Robotics: Science and Systems 2025
Los Angeles, CA, USA, June 21–25, 2025

Safe Beyond the Horizon: Efficient Sampling-based MPC with Neural Control Barrier Functions

Ji Yin[1]*, Oswin So[2]*, Eric Yang Yu[2], Chuchu Fan[2], and Panagiotis Tsiotras[1]

***Abstract*—A common problem when using model predictive control (MPC) in practice is the satisfaction of safety specifications beyond the prediction horizon. While theoretical works have shown that safety can be guaranteed by enforcing a suitable terminal set constraint or a sufficiently long prediction horizon, these techniques are difficult to apply and thus are rarely used by practitioners, especially in the case of general nonlinear dynamics. To solve this problem, we impose a tradeoff between exact recursive feasibility, computational tractability, and applicability to "black-box" dynamics by learning an approximate discrete-time control barrier function and incorporating it into a variational inference MPC (VIMPC), a sampling-based MPC paradigm. To handle the resulting state constraints, we further propose a new sampling strategy that greatly reduces the variance of the estimated optimal control, improving the sample efficiency, and enabling real-time planning on a CPU. The resulting Neural Shield-VIMPC (NS-VIMPC) controller yields substantial safety improvements compared to existing sampling-based MPC controllers, even under badly designed cost functions. We validate our approach in both simulation and real-world hardware experiments. Project website: https://mit-realm.github.io/ns-vimpc/.**

I. Introduction

Model Predictive Control (MPC) is a versatile control approach widely used in robotics applications such as autonomous driving [76], bio-inspired locomotion [12, 35], or manipulation of deformable objects [59], to name just a few. These methods address safety by incorporating state and control constraints into the finite-horizon optimization problem, ensuring that the system remains safe over the prediction horizon [12, 49, 59, 74]. However, safety of the system beyond the prediction horizon is often overlooked by practitioners, potentially leading to the violations of safety constraints at future timesteps.

This is a well-known problem in the field of MPC. The question of whether a sequence of safe control actions can always be found under an MPC controller has been studied extensively in the literature under the name of recursive feasibility [14, 32, 42, 50, 52]. A simple method of achieving recursive feasibility is by enforcing a control-invariant terminal set constraint at the end of the prediction horizon [14, 42]. However, it is difficult to find such a control-invariant set for general nonlinear systems. Methods such as bounding the system dynamics [13, 14, 26] often result in over-conservative sets, while methods that numerically solve the Hamilton-Jacobi (HJ) PDE [51] scale exponentially with the number of state variables, and hence this approach is computationally infeasible for systems with more than, say, 5 state variables [53].

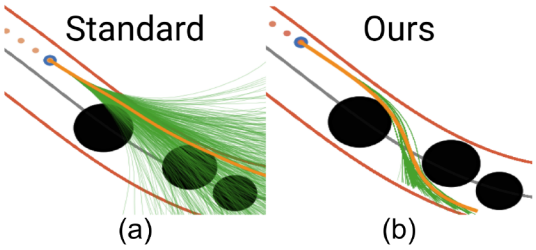


Fig. 1: **Standard sampling vs. Resampling-Based Rollout (RBR).** In this AutoRally example, the blue dot samples future trajectories and computes an optimal control that avoids the black obstacles. RBR rewires all sampled trajectories to be safe, resulting in a more accurate sampling distribution. In contrast, the standard approach samples from a Gaussian distribution and wastes computation on unsafe trajectories.

Even without considering recursive feasibility, nonlinear *constrained* optimization is a challenging problem. Traditionally, constraints are handled using techniques such as interior-point, sequential quadratic programming and augmented Lagrangian type methods [56], which mostly rely on accurate linear or quadratic approximations of the cost and constraints. However, these gradient-based methods can get stuck in local minima or fail to converge when the problem is highly nonlinear. Moreover, first and second-order gradient information is needed to solve these optimization problems, which is often not available for "black-box" systems. Consequently, MPC controllers that rely on these underlying nonlinear constrained optimizers [30, 38, 56] inherit the same limitations.

One way to address these previous limitations is to use sampling-based optimization methods, which have become popular within both the robotics and model-based reinforcement learning communities. In particular, variational inference MPC (VIMPC) [8, 45, 57, 60, 61, 74] has emerged as a popular family of methods that pose the control problem as an inference problem instead of an optimization problem through the control as inference framework [6, 65, 72], and perform variational inference to approximate the resulting intractable optimal control distribution. Popular MPC controllers within this family include Model Predictive Path Integral (MPPI) [76] and the Cross-Entropy Method (CEM) [66, 74], which have been applied to robotics tasks such as autonomous driving [76], bipedal locomotion [12], manipulation of deformable objects

[1]Ji Yin and Panagiotis Tsiotras are with D. Guggenheim School of Aerospace Engineering, Georgia Institute of Technology, Atlanta, GA. Email:{jyin81, tsiotras}@gatech.edu

[2]Oswin So, Eric Yang Yu and Chuchu Fan are with the Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA. Email:{oswinso, eyyu, chuchu}@mit.edu

* Equal contribution

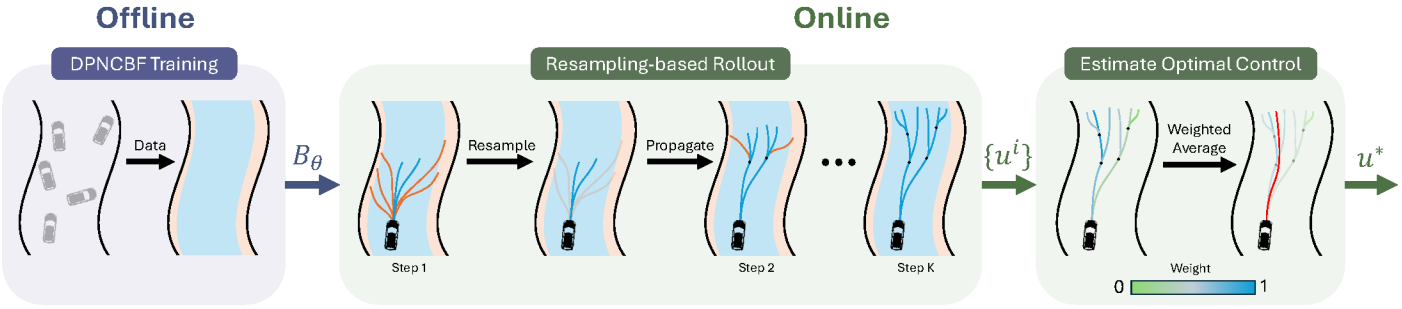


Fig. 2: **Overview of Neural Shield VIMPC (NS-VIMPC).** Offline, using DPNCBFs, we collect a dataset and train a NN approximation B_θ of a DCBF. B_θ is used to impose the DCBF descent condition (16) as a state constraint. Online, we modify the VIMPC architecture to use resampling-based rollouts to improve the sampling distribution of the Monte Carlo estimator in the presence of the DCBF state constraints. In addition, we integrate the DCBF safety condition (14b) into the optimization objective function as in (18).

[59], and in-hand manipulation [54], among many others. Using this framework, constraints can be easily handled by appropriately manipulating the posterior control distribution without facing the same optimization challenges as traditional gradient-based methods. However, the problem of safety beyond the prediction horizon still remains a challenge and has not been addressed by existing works that employ VIMPC.

In this work, we present a novel sampling-based MPC approach that provides safety beyond the prediction horizon by using control barrier functions to enforce the control-invariant set constraint for VIMPC controllers. To tackle the challenge of finding control-invariant sets for general nonlinear systems, we extend our previous work on learning neural network approximations of control barrier functions using policy neural control barrier functions (PNCBF) [69] to the discrete-time case. Inspired by particle filtering and sequential Monte Carlo methods, we further propose a novel sampling strategy for handling state constraints in VIMPC that significantly improves the sampling efficiency and enables real-time planning on a CPU. Results from both simulation and hardware experiments suggest that the resulting Neural Shield-VIMPC (NS-VIMPC) controller outperforms existing MPC baselines in terms of safety, even under adversarially tuned cost functions.

Contributions. We summarize our contributions below.

- We extend policy neural CBF (PNCBF) to the discrete-time case and propose a novel approach to train a discrete-time PNCBF (DPNCBF) using policy evaluation.
- We propose Resampling-Based Rollout (RBR), a novel sampling strategy for handling state constraints in VIMPC inspired by particle filtering, which significantly improves the sampling efficiency by lowering the variance of the estimated optimal control.
- Simulation results on two benchmark tasks show the efficacy of NS-VIMPC compared to existing sampling-based MPC controllers in terms of safety and sample efficiency.
- Hardware experiments on AutoRally [29], a $1/5$ scale autonomous driving platform, demonstrate the robustness of NS-VIMPC to unmodeled dynamical disturbances under adversarially tuned cost functions.

II. Related Work

Sampling-based MPC. Sampling-based MPC has become a popular alternative to traditional MPC methods that use gradient-based solvers, in part due to the advent of parallel computing and the recent advances in GPU hardware. As a gradient-free method, sampling-based MPC can be applied to any problem without requiring specific problem structures. MPPI [76] is a popular sampling-based MPC approach that formulates a variational inference problem, then solves it in the case of the Gaussian distribution, having strong connections to stochastic optimal control [77] and maximum entropy control [68]. Separately, the Cross-Entropy Method (CEM) [66] has become popular in the reinforcement learning community [74], in part due to its simplicity. Both approaches were recently shown to be part of the VIMPC family of MPC algorithms [57]. Consequently, some works have looked at expanding the VIMPC family to include different choices of divergences [74] and sampling distributions beyond Gaussians [45, 57].

Safety in Sampling-based MPC. Recent research efforts assess the risk associated with uncertain areas in the state space during the exploration phase, and enhance MPPI's safety by incorporating a risk penalty into its cost function [5, 85]. While these methods empirically enhance safety, they lack formal assurances. Another class of MPPI alternatives leverage an auxiliary tracking controller to follow the MPPI output trajectories [62, 77], improving robustness against unforeseeable disruptions, but these improvements are limited when the simulation-to-reality gap is significant. More recently, Control Barrier Functions (CBF) [3, 4, 82] have been used to provide formal safety guarantees for MPPI controllers [71, 84, 86]. However, these methods use distance functions as CBFs, and thus are unable to handle bounded input limits. Our proposed Neural Shield MPPI (NS-MPPI) controller, as a specific form of the proposed NS-VIMPC framework, effectively resolves this critical issue of safety under input constraints identified above.

Different Proposal Distributions for VIMPC. Many works investigate changing the sampling distribution of VIMPC to improve the performance of the sampling-based controller.

The MPPI variant in [83] uses covariance steering to assign a terminal covariance to the sampling distribution, but this relies on expert knowledge on how the covariance should be designed.

Normalizing flows are used in [60] to approximate the optimal sampling distribution, but this does not address the problem of recursive feasibility on its own. Another direction looks at changing the effective sampling distribution by modifying the underlying dynamical system to be more amenable to calculations. The works [62, 77] leverage an auxiliary tracking controller, thus changing the sampling distribution to be the output of the stable tracking controller. However, this requires the construction of such an auxiliary tracking controller, which can be difficult to perform for arbitrary nonlinear discrete-time systems. In contrast, our proposed resampling-based rollouts (RBR) can be viewed as a way of easily improving the proposal distribution *without* the need for any specialized problem structure.

Duality between Control and Inference. The proposed resampling strategy in our work is similar in spirit to [63, 80] in that factor graphs, a method used originally for estimation, is adapted for control purposes. In [89] a linear optimal control was used to improve the performance of particle filters. However, to the best of our knowledge, our approach is the first to adopt the resampling mechanism from particle filters to improve the performance of sampling-based MPC controllers.

