f\| > \overline{r}$ **do**
2: Run Algorithm 1 with $\tilde{x}_f^{\hat{n}}$
3: Update x_0 to current state
4: Calculate $\mathcal{R}_{\tilde{x}}(T, \tilde{x}_0)$ using [23, Theorem 3]
5: Calculate $\mathcal{R}_{\overline{x}}(T, x_0)$ in using Theorem 3
6: Choose $\tilde{x}_f^{\hat{n}+1} \in \mathcal{R}_{\tilde{x}}(T, \tilde{x}_0)$
7: Reset $r = \sup_u |\tilde{x}_p(k(m+1)\delta t)|$ for Algorithm 1
8: Reinitialize $\theta_0 = 0$ for Algorithm 1
9: $\overline{r} = \sup_{\overline{x}\in\mathcal{R}_{\overline{x}}(T,x_0)} \|\overline{x}_0 - \overline{x}\|$
10: $\hat{n} := \hat{n} + 1$
11: **end while**

Remark 1: Consider the notation introduced in Theorem 4. For $\overline{a} \in \mathbb{R}^m$ and $\mathcal{A} \subseteq \mathbb{R}^m$, define $\mathcal{A}+\overline{a} = \{a+\overline{a} \mid a \in \mathcal{A}\}$. Let $L_f^{\hat{n}}$ and $L_G^{\hat{n}}$ be the Lipschitz constants around a neighborhood of $x(0)$ and similarly $L_f^{\hat{n}+1}$ and $L_G^{\hat{n}+1}$ be the same around a different neighborhood centered at $y(0)$. For x, $y \in \mathbb{R}^n$ such that $x(0) \neq y(0)$, if $\|\tilde{G}^\dagger(\tilde{x}_0)\|^{-1} = \|\tilde{G}^\dagger(\tilde{y}_0)\|^{-1}$, $\|F^\dagger(x_0)\|^{-1} = \|F^\dagger(y_0)\|^{-1}$, $L_f^{\hat{n}} = L_f^{\hat{n}+1}$ and $L_G^{\hat{n}} = L_G^{\hat{n}+1}$, then $\mathcal{R}_{\tilde{x}}(T, \tilde{x}_0) + (\tilde{y}_0 - \tilde{x}_0) = \mathcal{R}_{\tilde{y}}(T, \tilde{y}_0)$. Additionally, if $\|\tilde{x}_0\| = \|\tilde{y}_0\|$, then $\overline{\mathcal{R}}_{\overline{x}}(T, x_0) + (y_0 - x_0) = \overline{\mathcal{R}}_{\overline{y}}(T, y_0)$. □

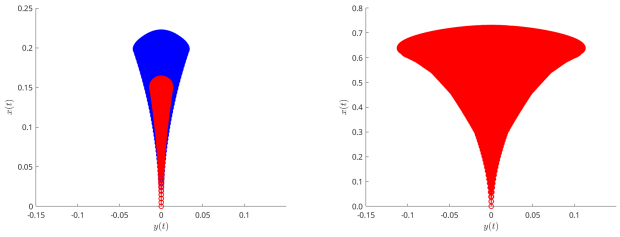

Fig. 4: **Left**: reachable set (blue) of states $\overline{x} = \begin{bmatrix} x & y \end{bmatrix}^T$ of system (11), calculated knowing the dynamics, and its *guaranteed* underapproximation (red). **Right**: set $\mathcal{R}_{\overline{x}}^{\tilde{n}}(T, x_0)$. The sets in red are calculated without knowledge of the dynamics, using solely the knowledge of $x(0) = \begin{bmatrix} 0 & 0 & 0 & 2.0 \end{bmatrix}^T$ for $T = 0.1$ seconds, $\tilde{f}(\tilde{x}_0) = 0$, $\tilde{G}(\tilde{x}_0) = \text{diag}(1, 2)$, $L_f^{\hat{n}} = 0$, and $L_G^{\hat{n}} = 3$ for $\hat{n} \in \{1, \ldots, 5\}$.