Control Barrier Functions. Designing CBFs with control-invariant safe sets is not a trivial task. As a result, previous works only seek saturating CBFs that do not consider input constraints [48, 78, 81]. Some works use hand-tuned CBFs to prevent saturation [17, 75], which can lead to overly conservative safe sets. Other works develop CBFs with input constraints for specific types of systems [18, 19]. The emerging neural CBFs [64, 88, 90, 91] allow for more general dynamics utilizing machine learning techniques; however, they can still saturate the control limits.

Reachability. Recent works in the safety community have realized that the value function of the reachability problem is a valid CBF that takes input constraints into account [16, 36, 69]. In continuous-time, the reachability problem can be formulated as a Hamilton-Jacobi PDE [51] and is conventionally solved using grid-based numerical PDE solvers [53]. However, grid-based methods suffer from the curse of dimensionality and cannot be practically applied to systems with a state space larger than 5 dimensions [53]. To resolve this problem, recent works have looked at using learning-based methods at solving reachability problems in both continuous-time [7] and discrete-time [27, 37]. In particular, the value functions of both the *avoid* [16] and *reach-avoid* problems [36] are CBFs. Though early works focused on the use of *optimal* value functions [16], recent works *any* policy is a CBF, which is closely related to the ideas of backup-CBFs [15]. While this idea has been used for the avoid problem continuous-time settings [67] and for the reach-avoid problem in discrete-time settings [36], it has not so far been used for the avoid problem in discrete-time, which is the approach we take in this work with Discrete-time Policy Neural Control Barrier Function (DPNCBF). We summarize the relationship of this work with existing deep reachability-based methods in Table I.

TABLE I: Relationship between DPNCBF and other reachability methods.

Method	Avoid	Discrete	(Arbitrary) Policy-Conditioned
Bansal and Tomlin [7]	✓	✗	✗
Fisac et al. [27]	✓	✓	✗
Hsu et al. [37]	✗	✓	✗
So et al. [69]	✓	✗	✓
He et al. [36]	✗	✓	✓
DPNCBF (ours)	✓	✓	✓

III. Problem Formulation

We consider the discrete-time, nonlinear dynamics

$$x_{k+1} = f(x_k, u_k), \tag{1}$$

with state $x \in \mathcal{X} \subseteq \mathbb{R}^{n_x}$ and control $u \in \mathcal{U} \subseteq \mathbb{R}^{n_u}$. Let $\mathcal{U}^K$ denote the set of control trajectories of length K. Following the MPC setup, we assume that a cost function $J : \mathcal{U}^K \to \mathbb{R}$ encoding the desired behavior of the system is given. Moreover, we consider state constraints defined by an *avoid set* $\mathcal{A} \subset \mathcal{X}$ described as the superlevel set of some specification function h, i.e.,

$$\mathcal{A} \coloneqq \{x \in \mathcal{X} \mid h(x) > 0\}. \tag{2}$$

The goal is then to find a sequence of controls $\mathbf{u} = \{u_0, u_1, \dots, u_{K-1}\} \in \mathcal{U}^K$ that minimizes the cost function J while satisfying the system dynamics (1) and safety constraints $x_k \notin \mathcal{A}$ for all $k \geq 0$.

A. Variational Inference MPC For Sampling-based Optimization

To solve the above optimization problem, we deviate from traditional MPC solvers and use a variational inference MPC formulation to solve the problem via sampling-based optimization. To this end, we make use of the control-as-inference framework [46] to model the problem. Specifically, let o be a binary variable that indicates "optimality" such that, for all controls $\mathbf{u} \in \mathcal{U}^K$,

$$p(o = 1 \mid \mathbf{u}) \propto \exp(-J(\mathbf{u})), \tag{3}$$

We assume a prior $p_0(\mathbf{u})$ on the control trajectory. The "optimal" distribution can then be obtained via the posterior distribution

$$p(\mathbf{u} \mid o = 1) = \frac{p(o = 1 \mid \mathbf{u}) p_0(\mathbf{u})}{p(o = 1)} = Z^{-1} \exp(-J(\mathbf{u})) p_0(\mathbf{u}), \tag{4}$$

where $Z \coloneqq \int \exp(-J(\mathbf{u})) p_0(\mathbf{u}) \, \mathrm{d}\mathbf{u}$ denotes the unknown normalization constant. Since sampling from $p(\mathbf{u} \mid o = 1)$ is intractable, we use variational inference to approximate the posterior $p(\mathbf{u} \mid o = 1)$ with a tractable distribution $q_{\mathbf{v}}(\mathbf{u})$ parametrized by some vector $\mathbf{v}$ by minimizing the forward KL divergence, i.e.,

$$\min_{\mathbf{v}} \quad \mathbb{D}_{\mathrm{KL}}\big(p(\mathbf{u} \mid o = 1) \parallel q_{\mathbf{v}}(\mathbf{u})\big). \tag{5}$$

In the special case of $q_{\mathbf{v}}$ being a Gaussian distribution with mean $\mathbf{v}$ and a fixed control input covariance Σ, intrinsic to the robotic system of interest [76], we can solve (5) in closed-form to obtain the optimal $\mathbf{v}^*$ as (see Section B1 for details)

$$\mathbf{v}^* = \mathbb{E}_{p(\mathbf{u}|o=1)}[\mathbf{u}]. \tag{6}$$

While this expectation cannot be readily computed because $p(\mathbf{u} \mid o = 1)$ is intractable to sample from, we can use importance sampling to change the sampling distribution to some other distribution $r(\mathbf{u})$ that is easier to sample from, leading to

$$\mathbf{v}^* = \mathbb{E}_{r(\mathbf{u})}\left[\frac{p(\mathbf{u} \mid o=1)}{r(\mathbf{u})}\mathbf{u}\right] \tag{7}$$

$$= \mathbb{E}_{r(\mathbf{u})}\Bigg[\underbrace{\frac{Z^{-1}\exp(-J(\mathbf{u}))p_0(\mathbf{u})}{r(\mathbf{u})}}_{:=\omega(\mathbf{u})}\mathbf{u}\Bigg]. \tag{8}$$

Using samples $\mathbf{u}^1, \dots, \mathbf{u}^N$ drawn from r, we compute a Monte Carlo estimate $\hat{\mathbf{v}}$ of the optimal control sequence $\mathbf{v}^*$ (see Section B2 for details) as follows,

$$\hat{\mathbf{v}} = \sum_{i=1}^{N} \tilde{\omega}^i \mathbf{u}^i, \tag{9}$$

$$\tilde{\omega}^i := \frac{\omega(\mathbf{u}^i)}{\sum_{j=1}^{N}\omega(\mathbf{u}^j)}. \tag{10}$$

Note that the Z in $\omega(\mathbf{u})$ is canceled out in the computation of $\tilde{\omega}^i$ in (10) and hence can be ignored.

Remark 1 (Self-normalized importance sampling)**.** Note that the weights $\tilde{\omega}^i$ in (9) are **not** from the regular importance sampling estimates in (8) due to the normalization by the sum of the weights in (10). Instead, (9) is a self-normalized importance sampling estimator (SNIS), which uses an *estimate* of the weights ω but results in a biased, though asymptotically unbiased, estimator. This fact is not present in many existing works on both VIMPC (e.g., [57, 74]) and MPPI (e.g., [76]). See Section B2 for more details.

Remark 2 (Connections to MPPI and CEM)**.** In the case where we choose $p_0(\mathbf{u}) = q_{\mathbf{0}}(\mathbf{u})$ and $r(\mathbf{u}) = q_{\bar{\mathbf{v}}}(\mathbf{u})$ for some previous estimate of the optimal control sequence $\bar{\mathbf{v}}$, the above variational inference MPC framework reduces to MPPI (see Section B3 for details). Moreover, Cross-Entropy Method (CEM) also falls in the VIMPC framework [57, 74].

B. Constraint Handling In Variational Inference MPC

One advantage of sampling-based MPC is that it is simple to incorporate hard constraints. One can include an indicator function in the cost function that heavily penalizes constraint violations (e.g., see [8, 10, 76, 77]). Specifically, for some large constant $C > 0$, we can modify the cost function as

$$J_{\text{new}}(\mathbf{u}) = J(\mathbf{u}) + C\sum_{k=0}^{K}\mathbb{1}_{x_k \in \mathcal{A}}. \tag{11}$$

From the inference perspective (3), we can interpret J_{new} (11) as saying that $o = o_{\text{cost}} \wedge o_{\text{constraint}}$, where,

$$p(o_{\text{cost}} = 1 \mid \mathbf{u}) \propto \exp(-J(\mathbf{u})), \tag{12}$$

$$p(o_{\text{constraint}} = 1 \mid \mathbf{u}) \propto \exp\left(-C\sum_{k=0}^{K}\mathbb{1}_{x_k \in \mathcal{A}}\right). \tag{13}$$

However, a problem with (13) is that it looks at safety only within the prediction horizon and does not consider the probability of staying safe beyond the prediction horizon. To tackle this problem, we will use a discrete-time control barrier function (DCBF), which we introduce in the next section, to enforce that the states remain within a control-invariant set.

C. Discrete-time Control Barrier Functions (DCBF)

A discrete-time control barrier function (DCBF) [84] associated to the avoid set $\mathcal{A}$ is a function $B : \mathcal{X} \to \mathbb{R}$ such that[1]

$$B(x) > 0, \quad \forall x \in \mathcal{A}, \tag{14a}$$

$$B(x) \leq 0 \implies \inf_{u \in \mathcal{U}} B(f(x,u)) - B(x) \leq -\alpha(B(x)), \tag{14b}$$

where α is an extended class-κ function [82]. As in [84], we restrict our attention to the class of linear extended class-κ functions, i.e.,

$$\alpha(B(x)) = a \cdot B(x), \quad a \in (0,1). \tag{15}$$

The following theorem from [84] proves that a controller satisfying the condition (14b) renders the sublevel set $\mathcal{S} = \{x \mid B(x) \leq 0\}$ forward-invariant.

Theorem 1 ([84, Property 3.1])**.** *Any control policy $\pi : \mathcal{X} \to \mathcal{U}$ satisfying the condition*

$$B(f(x, \pi(x))) - B(x) \leq -\alpha(B(x)), \tag{16}$$

renders the sublevel set $\mathcal{S} = \{x \mid B(x) \leq 0\}$ forward-invariant.

Hence, one way to guarantee recursive feasibility, and thus safety beyond the prediction horizon, is to enforce the condition (16) at every time step of the optimization problem. Another point to note is that $\mathcal{S}$ is a control-invariant set, and thus the constraint $B(x_K) \leq 0$ can be imposed as a terminal state constraint to guarantee recursive feasibility for MPC as is done classically [14, 42]. Thus, one can try to enforce these constraints by incorporating them into the cost function as in (11) [84], i.e., modify the cost according to one of the following options,

$$J_{\text{new}}(\mathbf{u}) = J(\mathbf{u}) + C\sum_{k=0}^{K}\mathbb{1}_{B(f(x,u))-B(x)>-\alpha(B(x))}, \tag{17}$$

$$J_{\text{new}}(\mathbf{u}) = J(\mathbf{u}) + C\sum_{k=0}^{K}\Big[B(f(x,u)) - B(x) + \alpha(B(x))\Big]_+. \tag{18}$$

[1]Note that we use the opposite sign convention as compared to [84].

However, this approach has two problems. First, for sufficiently large C, samples that violate the DCBF constraint will have a normalized weight of near zero, rendering these samples useless. Conservative DCBFs may cause the majority of the samples to violate the constraint (16) and hence have zero weight, resulting in poor estimates of the optimal control and wasted computation. Second, while constructing a function B that satisfies (14a) is relatively simple, it is much harder to construct a function B that also satisfies (14b), contrary to the case with (continuous-time) control barrier functions [69, 88] (see also Remark 3 below). Consequently, many works that integrate control barrier functions into MPC often only propose functions for which (14a) holds and not (14b) [84], rendering the safety guarantees of Theorem 1 invalid.

In the next section, we address these two problems via a novel resampling method that reuses computations from zero-weight samples and learns a DCBF that tries to respect both (14a) and (14b) using policy value functions.

Remark 3 (Differences between discrete-time and continuous-time control barrier functions under unbounded controls)**.** Note that in the continuous-time case where having an unbounded control space $\mathcal{U}$, control-affine dynamics, and a non-zero $\frac{\partial \dot{B}}{\partial u}$ are sufficient to guarantee that B is a valid control barrier function. This is because (14b) generally holds for a function B that satisfies (14a) in the continuous-time case under these assumptions. However, the same does not apply to discrete-time under general nonlinear dynamics f since (16) is nonlinear in u, let alone the fact that robotic systems in real life are unable to exert infinite forces and hence generally do not have unbounded controls.

Remark 4 (Constraint satisfaction in the variational inference framework)**.** One potential issue with incorporating DCBF constraints into the cost function J is that, while $p(\mathbf{u} \mid o = 1)$ may have zero density on the set of controls that violate the DCBF constraint, this does not necessarily hold for the approximating distribution q since we are minimizing the *forward* KL divergence [39]. In particular, the mean $\mathbf{v}$ of $q_{\mathbf{v}}$, which is the control to be used, may not satisfy the DCBF constraint. One way to guarantee that $\mathbf{v}$ does satisfy the DCBF constraint is to assume that the set of controls that satisfy the DCBF constraint is itself convex (see Appendix F). However, this is an unrealistic assumption that is often violated by state constraints such as obstacle avoidance. Despite these shortcomings, we observed in our experimental results that this method of enforcing DCBF constraints on $p(\mathbf{u} \mid o = 1)$ indeed drastically improved safety. Alternatively, this problem can be solved by checking for constraint satisfaction of the mean $\mathbf{v}$, and if not satisfied, replacing it with any of the rollouts that do satisfy the constraints, similar to [11]. We leave further theoretical exploration of this issue as future work.

IV. Neural Shield VIMPC

In this section, we propose Neural Shield VIMPC (NS-VIMPC), a sampling-based MPC paradigm that efficiently samples trajectories using a DCBF modeled using a neural network. We illustrate the proposed NS-VIMPC algorithm in Fig. 2.

A. Approximating DCBF Using Neural Policy Value Functions

Let x_k^π denote the state at time k following the control policy $\pi : \mathcal{X} \to \mathcal{U}$. Define the policy value function $V^{h,\pi}$ as,

$$V^{h,\pi}(x_0) \coloneqq \max_{k \geq 0} h(x_k^\pi). \tag{19}$$

We then have the following theorem.

Theorem 2. *$V^{h,\pi}$ satisfies* (14a) *and* (14b) *and is a DCBF.*

Proof: From the definition of $V^{h,\pi}$ (19), we have that

$$V^{h,\pi}(x_k) \geq h(x_k), \tag{20}$$
$$V^{h,\pi}(x_k) \geq V^{h,\pi}(f(x_k, \pi(x_k))). \tag{21}$$

Using the definition of the avoid set $\mathcal{A}$ (2) and (20), it follows that $V^{h,\pi}(x) > 0$ for all $x \in \mathcal{A}$, satisfying the first condition (14a) of a DCBF. When $V^{h,\pi}(x_k) \leq 0$, we have $-\alpha(V^{h,\pi}(x_k)) \geq 0$, and (21) implies that,

$$V^{h,\pi}(f(x_k, \pi(x_k))) - V^{h,\pi}(x_k) \leq 0 \leq -\alpha(V^{h,\pi}(x_k)). \tag{22}$$

Since $\pi(x_k) \in \mathcal{U}$, this implies the second condition (14b). Thus, the policy value function $V^{h,\pi}$ is a DCBF. ■

Although we have constructed a DCBF from (19), the challenge is that the policy value function $V^{h,\pi}$ cannot be easily evaluated at arbitrary states since the maximization in (19) is taken over an infinite horizon. To fix this, we train a neural network approximation $V_\theta^{h,\pi}$ of $V^{h,\pi}$, extending the approach of [69] to the discrete-time case. To begin, we first rewrite (19) in a dynamic programming form,

$$V^{h,\pi}(x_0) = \max\left\{ \max_{0 \leq k \leq T} h(x_k^\pi),\ V^{h,\pi}(x_T^\pi) \right\}. \tag{23}$$

We can then train a neural network $V_\theta^{h,\pi}$ to approximate the value function $V^{h,\pi}$ by minimizing the loss

$$L(\theta) = \left\| V_\theta^{h,\pi}(x_0) - \max\left\{ \max_{0 \leq k \leq T} h(x_k^\pi), V_\theta^{h,\pi}(x_T^\pi) \right\} \right\|^2, \tag{24}$$

over all states x_0. One problem, however, is that the minimizer of (24) is not unique. For example, if $h(x) \leq \bar{h}$ for all x, then $V_\theta^{h,\pi} = \bar{h}$ is a minimizer of (24) but does not necessarily satisfy (19). To fix this, we follow the approach of [27, 67] and, inspired by reinforcement learning [70], introduce a discount factor $\gamma \in (0, 1)$ to define the *discounted* value function

$$V^{h,\pi,\gamma}(x_k) = \max\left\{ h(x_k^\pi),\ (1-\gamma)h(x_k) + \gamma V^{h,\pi,\gamma}(x_{k+1}^\pi) \right\}, \tag{25}$$

and the corresponding loss L,

$$L(\theta) = \left\| V_\theta^{h,\pi,\gamma}(x_k) - \hat{V}_\theta^{h,\pi,\gamma}(x_k) \right\|^2, \tag{26}$$

$$\hat{V}_\theta^{h,\pi,\gamma}(x_k) = \max\left\{ h(x_k^\pi), (1-\gamma)h(x_k^\pi) + \gamma V_\theta^{h,\pi,\gamma}(x_{k+1}^\pi) \right\}. \tag{27}$$

Finally, instead of using the learned $V_\theta^{h,\pi,\gamma}$ directly in (16), we first take the maximum with h and use $\tilde{V}_\theta^{h,\pi,\gamma}$ defined as

$$\tilde{V}_\theta^{h,\pi,\gamma}(x) \coloneqq \max\{h(x), V_\theta^{h,\pi,\gamma}(x)\}. \tag{28}$$

This guarantees that $\tilde{V}_\theta^{h,\pi,\gamma}(x) \geq h(x)$ and hence the zero sublevel set of $\tilde{V}^{h,\pi,\gamma}$ will be a subset of h. Hence, imposing the state constraint $\tilde{V}^{h,\pi,\gamma}(x) \leq 0$ will, at the very least, prevent violations of the original state constraints during the prediction horizon, and potentially also induce a state constraint that is closer to the true control-invariant set than the original sublevel set of h.

Remark 5 (Neural Network Verification of DCBFs)**.** We emphasize that our goal here is to obtain a good approximation of a DCBF B_θ using a neural policy value function $V_\theta^{h,\pi,\gamma}$, and not necessarily to obtain a true DCBF. Verifying whether the learned $V_\theta^{h,\pi,\gamma}$ is a true DCBF requires neural network verification which can be intractable or inconclusive (see Appendix 1 in [41] on the NP-completeness of the NN-verification problem). This is especially true in the discrete-time case where the condition (16) may not be affine in the control u.

Nevertheless, as we show later, empirical results show that using an approximation of a DCBF is sufficient for enabling the use of much shorter prediction horizons without sacrificing safety.

B. Efficient Sampling Using Resampling-based Rollouts

We tackle the problem of wasted samples with zero weights by drawing inspiration from the sequential Monte Carlo [23] and particle filter [33] literature, and by performing a per-timestep resampling during the rollout, which we call Resampling-Based Rollouts (RBR). Specifically, the control-as-inference problem formulation (3) gives us a *temporal* decomposition of $p(o_{\text{constraint}} = 1 \mid \mathbf{u})$ (13) in the case of state constraints when $C \to \infty$, as follows

$$p(o_{\text{constraint}} = 1 \mid \mathbf{u}) \propto \prod_{k=0}^{K} \mathbb{1}_{x_k \notin \mathcal{A}}. \tag{29}$$

Hence, we can write the posterior $p(\mathbf{u} \mid o = 1)$ as

$$p(\mathbf{u} \mid o = 1) \propto p(o_{\text{cost}} = 1 \mid \mathbf{u}) p(\mathbf{u}) \left(\prod_{k=0}^{K} \mathbb{1}_{x_k \notin \mathcal{A}} \right). \tag{30}$$

By treating the term on the right as a "measurement model," particle filtering [33] can be used to solve the control-as-inference problem. The update of the particle filter weights $\hat{w}_k^i$ at time step k for particle i is written as

$$\hat{w}_k^i = \hat{w}_{k-1}^i \mathbb{1}_{x_k \notin \mathcal{A}}. \tag{31}$$

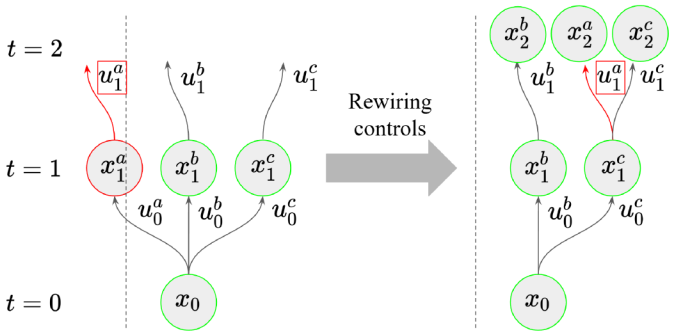


Fig. 3: **Resampling-based Rollouts (RBR).** Inspired by particle filtering, at each step of the trajectory rollout, we uniformly resample any samples that violate the state constraints among the set of safe samples. In this example, the two particles at states x_1^b, x_1^c satisfy the constraints and have weights $\hat{w}_1^b = \hat{w}_1^c = 1$. The particle at state x_1^a violates constraints and thus has a weight of $\hat{w}_1^a = 0$. Consequently, both x_1^a and the control prefix u_0^a are resampled away and replaced with equal probability by either x_1^b or x_1^c and their control prefixes respectively (in this example, x_1^c was chosen). Note that u_1^a can be left untouched, though is now applied from x_1^c instead of the unsafe state x_1^a.

Then, we *resample* the particles with probability proportional to their weights $\hat{w}_k^i$ to obtain a new set of particles $\mathbf{u}^i$. Due to the indicator function in the weight update (31), the weights are either 0 or 1. Assuming there exists a particle that satisfies the state constraint, applying systematic resampling [43] results in "rewiring" particles that violate the constraint to particles that still maintain safety, reusing the computation from zero-weight samples (see Fig. 3). If all particles violate the constraint, we do not resample. In this case, since we want to minimize constraint violations, we still use the cost term (18) such that trajectories with higher constraint violations have higher costs. Overall, at each timestep except for the last, we check constraint satisfaction for each particle. This gives a total of $N(K-1)$ queries per rollout.

While we can prove that this resampling is unbiased from the particle filter perspective, we also provide a more direct proof of this fact without the analogy to particle filters.