Intuitively, for short time horizons, the values in Remark 1 may not change significantly, making it feasible to approximate a larger set of states for path planning. We emphasize that controller synthesis is only computed using the guaranteed set of states $\mathcal{R}_{\tilde{x}}(T, \tilde{x}_0)$; thus, the paths we follow remaint *guaranteed* reachable paths. However, for high-level planning, we can closely approximate a set of states that are reachable to help inform which guaranteed path to follow. Let $\mathcal{R}_{\overline{x}}^{\tilde{n}}(T, x_0)$ be the reachable set $\mathcal{R}_{\overline{x}}(T, x_0)$ calculated under the assumption that Remark 1 holds for $\tilde{n}$

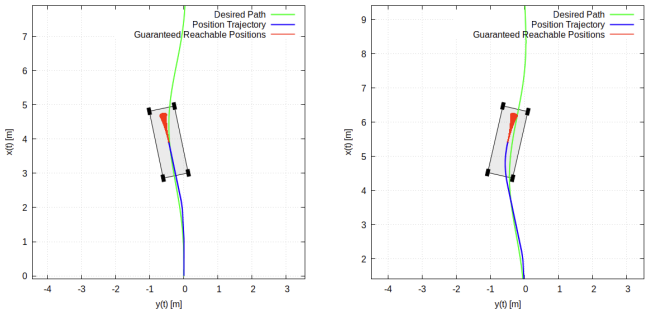

Fig. 5: Autonomous path following of the desired path (green) by informed decision making using calculated guaranteed reachable positions (red) over a total of 0.5 seconds.

time horizons of length T at $x(0) = x_0$. Fig. 4 illustrates the reachable sets calculated at $x_0 = \begin{bmatrix} 0 & 0 & 0 & 2.0 \end{bmatrix}^T$ for $T = 0.1$ and $\tilde{n} = 5$. We use $\mathcal{R}_{\overline{x}}^5(0.1, x_0)$ to guide the selection of $\tilde{x}_f^{\hat{n}+1}$ in Algorithm 2 at each iteration until Objective 1 is completed. This strategy is formally presented in the table below.

Control Action	Condition
Increase v	$\mathcal{R}_{\overline{x}}^5(0.1, x_0) \cap$ desired path $\neq \varnothing$, $v \leq 2.3(m/s)$
Zero Input	$\mathcal{R}_{\overline{x}}^5(0.1, x_0) \cap$ desired path $\neq \varnothing$, $v \geq 2.7(m/s)$
Increase θ	red $\cap$ green $= \varnothing$, car to left of desired path
Decrease θ	red $\cap$ green $= \varnothing$, car to right of desired path

TABLE I: Strategy for choosing $\tilde{x}_f^{\hat{n}+1}$ in Algorithm 2.

Fig. 5 illustrates the strategy shown in Table I. Intuitively, $\tilde{n}$ can be viewed as a tuning knob that adjusts how closely the vehicle should follow the position path; the smaller $\tilde{n}$, the more often the control effort would focus θ to remain on the desired position path. We next detail how we can tune the parameters of Algorithms 1 and 2 to provide a solution to design objective introduced in Objective 1.

D. Real-Time Performance

For RPC to operate in real time, it is necessary to compute $\mathcal{R}_{\tilde{x}}(T, \tilde{x}_0)$, $\mathcal{R}_{\overline{x}}(T, x_0)$, $\mathcal{R}_{\overline{x}}^{\tilde{n}}(T, x_0)$ rapidly. This is feasible due to the simple structure of the proxy systems (5) and (7). Existing results [27, Section 3] show that, to characterize the boundary of the reachable sets of these systems, it suffices to consider inputs on the boundary of $\mathcal{U}$. Consequently, we can efficiently compute the reachable sets by employing Monte Carlo simulations that solve ODEs with randomly sampled time-varying inputs restricted to the boundary of $\mathcal{U}$. This significantly reduces computation time of $\mathcal{R}_{\tilde{x}}(T, \tilde{x}_0)$, $\mathcal{R}_{\overline{x}}(T, x_0)$, $\mathcal{R}_{\overline{x}}^{\tilde{n}}(T, x_0)$. Additionally, as shown in Algorithm 1, the surrogate suboptimal control can be obtained via linear programming, achieving computation times comparable to those for the GRS computation. For the purposes of this application, we set $T = 0.1$ and $\tilde{n} = 5$ and for each iteration $\hat{n}$, we had a total computation time of ≈ 0.081 seconds to calculate all results of interest. For additional background and analysis about how Algorithm 1 can be calculated in real time, we refer the reader to [27].