Theorem 3. *For a probability density function* f *and set* S*, let* a *be sampled from the conditional density* $\mathsf{f}(\mathsf{x} \mid \mathsf{x} \in \mathsf{S})$*, and let* b *be sampled from the unconditional density* f*, such that* a *and* b *are independent. Define the "rewired" random variable* $\tilde{\mathsf{b}}$ *to be equal to* b *if* $\mathsf{b} \in \mathsf{S}$ *and* a *otherwise, i.e.,*

$$\tilde{\mathsf{b}} = \mathbb{1}_{\mathsf{b} \in \mathsf{S}} \mathsf{b} + \mathbb{1}_{\mathsf{b} \notin \mathsf{S}} \mathsf{a} \tag{32}$$

Then, $\tilde{\mathsf{b}}$ *is also sampled from the conditional density* $\mathsf{f}(\mathsf{x} \mid \mathsf{x} \in \mathsf{S})$*, such that* $\tilde{\mathsf{b}}$ *and* a *have the same distribution,* $\tilde{\mathsf{b}} \overset{d}{=} \mathsf{a}$.

The proof is given in Section D2. Consequently, we can use $\tilde{\mathsf{b}}$ in a Monte Carlo estimator and still obtain unbiased estimates, as shown in the following Corollary (proof is given

in Section D3).

Corollary 1. *For any function w, the Monte Carlo estimate of $\mathbb{E}[w(\mathsf{x})]$ under the conditional density $\mathsf{f}(\mathsf{x} \mid \mathsf{x} \in \mathsf{S})$ using random variables a and $\tilde{\mathsf{b}}$ is unbiased, i.e.,*

$$\mathbb{E}[\frac{1}{2}w(\mathsf{a}) + \frac{1}{2}w(\tilde{\mathsf{b}})] = \mathbb{E}[w(\mathsf{x})]. \tag{33}$$

Intuitively, resampling improves the efficiency of the Monte Carlo estimator, since most of the samples do not violate the state constraints and hence contribute to the weighted sum with non-zero weight. We can also theoretically prove that resampling improves the variance of the resulting Monte Carlo estimator. In the following theorem, we show how resampling reduces the *exponential* growth of the variance on the prediction horizon to a constant factor in the limit as the number of samples N goes to infinity.

Theorem 4. *Let the horizon $K > 0$, consider $\mathcal{U} = [-1, 1]$, and define the avoid set and the dynamics such that $\mathbf{u} \in [0, 1]^K$ is safe, and unsafe otherwise. Let the prior distribution $p(\mathbf{u})$ be uniform on $[-1, 1]^K$ and let $p(o = 1 \mid \mathbf{u})$ be the indicator function for $\mathbf{u} \in [0, 1]$ such that the posterior distribution $p(\mathbf{u} \mid o = 1)$ is uniform on $[0, 1]^K$, and the proposal distribution $r(\mathbf{u}) = 1/2^K$ is uniform on $\mathcal{U}$. Then, the variance of the Monte Carlo estimator of the optimal control law*

$$\hat{\mathbf{v}} = \frac{1}{N}\sum_{i=1}^{N} \frac{p(\mathbf{u}^i \mid o = 1)}{r(\mathbf{u}^i)}\mathbf{u}^i, \tag{34}$$

grows exponentially in K, i.e.,

$$\mathrm{Var}[\hat{v}_k] = \frac{1}{N}\left(\frac{1}{3}2^K - \frac{1}{4}\right), \qquad k = 1, \ldots, K. \tag{35}$$

Using resampling, the variance is upper-bounded by,

$$\mathrm{Var}[\hat{v}_{k,\mathrm{resample}}] = O\left(\left(\frac{1}{1 - 2^{-N}}\right)^K + (1 - 2^{-N})^K\right). \tag{36}$$

The proof of Theorem 4 is given in Section D4. As shown in Theorem 4, although the variance is still exponential in K using resampling, the base of the exponential decreases to 1 exponentially in the number of samples N. In other words, in the limit as $N \to \infty$, the variance of the estimator using the proposed RBR is bounded by a constant factor. This novel approach reduces the variance of the Monte Carlo estimator of the optimal control law in a way that mirrors the relationship between Sequential Importance Sampling (i.e., particle filters without resampling), where the variance increases exponentially with the horizon length, and Sequential Monte Carlo (i.e., particle filters with resampling), where the (asymptotic) variance only increases linearly [24].

Another method to theoretically quantify the improvement in the variance of the estimator is via the *effective sample size* (ESS) [24, 25]. ESS is defined as the ratio between the variance of the estimator with N samples from the target and the variance of the SNIS estimator [25]. It can be interpreted as the number of samples simulated from the target pdf that would provide an estimator with variance equal to the performance of the N-sample SNIS estimator. However, since the ESS is computationally intractable, the *approximation* (made formal in [25])

$$\widehat{ESS} \coloneqq \frac{1}{\sum_{n=1}^{N} w_n^2}, \tag{37}$$

is more often used in practice. The following theorem shows that performing RBR results in either the same or higher $\widehat{ESS}$, given certain assumptions (proof is given in Section D5).

Theorem 5. *Let $\mathbf{w} = [w_1, \ldots, w_m, 0, \ldots, 0]$ denote the unnormalized weight vector without resampling, where the last $N-m$ entries are zero due to violating the safety constraints. Let $\mathbf{w}' = \mathbf{w} + \mathbf{c} = [w_1, \ldots, w_m, c_{m+1}, \ldots, c_N]$ denote the unnormalized weight vector resulting from safe resampling, where $\mathbf{c} = [0, \ldots, 0, c_{m+1}, \ldots, c_N]$. Let $\tilde{\mathbf{w}} = \mathbf{w}/\|\mathbf{w}\|_1$ and $\tilde{\mathbf{w}}' = \mathbf{w}'/\|\mathbf{w}'\|_1$ denote the normalized weights. Suppose that the weights of the resampled trajectories $\mathbf{c}$ are not "drastically larger" than the weights of the original trajectories $\mathbf{w}$, i.e.,*

$$\|\mathbf{c}\|_1 \le 2\frac{N}{N-1}\frac{\|\mathbf{w}\|_2^2}{\|\mathbf{w}\|_1}. \tag{38}$$

Then, the $\widehat{ESS}$ using RBR is no smaller than the $\widehat{ESS}$ without resampling, and is strictly greater if the inequality in (38) is strict. In other words,

$$\frac{1}{\|\tilde{\mathbf{w}}\|_2^2} \le \frac{1}{\|\tilde{\mathbf{w}}'\|_2^2}. \tag{39}$$

C. Summary of NS-VIMPC

We now summarize the NS-VIMPC algorithm, shown in Fig. 2.

Offline, we first approximate a DCBF by using the DPNCBF algorithm to learn the policy value function for a user-specified policy (Section IV-A). In our experiments, we chose this to be Shield MPPI (S-MPPI) [84]. Online, we use the learned DPNCBF to enforce the DPCBF constraint (16) to try to enforce safety beyond the prediction horizon. During sampling, we use RBR to efficiently sample control sequences $\{\tilde{\mathbf{u}}^i\}$ from the raw samples $\{\mathbf{u}^i\}$ to satisfy the DPNCBF constraint, thus improving the sample efficiency of the Monte Carlo estimate of the optimal control. The sampled control sequences $\{\tilde{\mathbf{u}}^i\}$ are then used to compute the estimate of the optimal control $\hat{\mathbf{v}}$ using (9). As in MPC fashion, we only execute the first control $\hat{\mathbf{v}}_0$, and $\hat{\mathbf{v}}$ is then used as the parameter vector of the sampling distribution $q_{\mathbf{v}}$ for the next iteration.

V. Simulations

We first performed simulation experiments to better understand the performance of the proposed Neural Shield VIMPC (NS-VIMPC) controller. Although many sampling-based MPC controllers fall under the VIMPC family with different choices

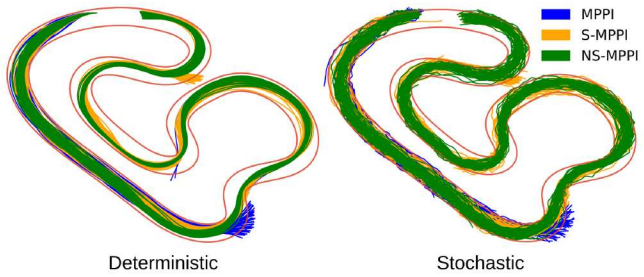


Fig. 4: **AutoRally Trajectories.** We visualize the trajectories of the three MPPI baselines under a challenging target velocity of 15 ms^{-1}. Both MPPI and S-MPPI veer off course and crash while NS-MPPI stays within the track even under Gaussian disturbances.

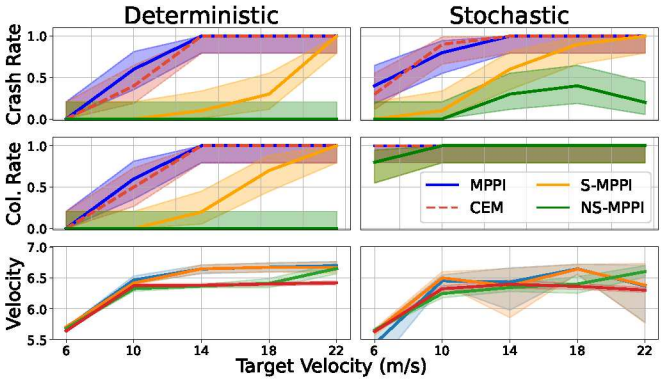


Fig. 5: **Varying target velocities on AutoRally.** Our NS-MPPI achieves the lowest crash and collision (Col.) rates under both deterministic and stochastic dynamics. While the collision rate is close to 1 for every method in the stochastic environment, NS-MPPI achieves a crash rate of near 0.

of the prior p_0 and r, we choose to instantiate the MPPI algorithm (see Section B1 for details), and call the resulting controller Neural Shield-MPPI (NS-MPPI).

Baseline methods. We compared NS-MPPI against the following sampling-based MPC methods.

- Baseline MPPI (MPPI) [76], which forward simulates a set of randomly sampled trajectories for optimal control.
- Shield MPPI (S-MPPI) [84], which extends MPPI by taking h in (2) to be a DCBF and by adding the DCBF constraint violation into the cost as in (18).[2]
- CEM [9], which samples trajectories similar to the baseline MPPI but with the weight $\omega = 1$ for only the k-lowest cost trajectories and 0 otherwise. This corresponds to an average of the k-lowest cost trajectories.

In all simulations, we use S-MPPI as the control policy π to learn the DPNCBF. We provide further details on the simulation experiments in Appendix A.

[2]We remove the local repair step from S-MPPI as this requires known gradients from the dynamics and would make the method gradient-based

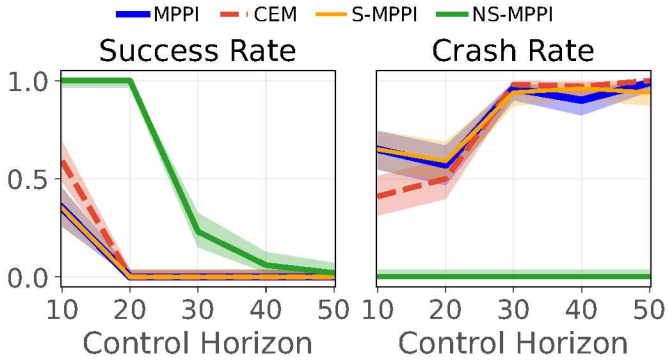


Fig. 6: **Varying control horizon on Drone.** (a) Only NS-MPPI has a crash rate of zero with a control horizon of 10, demonstrating the benefit of enforcing the DCBF constraint for maintaining safety beyond the prediction horizon.

A. *Simulations on AutoRally*

We first compare all methods on the AutoRally [31] testbed, a $1/5$ scale autonomous racing car. The goal for this task is to track a given fixed velocity without exiting the track. The vehicle *collides* when it contacts the track boundary and *crashes* when it fully exceeds the track boundary.

We tested our algorithm under both deterministic dynamics and stochastic dynamics with a Gaussian state disturbance added at each timestep. We visualize the resulting trajectories in Fig. 4 with a target velocity of 15 ms^{-1}. This is a very high target velocity, as previous works considered at most velocities of 8 ms^{-1} [84] or 9 ms^{-1} [28]. Under this challenging speed, both MPPI and S-MPPI frequently veer off course. In contrast, the proposed NS-MPPI successfully retains the vehicle within the confines of the track, thereby ensuring safety.

Next, we vary the target velocities, showing the resulting crash rate, collision rate, and velocity in Fig. 5 over 20 trials. With higher velocities, the vehicle has less time to turn, increasing the likelihood of leaving the track and colliding or crashing. We see that NS-MPPI consistently outperforms all other methods in both settings without having a significantly lower average velocity.

B. *Simulations on Drone*

We next tested our algorithm on Drone, a simulated planar quadrotor that incorporates ground effects arising from the intricate interaction between the blade airflow and the ground surface. The goal is for the drone to navigate as fast as possible through a narrow corridor close to the ground. We vary the control horizon for 500 sampled trajectories and plot the results in Fig. 6. Only NS-MPPI has a crash rate of zero over all control horizons, while all other controllers have high crash rate in this challenging task.

To understand why this is the case, we visualize the trajectories for NS-MPPI and S-MPPI with a control horizon of 10 in Fig. 7. Due to the DCBF constraint, NS-MPPI starts descending to avoid the obstacles even before any of the original collision constraints are violated. On the other hand, S-MPPI

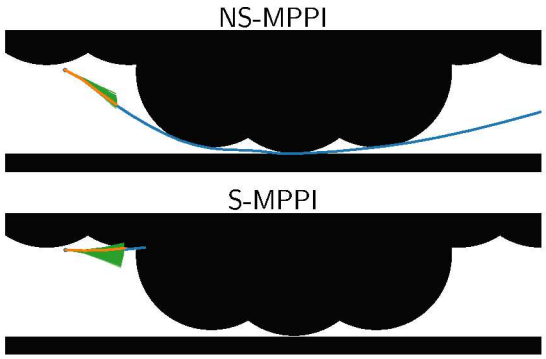


Fig. 7: **Differences between NS-MPPI and S-MPPI.** We compare the sampling trajectory $\mathbf{u}^i$ (green) and estimated optimal trajectory $\hat{\mathbf{v}}$ (orange) under a control horizon of 10. NS-MPPI descends early enough to avoid collisions due to the DCBF constraints despite none of the sampled trajectories violating any constraints due to the short control horizon.

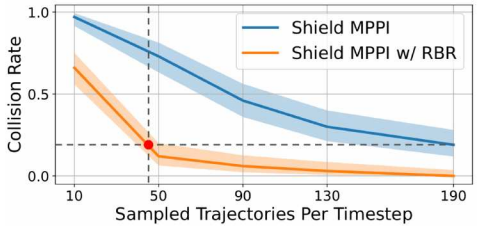


Fig. 8: **RBR needs 5X less samples.** On AutoRally, applying RBR to S-MPPI significantly reduces the number of sampled trajectories needed to achieve the same level of safety (190 to 40, horizontal line). From another perspective, RBR reduces the collision rate by 74% at 50 sampled trajectories.

is unaware of the obstacles beyond the prediction horizon and keeps accelerating rightward until collision. This suggests that enforcing the DCBF constraint even with an approximate $V_\theta^{h,\pi,\gamma}$ is beneficial for safety with short control horizons.

C. Deeper Investigation Into RBR

We next perform various case studies to better understand the sample efficiency benefits of the proposed RBR method.

RBR improves sample efficiency by 5X. To isolate the effects of RBR without the other improvements, we considered a new method that extends S-MPPI with RBR (S-MPPI w/ RBR), and compared it against the original S-MPPI across varying numbers of sampled trajectories on AutoRally over 100 trials in Fig. 8. S-MPPI with RBR achieves the same collision rates as S-MPPI without RBR while using 5 times *fewer* trajectories. This matches our expectations, both from the intuition that RBR results in a more closely aligned proposal distribution (e.g., see Fig. 1), and from the theoretical results in Section IV-B that show that RBR improves the quality of the estimator.

RBR achieves larger $\widehat{\text{ESS}}$. RBR theoretically improves the $\widehat{ESS}$, as mentioned in Section IV-B. We verified this claim empirically on AutoRally and plot the results in Fig. 9. While MPPI and S-MPPI exhibit values of $\widehat{ESS}$ concentrated

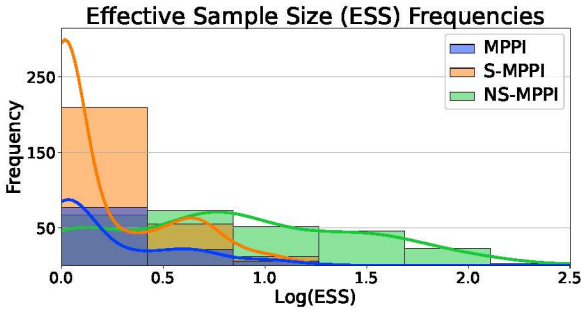
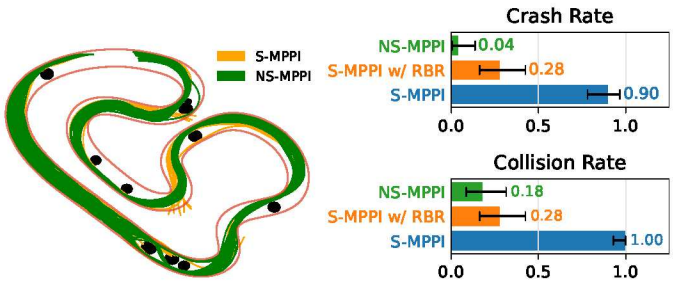


Fig. 9: **NS-MPPI has larger $\widehat{\text{ESS}}$.** The proposed NS-MPPI achieves larger effective sample sizes, verifying that RBR enables more efficient use of samples and computations.



(a) (b)

Fig. 10: **AutoRally with obstacles.** (a) While NS-MPPI consistently clears the entire track, S-MPPI often crashes into obstacles and walls. (b) NS-MPPI has 85% fewer crashes compared to S-MPPI. Adding RBR alone to S-MPPI fixes 62% of the crashes made by S-MPPI.

near 1, the $\widehat{ESS}$ values for NS-MPPI are more uniformly distributed, indicating a more extensive utilization of the sampled trajectories and thus a better use of the available computational resources.

RBR is especially beneficial in harder environments. To further stress-test RBR, we ran 50 trials with 100 sampled trajectories on a harder variant of AutoRally with obstacles of various sizes. Note that the presence of obstacles close to the center of the track violates the assumption in Remark 4 regarding the safety of the mean $\mathbf{v}$. Comparing the trajectories from S-MPPI and NS-MPPI in Fig. 10a, S-MPPI is unable to avoid crashing into walls and obstacles since it relies on a heuristic DCBF. On the other hand, NS-MPPI avoids crashes in almost all runs. We also compare against S-MPPI w/ RBR and plot the crash and collision rates in Fig. 10b, where we see that RBR constitutes a large fraction of the performance improvements, being responsible for 72% of the crash rate reduction between S-MPPI and NS-MPPI. Nevertheless, RBR alone is not sufficient, and the use of an approximate DPNCBF is still required to bring the crash rate to near 0.

To demonstrate the enhancement in sampling efficiency achieved by the proposed sampling method, we provide visual representations of trajectory sampling distributions obtained from simulations in environments with obstacles in Fig. 1. As demonstrated by Fig. 1 (b), the RBR concentrates trajectory

TABLE II: Performance and Timing Comparison

Controller	Crash Rate	Collision Rate	Control Rate (Hz)
NS-MPPI	0.04	0.06	73.64
S-MPPI	0.46	0.70	107.31
MPPI	0.98	1.00	130.05
CEM	1.00	1.00	88.17

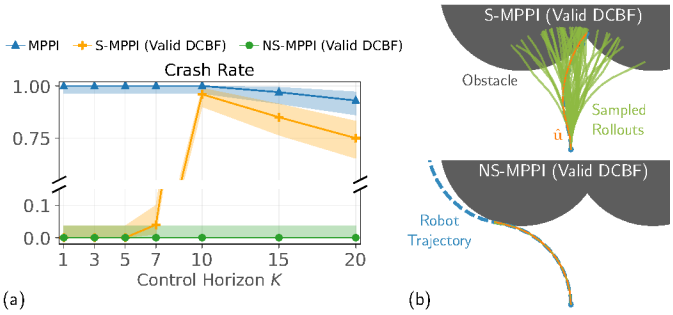


Fig. 11: **RBR improves performance even with *valid* DCBF.** (a) As the estimator variance grows exponentially with K, S-MPPI crashes at larger K *despite* using a valid DCBF. Using RBR (NS-MPPI) mitigates this, maintaining safety across all horizons. (b) We visualize this exponential growth in variance by comparing the rollouts of the two methods at $K = 10$. S-MPPI is unable to sample a control sequence that turns left for all $K = 10$ steps with only $N = 50$ samples.

samples within a feasible narrow passage that satisfies the DCBF safety criteria. Conversely, the standard sampling approach, exemplified by Fig. 1 (a), produces a sparse distribution, misallocating samples to unsafe zones and thereby failing to sufficiently explore safe areas. This inadequacy leads to a collision with an obstacle.

Enhanced sampling efficiency enables safe real-time planning on a CPU. We compare the computation times of different VI-MPC controllers on AutoRally with a $12\,\mathrm{m\,s^{-1}}$ target speed, horizon of $K = 15$ and small sample size of $N = 30$ trajectories per optimization iteration (Table II). NS-MPPI achieves the lowest crash and collision rates, albeit with a slightly lower control rate of $> 70\,\mathrm{Hz}$ that suffices for most robotic tasks. S-MPPI suffers from over 11 times larger crash and collision rates, while both MPPI and CEM controllers exhibit crash and collision rates close to 1.

RBR improves performance even with a *valid* DCBF. We further investigate whether RBR is beneficial when we have a *valid* DCBF by performing experiments on Dubins, a Dubins car that travels with fixed velocity, using only $N = 50$ samples (Fig. 11). Here, NS-MPPI (Valid DCBF) uses RBR while S-MPPI (Valid DCBF) does not. Both methods use a valid DCBF. At $K = 1$, RBR has no effect since no resampling occurs, but both methods avoid crashes. However, since the estimator variance grows exponentially in the horizon length K (Section IV-B), S-MPPI crashes at $K \geq 7$, where $N = 50$ samples is not enough to sample a trajectory that satisfies the DCBF constraints at all timesteps. Using RBR mitigates this

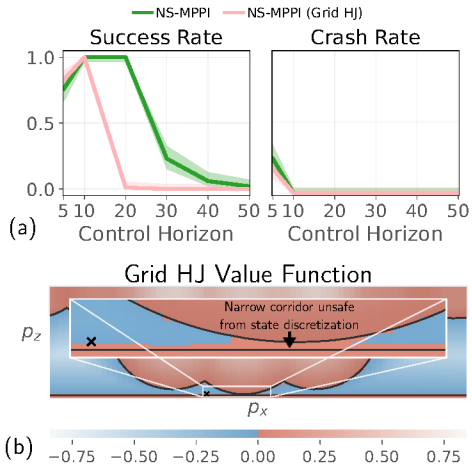


Fig. 12: **Value functions from grid-based solvers suffer in multi-fidelity problems.** (a) Using a value function from a grid-based HJ reachability solver results in much lower success rates at longer control horizons compared to the learned DPNCBF. (b) Due to state discretization, the narrow corridor is marked unsafe, causing the drone to be overly conservative and not cross the corridor to remain safe.

exponential growth, and NS-MPPI remains safe across all time horizons. See Appendix G for more details and additional plots.