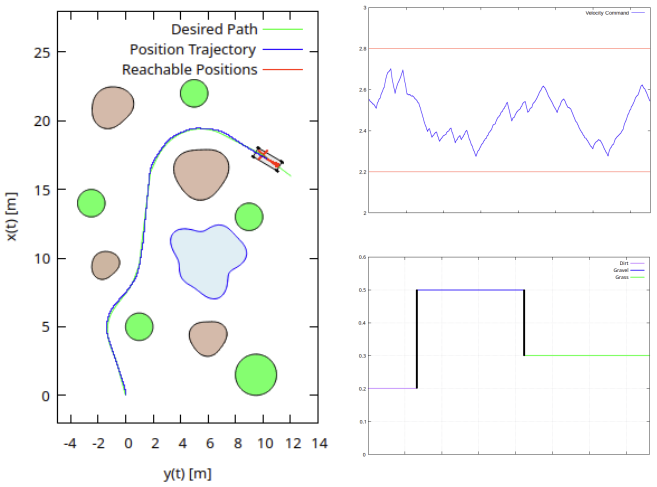

Fig. 6: **Left:** vehicle position trajectory (blue) closely following an optimal desired path (green) guided by its reachable positions (red) without any obstacle collisions. **Top right:** velocity over time such that $v(t) \notin \mathcal{B}$ for all t. **Bottom right:** varying unknown r_c disturbance magnitudes as the vehicle traversed the unknown terrain.

Throughout the simulation, we used $L_f^{\hat{n}} = 0$ and $L_G^{\hat{n}} = 3$ for all $\hat{n}$, with $\tilde{f}(\tilde{x}_0)$ and $\tilde{G}(\tilde{x}_0)$ updated at each $\hat{n}$ depending on the reinitialized value of x_0 at the start of each iteration of Algorithm 1. We simulated results using the dynamics from (3) with $r_c = 0.2, 0.5, 0.3$, which pertains to the rolling resistance coefficients of dirt, gravel, and grass respectively. The reachable positions for the high-level planning are calculated using Theorem 3. Adopting the notation of Theorem 2, we thus require $b(t) \leq \|a\|$ where $\|a\| = \|\tilde{x}_0\|$ and $b(t) = \|y - \tilde{x}_0\|$ for any $y \in \partial\mathbb{B}^m(\tilde{x}_0; \|y - \tilde{x}_0\|) \subseteq \mathcal{R}_{\tilde{x}}(T, \tilde{x}_0)$. These conditions are always met because $2.2 \leq v_0 \leq 2.8$ and because Algorithm 1 uses short time horizons, so $b(t)$ is always sufficiently small. In practice, if $\|\tilde{x}_0\|$ were small, the system would be near rest and there may be no need for the implementation of safety-critical path following.

Fig. 6 illustrates the results for a randomly generated set of obstacles with varying terrain. The analysis from Section IV and [27] showed that, if the inequality (9) was satisfied, then there exist parameters δt, ϵ, and k such that there is a minimizing solution to $u_{n+1,0} = (1-\epsilon)\,\mathrm{argmin}_\lambda \langle 2(\tilde{x}(\tau_{n+1}) - z_{n+1}), \sum_{j=0}^{m} \lambda_j(\tilde{x}_{n,j+1} - \tilde{x}_{n,j})\rangle$, where $u_{n+1,0}$ drives the system towards the desired state $\tilde{x}_f^{\hat{n}}$ at every n. By setting $\delta t = 0.015$, $\epsilon = 0.1$, and $k = 5$, Algorithm 1 was able to follow a desired trajectory to $\tilde{x}_f^{\hat{n}}$ at each iteration. Implementing the control strategy from Table I, we see that RPC is capable of closely following the desired optimal path without ever intersecting an unsafe set by iteratively applying Algorithm 1. Eventually, RPC completed Objective 1 without any position obstacle collisions while the vehicle velocity remained within the desired bound.