D. Comparing DPNCBF against grid-based reachability.

Finally, we perform case studies to investigate the learned DPNCBF. In particular, we question whether the value function obtained via continuous-time techniques can be used in place of the proposed DPNCBF.

Grid-based reachability suffers with multi-fidelity problems. We compare against a grid-based reachability method solved using methods from *continuous-time* HJ reachability [53] (denoted NS-MPPI (Grid)) on Drone in Fig. 12. The height of the narrow corridor is much smaller than the length of the overall space. Consequently, given the 6-dimensional state-space of Drone, the grid size used by the grid-based solver is too coarse to capture the safe region in the narrow corridor accurately and marks the narrow corridor as unsafe, blocking traversal. This causes NS-MPPI (Grid HJ) to have a much lower success rate than NS-MPPI at longer control horizons.

Optimal safe controls from continuous-time reachability can cause crashes in discrete-time. Finally, we compare against a concurrent work DualGuard-MPPI [11] that uses the value function solved using continuous-time HJ reachability as a safety filter during the rollout process of MPPI on Dubins (Fig. 13), where we add an additional box-constraint $|\theta| \leq \pi/2$ on the heading. For this experiment, to rule out crashed caused by errors in the learned DPNCBF, we wrote a *discrete-time* grid-based solver and used the corresponding value function as the DCBF. Different from our method, DualGuard uses the value function as a safety filter during the rollout, replacing the original control with the *continuous-time* optimal safe control when the *current* value function is unsafe $V(x) > -\epsilon$, with a

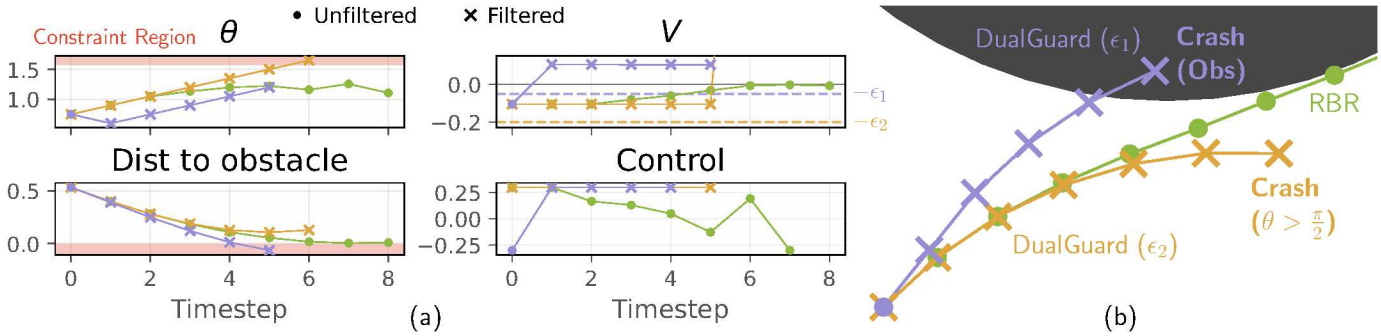


Fig. 13: **Optimal safe controls from continuous-time reachability can cause crashes in discrete-time.** The optimal safe control from continuous-time is different from the optimal control in discrete-time. Consequently, using the value function from continuous-time HJ reachability as a safety filter during the rollout process (i.e., concurrent work DualGuard-MPPI [11]) leads to crashes, even if a margin ϵ is added to try to account for this gap by being more conservative.

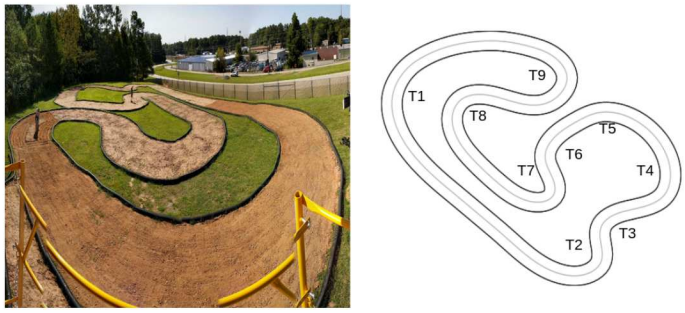
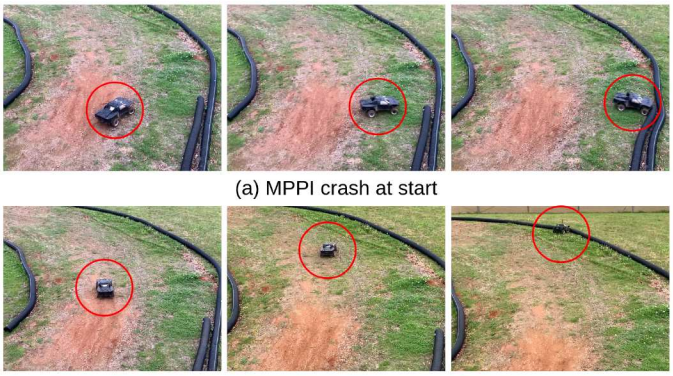


Fig. 14: **AutoRally Track Setup.** The site is equipped with a spacious carport, a 3-meters-tall observation tower, and a storage shed [29]. Each turn of the track is numbered as Ti, with T1 is the first turn and T9 is the last.



(a) MPPI crash at start

(b) Shield-MPPI crash at turn T1

Fig. 15: **Adversarial cost.** With a cost function that rewards crashes, both MPPI and S-MPPI crash. Only NS-MPPI completes 10 laps collision-free.

static margin ϵ added to try to account for the gap between continuous-time and discrete-time. However, Fig. 13 shows that neither small nor large value of ϵ can bridge this gap, and DualGuard-MPPI violates safety constraints in both cases. When the margin is small ($\epsilon = \epsilon_1$), the safety filter acts too late to prevent collision with the obstacle. When the margin is larger ($\epsilon = \epsilon_2$), the safety filter acts early and avoids the obstacle, but overshoots and violates the θ constraint instead due to the zero-order hold used in discrete-time.

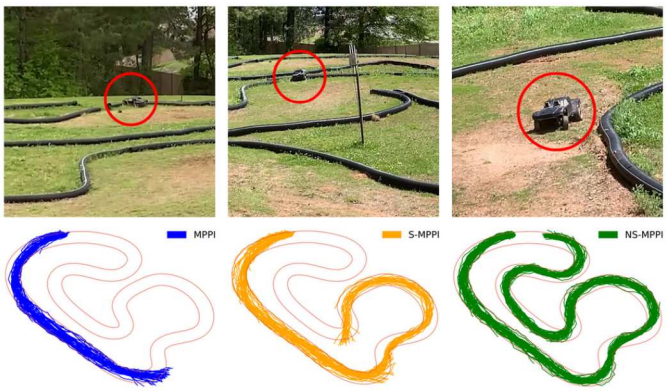
VI. Hardware Experiments

Finally, we deployed our method and other sampling-based MPC baselines on hardware using the AutoRally experimental platform [31]. Testing was conducted on a dirt track outlined by drainage pipes, as shown in Fig. 14. All computations use the onboard Intel i7-6700 CPU and Nvidia GTX-750ti GPU [31]. See Appendix E for more details on the hardware experiments.

A. Robustness Against Adversarial Costs

Robots depend on sensors such as cameras to gather information about their environment, and use the collected data to construct suitable cost functions. Deep Neural Networks (DNNs) are frequently utilized for object detection, classification, and semantic segmentation [92]. However, DNNs are susceptible to adversarial attacks [34], which can mislead the algorithms into generating incorrect outputs. These attacks often involve altering the appearance of the object of interest directly [47], eventually resulting in erroneous cost functions. In this hardware experiment, we tested the robustness of the proposed NS-MPPI to these erroneous or misspecified cost functions by using an *adversarial* cost function on the actual AutoRally hardware platform. Namely, instead of penalizing infeasible system states, we *reward* unsafe states by assigning negative costs, effectively incentivizing collisions. We compared against the S-MPPI and MPPI baselines.

The results are depicted in Fig. 15. As MPPI only uses cost information, it rapidly steers the AutoRally vehicle toward the closest boundary, resulting in a crash. S-MPPI crashes the vehicle at the first turn (T1) as well, but it covers a greater distance than MPPI. Only NS-MPPI achieves a collision-free completion of 10 laps. Nonetheless, there are noticeable oscillatory movements of the car, indicating susceptibility to attractive costs driving it toward boundary collisions.



(a) MPPI tracking 50m/s, collision at T2 (b)Shield-MPPI tracking 80m/s, collision at T7 (c) NS-MPPI tracking 80m/s without collisions

Fig. 16: **Safety under unsafe user inputs.** Using an erroneously large velocity target, both MPPI and S-MPPI crash. The visualization on the bottom row visualizes simulation trajectories that lead to crashes in the same turn.

B. Safety Under Unsafe User Input

Exceeding speed limits is a significant contributor to road accidents [55] due to noncompliance by many drivers. The proposed Shield VIMPC control framework can smartly manage speeds supervised by the NCBF to achieve safety, even given unsafe user inputs. To this end, we tested MPPI, S-MPPI, and the proposed NS-MPPI on the AutoRally hardware using large, unsafe target velocities to examine their safety performance. The results are shown in Fig. 16. Designating high target velocities encourages maximizing speed, thus disrupting the equilibrium in the controllers' cost structure by diminishing the emphasis on cost penalties associated with obstacles. As a result, MPPI causes the vehicle to deviate from the path, resulting in a crash at the second turn (T2), while S-MPPI impacts at the seventh turn (T7). In contrast, NS-MPPI ensures safety, preventing any accidents.

VII. Limitations

One limitation of our approach is that the DPNCBF is only an approximation of a DCBF. As such, the typical safety guarantees of DCBFs may not hold if the function is not a true DCBF. Moreover, as noted in Remark 4, the use of variational inference means that $\mathbf{v}^*$ may not satisfy the state constraints despite the optimal distribution $p(\mathbf{u} \mid o = 1)$ having zero density at states that violate the state constraints. While theoretically we were only able to show that $\mathbf{v}^*$ stays safe under the assumption that $\mathcal{U}^K_{\text{safe}}$ is convex, empirical results in Fig. 10 show that NS-MPPI can perform well despite the fact that $\mathcal{U}^K_{\text{safe}}$ is typically not convex.

VIII. Conclusion

In this study, we have adapted the policy neural control barrier function (PNCBF) to a discrete-time setting and utilized the resulting discrete-time policy neural control barrier function (DPNCBF) to supervise the proposed resampling-based rollout (RBR) method. This novel method enhances sampling efficiency and safety for all variational inference model predictive controllers (VIMPCs). Leveraging RBR, we developed the neural shield VIMPC (NS-VIMPC) control framework and demonstrated its benefits for safe planning through the novel neural shield model predictive path integral (NS-MPPI) controller. We conducted tests of the NS-MPPI, benchmarking its performance against state-of-the-art sampling-based MPC controllers using both an autonomous vehicle and a drone. Our simulations and experimental data indicate that the NS-MPPI outperforms existing VIMPC methods in terms of safety and sampling efficiency. The improved sampling efficiency allows NS-MPPI to reach similar performance levels as other VIMPC methods, while using significantly fewer sampled trajectories, facilitating real-time control on a CPU rather than necessitating costly GPU resources.