VI. Conclusion

In this paper, we have developed a Reachable Predictive Control (RPC) framework, illustrated with an example of an unknown vehicle model under reasonably mild assumptions—requiring only knowledge of the bounds on the Lipschitz constants. The proposed framework is capable of driving the system to follow a piecewise-linear trajectory on-the-run without prior knowledge of the system dynamics. In particular, we extend previous work from the control of fully actuated systems to the more challenging setting of underactuated systems. This includes a specific extension of GRS underapproximation techniques and a demonstration of how the control synthesis algorithm can be embedded into the RPC framework to enable an automated path-following procedure. The framework can generate control inputs automatically, using tunable parameters that allow the perturbations arising from learning and control to be made arbitrarily small. Moreover, it provides a principled approach for handling abrupt changes in the system dynamics, enabling online learning and control with provable guarantees.

This work fills an important gap for scenarios where trajectory data is scarce, and traditional system identification methods or reinforcement learning cannot be applied effectively. The RPC framework complements and integrates well with MPC/CBF-based methods. Finally, for future engineering applications, it is of mathematical interest to sharpen the estimation bounds, thereby simplifying parameter tuning and improving practical usability.

References

[1] A. Lin, S. Peng, and S. Bansal, "One filter to deploy them all: Robust safety for quadrupedal navigation in unknown environments," *arXiv preprint arXiv:2412.09989*, 2024.

[2] N. Ramdani and N. S. Nedialkov, "Computing reachable sets for uncertain nonlinear hybrid systems using interval constraint propagation techniques," *IFAC Proceedings Volumes*, vol. 42, no. 17, pp. 156–161, 2009.

[3] M. Rungger and M. Zamani, "Accurate reachability analysis of uncertain nonlinear systems," in *Proceedings of the 21st international conference on hybrid systems: Computation and control (part of CPS week)*, 2018, pp. 61–70.

[4] L. Knoedler, O. So, J. Yin, M. Black, Z. Serlin, P. Tsiotras, J. Alonso-Mora, and C. Fan, "Safety on the fly: Constructing robust safety filters via policy control barrier functions at runtime," *IEEE Robotics and Automation Letters*, 2025.

[5] J. B. Rawlings, D. Q. Mayne, M. Diehl *et al.*, *Model predictive control: theory, computation, and design*. Nob Hill Publishing Madison, WI, 2020, vol. 2.

[6] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[7] D. Q. Mayne, M. M. Seron, and S. V. Raković, "Robust model predictive control of constrained linear systems with bounded disturbances," *Automatica*, vol. 41, no. 2, pp. 219–224, 2005.

[8] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics, and constraints," *Automatica*, vol. 35, no. 3, pp. 407–427, 1999.

[9] Y. S. Quan, J. S. Kim, and C. C. Chung, "Robust model predictive control with control barrier function for nonholonomic robots with obstacle avoidance," in *2021 21st International Conference on Control, Automation and Systems (ICCAS)*. IEEE, 2021, pp. 1377–1382.

[10] U. Rosolia, X. Zhang, and F. Borrelli, "Data-driven predictive control for autonomous systems," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, no. 1, pp. 259–286, 2018.

[11] M. Wang, X. Lou, and B. Cui, "Learning-based robust model predictive control with data-driven koopman operators," *International Journal of Machine Learning and Cybernetics*, vol. 14, no. 9, pp. 3295–3321, 2023.