We have used the novel RBR method to concentrate sampled trajectories on safe regions. While this approach can significantly improve sampling efficiency in terms of safety, it does not improve other aspects of performance. To further enhance performance, we may, for instance, integrate a Control Lyapunov Function [21] or a Control Lyapunov Barrier Function [79] with the novel RBR approach to sample safe and higher-performance trajectories.

Acknowledgments

This work was funded by the National Science Foundation (NSF) under CAREER award CCF-2238030 and award CNS-2219755, and Office of Naval Research (ONR) under award N00014-23-1-2353.

References

[1] Manuel Acosta and Stratis Kanarachos. Tire lateral force estimation and grip potential identification using neural networks, extended Kalman filter, and recursive least squares. *Neural Computing and Applications*, 30(11): 3445–3465, 2018.

[2] Mohamadreza Ahmadi, Andrew Singletary, Joel W. Burdick, and Aaron D. Ames. Safe policy synthesis in multi-agent POMDPs via discrete-time barrier functions. In *IEEE 58th Conference on Decision and Control (CDC)*, pages 4797–4803, Nice, France, Dec. 11 – 13, 2019.

[3] Aaron D Ames, Xiangru Xu, Jessy W Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control*, 62(8):3861–3876, 2016.

[4] Aaron D. Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications. In *18th European Control Conference (ECC)*, pages 3420–3431, Naples, Italy, June 2019.

[5] Ermano Arruda, Michael J. Mathew, Marek Kopicki, Michael Mistry, Morteza Azad, and Jeremy L. Wyatt. Uncertainty averse pushing with model predictive path integral control. In *IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pages 497–502, Birmingham, UK, Nov. 15 – 17, 2017.

[6] Hagai Attias. Planning by probabilistic inference. In *International Workshop on Artificial Intelligence and Statistics*, pages 9–16, Key West, FL, Jan. 3 – 6, 2003.

[7] Somil Bansal and Claire J Tomlin. Deepreach: A deep learning approach to high-dimensional reachability. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1817–1824. IEEE, 2021.

[8] Lucas Barcelos, Alexander Lambert, Rafael Oliveira, Paulo Borges, Byron Boots, and Fabio Ramos. Dual Online Stein Variational Inference for Control and Dynamics. In *Proceedings of Robotics: Science and Systems*, Virtual, July 12 – 16, 2021.

[9] Homanga Bharadhwaj, Kevin Xie, and Florian Shkurti. Model-predictive control via cross-entropy and gradient-based optimization. In *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, volume 120 of *Proceedings of Machine Learning Research*, pages 277–286, Virtual, June 11 – 12, 2020.

[10] Mohak Bhardwaj, Balakumar Sundaralingam, Arsalan Mousavian, Nathan D. Ratliff, Dieter Fox, Fabio Ramos, and Byron Boots. STORM: An integrated framework for fast joint-space model-predictive control for reactive manipulation. In *Conference on Robot Learning*, pages 750–759, Auckland, New Zealand, Dec. 14 – 18, 2022.

[11] Javier Borquez, Luke Raus, Yusuf Umut Ciftci, and Somil Bansal. Dualguard mppi: Safe and performant optimal control by combining sampling-based mpc and hamilton-jacobi reachability. *arXiv:2502.01924*, 2025.

[12] Camille Brasseur, Alexander Sherikov, Cyrille Collette, Dimitar Dimitrov, and Pierre-Brice Wieber. A robust linear MPC approach to online generation of 3d biped walking motion. In *IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 595–601, Seoul, South Korea, Nov. 3 – 5, 2015.

[13] José Manuel Bravo, Daniel Limón, Teodoro Alamo, and Eduardo F. Camacho. On the computation of invariant sets for constrained nonlinear systems: An interval arithmetic approach. *Automatica*, 41(9):1583–1589, 2005.

[14] Wen-Hua Chen, John O'Reilly, and Donald J. Ballance. On the terminal region of model predictive control for non-linear systems with input/state constraints. *International Journal of Adaptive Control and Signal Processing*, 17 (3):195–207, 2003.

[15] Yuxiao Chen, Mrdjan Jankovic, Mario Santillo, and Aaron D Ames. Backup control barrier functions: Formulation and comparative study. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 6835–6841. IEEE, 2021.

[16] Jason J Choi, Donggun Lee, Koushil Sreenath, Claire J Tomlin, and Sylvia L Herbert. Robust control barrier–value functions for safety-critical control. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 6814–6821. IEEE, 2021.

[17] Andrew Clark. Verification and synthesis of control barrier functions. In *IEEE Conference on Decision and Control (CDC)*, pages 6105–6112, Austin, TX, Dec. 13 – 17, 2021.

[18] Wenceslao Shaw Cortez and Dimos V. Dimarogonas. Safe-by-design control for Euler–Lagrange systems. *Automatica*, 146:110620, 2022.

[19] Wenceslao Shaw Cortez, Xiao Tan, and Dimos V. Dimarogonas. A robust, multiple control barrier function framework for input constrained systems. *IEEE Control Systems Letters*, 6:1742–1747, 2021.

[20] Li Danjun, Zhou Yan, Shi Zongying, and Lu Geng. Autonomous landing of quadrotor based on ground effect modelling. In *Chinese Control Conference (CCC)*, pages 5647–5652, Hangzhou, China, July 28 – 30, 2015.

[21] Charles Dawson, Sicun Gao, and Chuchu Fan. Safe control with learned certificates: A survey of neural Lyapunov, barrier, and contraction methods for robotics and control. *IEEE Transactions on Robotics*, 39(3):1749–1767, 2023.

[22] Christopher De Sa, Megan Leszczynski, Jian Zhang, Alana Marzoev, Christopher R Aberger, Kunle Olukotun, and Christopher Ré. High-accuracy low-precision training. *arXiv preprint arXiv:1803.03383*, 2018.

[23] Arnaud Doucet, Nando De Freitas, and Neil Gordon. An introduction to sequential Monte Carlo methods. *Sequential Monte Carlo Methods in Practice*, pages 3–14, 2001.

[24] Arnaud Doucet, Adam M. Johansen, et al. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of Nonlinear Filtering*, 12(656-704):3, 2009.

[25] Víctor Elvira, Luca Martino, and Christian P. Robert. Rethinking the effective sample size. *International Statistical Review*, 90(3):525–550, 2022.

[26] Mirko Fiacchini, Teodoro Alamo, and Eduardo F. Camacho. On the computation of local invariant sets for nonlinear systems. In *IEEE Conference on Decision and Control*, pages 3989–3994, New Orleans, LA, Dec. 12 – 14, 2007.

[27] Jaime F. Fisac, Neil F. Lugovoy, Vicenç Rubies-Royo, Shromona Ghosh, and Claire J. Tomlin. Bridging Hamilton-Jacobi safety analysis and reinforcement learning. In *International Conference on Robotics and Automation (ICRA)*, pages 8550–8556, Montreal, Canada, May 20 – 24, 2019.

[28] Manan S. Gandhi, Bogdan Vlahov, Jason Gibson, Grady Williams, and Evangelos A. Theodorou. Robust model predictive path integral control: Analysis and performance guarantees. *IEEE Robotics and Automation Letters*, 6(2): 1423–1430, 2021.

[29] GeorgiaTech. AutoRally. Online. Available: https://autorally.github.io/, 2022. Accessed: Nov. 9, 2024.

[30] Philip E Gill, Walter Murray, and Michael A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Review*, 47(1):99–131, 2005.

[31] Brian Goldfain, Paul Drews, Changxi You, Matthew Barulic, Orlin Velev, Panagiotis Tsiotras, and James M Rehg. AutoRally: An open platform for aggressive autonomous driving. *IEEE Control Systems Magazine*,

39(1):26–55, 2019.

[32] Ravi Gondhalekar, Jun-ichi Imura, and Kenji Kashima. Controlled invariant feasibility—A general approach to enforcing strong feasibility in MPC applied to move-blocking. *Automatica*, 45(12):2869–2875, 2009.

[33] Neil J. Gordon, David J. Salmond, and Adrian F.M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. In *IEE Proceedings F (Radar and Signal Processing)*, volume 140, pages 107–113, 1993.

[34] Amira Guesmi, Muhammad Abdullah Hanif, Bassem Ouni, and Muhammad Shafique. Physical adversarial attacks for camera-based smart systems: Current trends, categorization, applications, research challenges, and future outlook. *IEEE Access*, 11:109617–109668, 2023.

[35] Emily Hannigan, Bing Song, Gagan Khandate, Maximilian Haas-Heger, Ji Yin, and Matei Ciocarlie. Automatic snake gait generation using model predictive control. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5101–5107, Paris, France, May 31 – Aug. 31, 2020.

[36] Tairan He, Chong Zhang, Wenli Xiao, Guanqi He, Changliu Liu, and Guanya Shi. Agile But Safe: Learning Collision-Free High-Speed Legged Locomotion. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024. doi: 10.15607/RSS.2024.XX.059.

[37] Kai-Chieh Hsu, Vicenç Rubies-Royo, Claire Tomlin, and Jaime F Fisac. Safety and Liveness Guarantees through Reach-Avoid Reinforcement Learning. In *Proceedings of Robotics: Science and Systems*, Virtual, July 2021. doi: 10.15607/RSS.2021.XVII.077.

[38] Wilson Jallet, Antoine Bambade, Nicolas Mansard, and Justin Carpentier. Constrained differential dynamic programming: A primal-dual augmented Lagrangian approach. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 13371–13378, Kyoto, Japan, Oct. 23 – 27, 2022.

[39] Ghassen Jerfel, Serena Wang, Clara Wong-Fannjiang, Katherine A. Heller, Yian Ma, and Michael I. Jordan. Variational refinement for importance sampling using the forward kullback-leibler divergence. In *Uncertainty in Artificial Intelligence*, pages 1819–1829, 2021.

[40] Zichao Jiang, Junyang Jiang, Qinghe Yao, and Gengchao Yang. A neural network-based pde solving algorithm with high precision. *Scientific Reports*, 13(1):4479, 2023.

[41] Guy Katz, Clark Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. In *Computer Aided Verification: 29th International Conference (CAV)*, pages 97–117, Heidelberg, Germany, July 24–28, 2017.

[42] Eric C. Kerrigan and Jan M. Maciejowski. Invariant sets for constrained nonlinear discrete-time systems with application to feasibility in model predictive control. In *Proceedings of the 39th IEEE conference on decision and control*, volume 5, pages 4951–4956, Sydney, Australia, Dec. 12 – 15, 2000.

[43] Genshiro Kitagawa. Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, 5(1):1–25, 1996.

[44] Jacob Knaup, Kazuhide Okamoto, and Panagiotis Tsiotras. Safe high-performance autonomous off-road driving using covariance steering stochastic model predictive control. *IEEE Transactions on Control Systems Technology*, 31 (5):2066–2081, 2023.

[45] Alexander Lambert, Fabio Ramos, Byron Boots, Dieter Fox, and Adam Fishman. Stein variational model predictive control. In *Proceedings of the Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pages 1278–1297, 16–18 Nov, 2021.

[46] Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

[47] Juncheng Li, Frank Schmidt, and Zico Kolter. Adversarial camera stickers: A physical camera-based attack on deep learning systems. In *International Conference on Machine Learning*, pages 3896–3904, Long Beach, CA, Jun. 10 – 15, 2019.

[48] Lars Lindemann and Dimos V. Dimarogonas. Control barrier functions for signal temporal logic tasks. *IEEE Control Systems Letters*, 3(1):96–101, 2018.