[12] A. Dittmer, B. Sharan, and H. Werner, "Data-driven adaptive model predictive control for wind farms: A koopman-based online learning approach," in *2022 IEEE 61st Conference on Decision and Control (CDC)*. IEEE, 2022, pp. 1999–2004.

[13] J.-R. Chiu, J.-P. Sleiman, M. Mittal, F. Farshidian, and M. Hutter, "A collision-free mpc for whole-body dynamic locomotion and manipulation," in *2022 international conference on robotics and automation (ICRA)*. IEEE, 2022, pp. 4686–4693.

[14] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 3387–3395.

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

[16] B. Thananjeyan, A. Balakrishna, S. Nair, M. Luo, K. Srinivasan, M. Hwang, J. E. Gonzalez, J. Ibarz, C. Finn, and K. Goldberg, "Recovery rl: Safe reinforcement learning with learned recovery zones," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4915–4922, 2021.

[17] T. He, C. Zhang, W. Xiao, G. He, C. Liu, and G. Shi, "Agile but safe: Learning collision-free high-speed legged locomotion," *arXiv preprint arXiv:2401.17583*, 2024.

[18] H. Bharadhwaj, A. Kumar, N. Rhinehart, S. Levine, F. Shkurti, and A. Garg, "Conservative safety critics for exploration," *arXiv preprint arXiv:2010.14497*, 2020.

[19] K.-C. Hsu, A. Z. Ren, D. P. Nguyen, A. Majumdar, and J. F. Fisac, "Sim-to-lab-to-real: Safe reinforcement learning with shielding and generalization guarantees," *Artificial Intelligence*, vol. 314, p. 103811, 2023.

[20] F. Jenelten, J. He, F. Farshidian, and M. Hutter, "Dtc: Deep tracking control," *Science Robotics*, vol. 9, no. 86, p. eadh5401, 2024.

[21] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science robotics*, vol. 7, no. 62, p. eabk2822, 2022.

[22] A. Agarwal, A. Kumar, J. Malik, and D. Pathak, "Legged locomotion in challenging terrains using egocentric vision," in *Conference on robot learning*. PMLR, 2023, pp. 403–415.

[23] T. Shafa and M. Ornik, "Reachability of nonlinear systems with unknown dynamics," *IEEE Transactions on Automatic Control*, vol. 68, no. 4, pp. 2407–2414, 2022.

[24] ——, "Maximal ellipsoid method for guaranteed reachability of unknown fully actuated systems," in *2022 IEEE 61st Conference on Decision and Control (CDC)*. IEEE, 2022, pp. 5002–5007.

[25] ——, "Reachability of nonlinear systems with unknown dynamics," *IEEE Transactions on Automatic Control*, vol. 68, no. 4, pp. 2407–2414, 2023.

[26] M. Ornik, S. Carr, A. Israel, and U. Topcu, "Control-oriented learning on the fly," *IEEE Transactions on Automatic Control*, vol. 65, no. 11, pp. 4800–4807, 2019.

[27] Y. Meng, T. Shafa, J. Wei, and M. Ornik, "Online learning and control synthesis for reachable paths of unknown nonlinear systems," *IEEE Transactions on Automatic Control (conditionally accepted)*, 2025.

[28] Y. Liu and H. Yu, "A survey of underactuated mechanical systems," *IET Control Theory & Applications*, vol. 7, no. 7, pp. 921–935, 2013.

[29] M. W. Spong, S. Hutchinson, and M. Vidyasagar, "Robot modeling and control," *John Wiley &amp*, 2020.

[30] H. El-Kebir, A. Pirosmanishvili, and M. Ornik, "Online guaranteed reachable set approximation for systems with changed dynamics and control authority," *IEEE Transactions on Automatic Control*, vol. 69, no. 2, pp. 726–740, 2023.

[31] P. Polack, F. Altché, B. d'Andréa Novel, and A. de La Fortelle, "The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?" in *2017 IEEE intelligent vehicles symposium (IV)*. IEEE, 2017, pp. 812–818.

[32] J. Yuan, H. Chen, F. Sun, and Y. Huang, "Trajectory planning and tracking control for autonomous bicycle robot," *Nonlinear Dynamics*, vol. 78, no. 1, pp. 421–431, 2014.

[33] J. A. Matute, M. Marcano, S. Diaz, and J. Perez, "Experimental validation of a kinematic bicycle model predictive control with lateral acceleration consideration," *IFAC-PapersOnLine*, vol. 52, no. 8, pp. 289–294, 2019.

[34] H. El-Kebir, T. Shafa, A. Purushottam, M. Ornik, and A. Soylemezoglu, "High-frequency vibration reduction for unmanned ground vehicles on unstructured terrain," in *International Conference on Modelling and Simulation for Autonomous Systems*. Springer, 2021, pp. 74–92.

[35] Y. Ma, N. Zhang, and J. Li, "Improved sequential least squares programming–driven feasible path algorithm for process optimisation," in *Computer aided chemical engineering*. Elsevier, 2022, vol. 51, pp. 1279–1284.

APPENDIX I
ERROR ESTIMATION AND GROWTH RATE BOUNDS

We summarize the perturbation terms in (9) generated from system learning under control of the form (8).

Denote $M_0 := \max\{\sup_{\tilde{x}\in\mathbb{B}} |\tilde{f}(\tilde{x})|, \sup_{\tilde{x}\in\mathbb{B}} |\tilde{G}(\tilde{x})|\}$, $L_0 := \max\{L_f^{\mathcal{N}}, L_G^{\mathcal{N}}\}$, $C := \|\tilde{G}(\tilde{x}_0)\|\|\tilde{G}^\dagger(\tilde{x}_0)\|$, $C_0 := M_0(m+1)$, $C_1 := M_0(m+1)^2$, $C_2 := 2M_0L_0(m+1)^2$, and $C_3 := M_0L_0(m+1)^3$. Let $\tilde{x}_{n,j} = \tilde{x}(\tau_n + j\delta t)$ and $\mathfrak{x}_{n+1} := \tilde{x}_{n,m+1}$. Let $v_x(u) := \tilde{f}(\tilde{x}) + \tilde{G}(\tilde{x})u$. Then, it is clear that $\mathfrak{x}_{n+1} = \tilde{x}_{n+1,0}$. We have the following approximation precision [26, Lemma 4]:

(1) $|\tilde{x}(t_1) - \tilde{x}(t_2)| \leq C_0|t_2 - t_1|$, $\forall t_1, t_2 \in [\tau_n, \tau_{n+1}]$. In particular, $|\mathfrak{x}_{n+1} - \mathfrak{x}_n| \leq C_1 \cdot \delta t$;
(2) $|\frac{(\tilde{x}_{n,j+1} - \tilde{x}_{n,j})}{\delta t} - v_{\tilde{x}_{n,j+1}}(u_{n,j})| \leq (C_2/4) \cdot \delta t$, $\forall j \in \mathcal{I}$.
(3) $|v_{x_{n,j+1}}(u_{n,j}) - v_{\mathfrak{x}_{n+1}}(u_{n,j})| \leq C_3 \cdot \delta t$, $\forall j \in \mathcal{I}$.

The perturbation terms in (9) are summarized as follows: $\mathcal{E}_r(\delta t, \epsilon) := C_2\delta t + \epsilon M_0$, $\mathcal{E}_n(\delta t, \epsilon) := 2M_0C_1\delta t + C_1\mathcal{E}_r(\delta t, \epsilon)$, $\mathcal{E}_\mu(\delta t, \epsilon) = (3L_d(C_4/C_3)\mathcal{E}_\lambda(\delta t, \epsilon) + L_dC_0\delta t)/2$, where $\mathcal{E}_\lambda(\delta t, \epsilon) = 2((4m^{\frac{3}{2}} + \epsilon)/\epsilon) \cdot C_3 \cdot \delta t$.

We introduce growth rate bounds on $F(x(t))$ from (2b) using the knowledge available from Assumption 1.

Lemma 2: Let $\tilde{x}(t) \in \mathbb{B}^m(\tilde{x}_0; b(t))$ where $b(t) \in \mathbb{R}_+$ and $\|\tilde{x}(t)\| > 0$. Let $\alpha_{\min} = \|\tilde{x}_0\| - b(t)$ and $\alpha_{\max} = \|\tilde{x}_0\| + b(t)$. If $L_F(\|x - x_0\|) = \frac{L_f^{\mathcal{N}}\alpha_{\min}\alpha_{\max} + \|\overline{f}(x_0)\|(1+2\alpha_{\max})}{\alpha_{\min}^3} + \mathcal{O}(\|x - x_0\|)$ where $\mathcal{O}(\|x - x_0\|) = \frac{L_f^{\mathcal{N}}(\alpha_{\min}+2\alpha_{\max})}{\alpha_{\min}^3}\|x - x_0\|$, then $\|F(x) - F(x_0)\| \leq L_F(\|x - x_0\|)\|x - x_0\|$.

Proof: Consider $F(x) := \frac{\overline{f}(x)\tilde{x}^T}{\|\tilde{x}\|^2}$. Then, for any x, y, $\|F(x) - F(y)\| \leq \left\|\frac{\overline{f}(x)\tilde{x}^T - \overline{f}(y)\tilde{y}^T}{\|\tilde{x}\|^2}\right\| + \|\overline{f}(y)\tilde{y}^T\|\left|\frac{1}{\|\tilde{x}\|^2} - \frac{1}{\|\tilde{y}\|^2}\right| \leq \frac{\|\overline{f}(x) - \overline{f}(y)\|\|\tilde{x}\| + \|\overline{f}(y)\|\|\tilde{x} - \tilde{y}\|}{\|\tilde{x}\|^2} + \|\overline{f}(y)\|\frac{\left|\|\tilde{x}\|^2 - \|\tilde{y}\|^2\right|}{\|\tilde{x}\|^2\|\tilde{y}\|}$. We have $\|\overline{f}(x) - \overline{f}(y)\| \leq L_f^{\mathcal{N}}\|x - y\|$, $\|\tilde{x} - \tilde{y}\| \leq \|x - y\|$, $\|\overline{f}(y)\| \leq \|\underline{f}(x)\| + L_f\|x - y\|$, and $\|\tilde{x}\| \leq \|\tilde{x}_0\| + c(t)$, so $\|\overline{f}(x) - \overline{f}(y)\|\|\tilde{x}\| + \|\overline{f}(y)\|\|\tilde{x} - \tilde{y}\| \leq (L_f^{\mathcal{N}}(\|\tilde{x}_0\| + b(t)) + \|\overline{f}(x)\|)\|x - y\| + L_f^{\mathcal{N}}\|x - y\|^2$. Additionally, using the property that $\left|\|\tilde{x}\|^2 - \|\tilde{y}\|^2\right| \leq (\|\tilde{x}\| + \|\tilde{y}\|)\|\tilde{x} - \tilde{y}\|$, we get $\|\overline{f}(y)\|\|\tilde{y}\|\left|\|\tilde{x}\|^2 - \|\tilde{y}\|^2\right| \leq 2\|\overline{f}(x)\|(\|\tilde{x}_0\| + b(t))\|x - y\| + 2L_f^{\mathcal{N}}(\|\tilde{x}_0\| + b(t))\|x - y\|^2$. Finally, substituting the right-hand side of the inequality $\|\tilde{x}\|, \|\tilde{y}\| \geq \|\tilde{x}_0\| - b(t)$ provides our result. ∎

The RPC algorithm calculates reachable sets and performs controller synthesis for short time horizons, so in practice, $\|x - x_0\|$ is not large. Additionally, as $t \to 0$, $\alpha_{\max} \to \alpha_{\min}$, so in practice, $L_f^{\mathcal{N}}$ often serves as an acceptable Lipschitz bound for $F(x)$. For cases where $L_f^{\mathcal{N}}$ is not a viable Lipschitz bound for $F(x(t))$, we could use the bound from Lemma 2.