[49] Björn Lindqvist, Sina Sharif Mansouri, Ali-akbar Agha-mohammadi, and George Nikolakopoulos. Nonlinear MPC for collision avoidance and control of UAVs with dynamic obstacles. *IEEE Robotics and Automation Letters*, 5(4):6001–6008, 2020.

[50] Johan Löfberg. Oops! I cannot do it again: Testing for recursive feasibility in MPC. *Automatica*, 48(3):550–555, 2012.

[51] Kostas Margellos and John Lygeros. Hamilton–Jacobi formulation for reach–avoid differential games. *IEEE Transactions on Automatic Control*, 56(8):1849–1861, 2011.

[52] David Q. Mayne, James B. Rawlings, Christopher V. Rao, and Pierre O.M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000.

[53] Ian M. Mitchell. The flexible, extensible and efficient toolbox of level set methods. *Journal of Scientific Computing*, 35:300–329, 2008.

[54] Anusha Nagabandi, Kurt Konolige, Sergey Levine, and Vikash Kumar. Deep dynamics models for learning dexterous manipulation. In *Conference on Robot Learning*, pages 1101–1112, Virtual, Nov. 16 – 18, 2020.

[55] National Highway Traffic Safety Administration (NHTSA). Speeding. https://www.nhtsa.gov/risky-driving/speeding. Accessed: Jan. 27, 2025.

[56] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, 1999.

[57] Masashi Okada and Tadahiro Taniguchi. Variational inference MPC for Bayesian model-based reinforcement learning. In *Conference on robot learning*, pages 258–272, Virtual, Nov. 16 – 18, 2020.

[58] Art B. Owen. Monte Carlo theory, methods and examples,

2013.

[59] Thomas Power and Dmitry Berenson. Keep it simple: Data-efficient learning for controlling complex systems with simple models. *IEEE Robotics and Automation Letters*, 6(2):1184–1191, 2021.

[60] Thomas Power and Dmitry Berenson. Variational Inference MPC using Normalizing Flows and Out-of-Distribution Projection. In *Proceedings of Robotics: Science and Systems*, New York City, NY, June 27 – July 1, 2022.

[61] Thomas Power and Dmitry Berenson. Learning a generalizable trajectory sampling distribution for model predictive control. *IEEE Transactions on Robotics*, 40: 2111–2127, 2024.

[62] Jintasit Pravitra, Kasey A. Ackerman, Chengyu Cao, Naira Hovakimyan, and Evangelos A. Theodorou. L1-adaptive MPPI architecture for robust and agile control of multirotors. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7661–7666, Las Vegas, NV, Oct. 25 – 29, 2020.

[63] Mohamad Qadri, Paloma Sodhi, Joshua G. Mangelson, Frank Dellaert, and Michael Kaess. InCOpt: Incremental constrained optimization using the Bayes tree. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6381–6388, Kyoto, Japan, Oct. 23 – 27, 2022.

[64] Zengyi Qin, Kaiqing Zhang, Yuxiao Chen, Jingkai Chen, and Chuchu Fan. Learning safe multi-agent control with decentralized neural barrier certificates. *arXiv preprint arXiv:2101.05436*, 2021.

[65] Konrad Rawlik, Marc Toussaint, and Sethu Vijayakumar. On stochastic optimal control and reinforcement learning by approximate inference. In *International Joint Conference on Artificial Intelligence*, Beijing, China, Aug. 3 – 9, 2013.

[66] Reuven Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability*, 1:127–190, 1999.

[67] Oswin So and Chuchu Fan. Solving stabilize-avoid optimal control via epigraph form and deep reinforcement learning. In *Robotics: Science and Systems*, Daegu, South Korea, July 10 – 14, 2023.

[68] Oswin So, Ziyi Wang, and Evangelos A. Theodorou. Maximum entropy differential dynamic programming. In *International Conference on Robotics and Automation (ICRA)*, pages 3422–3428, Philadelphia, PA, May 23 – 27, 2022.

[69] Oswin So, Zachary Serlin, Makai Mann, Jake Gonzales, Kwesi Rutledge, Nicholas Roy, and Chuchu Fan. How to train your neural control barrier function: Learning safety filters for complex input-constrained systems. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 11532–11539, Yokohama, Japan, May 13 – 17, 2024.

[70] Richard S. Sutton and Andrew G. Barto. Reinforcement learning: An introduction, 2018.

[71] Chuyuan Tao, Hunmin Kim, Hyungjin Yoon, Naira Hovakimyan, and Petros Voulgaris. Control barrier function augmentation in sampling-based control algorithm for sample efficiency. In *American Control Conference (ACC)*, pages 3488–3493, Atlanta, GA, June 8 – 10, 2022.

[72] Marc Toussaint and Amos Storkey. Probabilistic inference for solving discrete and continuous state Markov decision processes. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 945–952, Pittsburgh, PA, June 25 – 29, 2006.

[73] Efstathios Velenis, Emilio Frazzoli, and Panagiotis Tsiotras. Steady-state cornering equilibria and stabilisation for a vehicle during extreme operating conditions. *International Journal of Vehicle Autonomous Systems*, 8(2-4): 217–241, 2010.

[74] Ziyi Wang, Oswin So, Jason Gibson, Bogdan Vlahov, Manan Gandhi, Guan-Horng Liu, and Evangelos Theodorou. Variational Inference MPC using Tsallis Divergence. In *Proceedings of Robotics: Science and Systems*, Virtual, July 12 – 16, 2021.

[75] Tianhao Wei and Changliu Liu. Safe control with neural network dynamic models. In *Learning for Dynamics and Control Conference*, pages 739–750, Palo Alto, CA, June 23 – 24, 2022.

[76] Grady Williams, Paul Drews, Brian Goldfain, James M. Rehg, and Evangelos A. Theodorou. Information-theoretic model predictive control: Theory and applications to autonomous driving. *IEEE Transactions on Robotics*, 34(6):1603–1622, 2018.

[77] Grady Williams, Brian Goldfain, Paul Drews, Kamil Saigol, James M. Rehg, and Evangelos A. Theodorou. Robust sampling based model predictive control with sparse objective information. In *Robotics: Science and Systems*, volume 14, Pittsburgh, PA, June 26 – 30, 2018.

[78] Sean Wilson, Paul Glotfelter, Li Wang, Siddharth Mayya, Gennaro Notomista, Mark Mote, and Magnus Egerstedt. The Robotarium: Globally impactful opportunities, challenges, and lessons learned in remote-access, distributed control of multirobot systems. *IEEE Control Systems Magazine*, 40(1):26–44, 2020.

[79] Zhe Wu, Fahad Albalawi, Zhihao Zhang, Junfeng Zhang, Helen Durand, and Panagiotis D. Christofides. Control Lyapunov-barrier function-based model predictive control of nonlinear systems. In *American Control Conference (ACC)*, pages 5920–5926, Milwaukee, WI, June 27 – 29, 2018.

[80] Mandy Xie, Alejandro Escontrela, and Frank Dellaert. A factor-graph approach for optimization problems with dynamics constraints. *arXiv preprint arXiv:2011.06194*, 2020.

[81] Bin Xu and Koushil Sreenath. Safe teleoperation of dynamic UAVs through control barrier functions. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 7848–7855, Brisbane, Australia, May 21 – 26, 2018.

[82] Xiangru Xu, Paulo Tabuada, Jessy W. Grizzle, and

Aaron D. Ames. Robustness of control barrier functions for safety critical control. *IFAC-PapersOnLine*, 48(27): 54–61, 2015.

[83] Ji Yin, Zhiyuan Zhang, Evangelos Theodorou, and Panagiotis Tsiotras. Trajectory distribution control for model predictive path integral control using covariance steering. In *International Conference on Robotics and Automation (ICRA)*, pages 1478–1484, Philadelphia, PA, May 23 – 27, 2022.

[84] Ji Yin, Charles Dawson, Chuchu Fan, and Panagiotis Tsiotras. Shield model predictive path integral: A computationally efficient robust MPC method using control barrier functions. *IEEE Robotics and Automation Letters*, 8(11):7106–7113, 2023.

[85] Ji Yin, Zhiyuan Zhang, and Panagiotis Tsiotras. Risk-aware model predictive path integral control using conditional value-at-risk. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 7937–7943, London, UK, May 29 – June 2, 2023.

[86] Ji Yin, Panagiotis Tsiotras, and Karl Berntorp. Chance-constrained information-theoretic stochastic model predictive control with safety shielding. *63rd IEEE Conference on Decision and Control*, pages 653–658, Dec. 16–19, 2024.

[87] Changxi You and Panagiotis Tsiotras. Vehicle modeling and parameter estimation using adaptive limited memory joint-state UKF. In *American Control Conference*, pages 322–327, Seattle, WA, May 24–26, 2017.

[88] Hongzhan Yu, Chiaki Hirayama, Chenning Yu, Sylvia Herbert, and Sicun Gao. Sequential neural barriers for scalable dynamic obstacle avoidance. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11241–11248, Detroit, MI, Oct. 1 – 5, 2023.

[89] Qinsheng Zhang, Amirhossein Taghvaei, and Yongxin Chen. An optimal control approach to particle filtering. *Automatica*, 151:110894, 2023.

[90] Songyuan Zhang, Kunal Garg, and Chuchu Fan. Neural graph control barrier functions guided distributed collision-avoidance multi-agent control. In *Conference on Robot Learning*, pages 2373–2392, Atlanta, GA, Nov. 6 – 9, 2023.

[91] Songyuan Zhang, Oswin So, Kunal Garg, and Chuchu Fan. GCBF+: A neural graph control barrier function framework for distributed safe multi-agent control. *IEEE Transactions on Robotics*, 2025.

[92] Bo Zhao, Jiashi Feng, Xiao Wu, and Shuicheng Yan. A survey on deep learning-based fine-grained object classification and semantic segmentation. *International Journal of Automation and Computing*, 14(2):119–135, 2017.

APPENDIX A
DETAILS ON THE SIMULATION EXPERIMENTS

A1 AutoRally Environment

The numerical simulations use the AutoRally platform, a 1/5-scale autonomous vehicle system capable of executing high-speed, limit-handling maneuvers in complicated environments [31]. Figure17 shows the AutoRally hardware. In this study, we define a crash as an event when the vehicle makes contact with the track boundary, constructed from soft drainage pipes, resulting in the inability to continue driving. We define a collision as an event when the vehicle contacts the track boundary but can continue driving.

Fig. 17: AutoRally chassis and compute box [29].

A2 AutoRally Dynamics Modelling

We model the AutoRally using a rear-wheel drive, single-track bicycle model [44]. The vehicle system is represented in curvilinear coordinates using the centerline of the track as the reference curve. Using a curvilinear coordinate system provides a more intuitive interpretation of the vehicle's position and heading relative to the track, as compared to Cartesian coordinates.

To this end, we consider a nonlinear, continuous-time system,

$$\dot{x} = F(x, u), \tag{A.1}$$

where the system state of the AutoRally is given by,

$$x = \left[v_x, v_y, \dot{\psi}, \omega_F, \omega_R, e_\psi, e_y, s\right]^\intercal, \tag{A.2}$$

and where v_x is the longitudinal velocity, v_y is the lateral velocity, and $\dot{\psi}$ is the yaw rate. The front and rear wheel angular velocities are denoted by ω_F and ω_R. The yaw error and the lateral distance error from the centerline of the track are denoted by e_ψ and e_y, respectively (see Fig. 18). The variable s is the curvilinear position along the track centerline. The control input to the system (A.1) is,

$$u = \left[\delta, T\right]^\intercal, \tag{A.3}$$

where δ is the steering angle and T denotes the values for throttle input (if positive) or braking input (if negative). The dynamics of a single-track dynamic bicycle model [44] used to model (A.1) are given by,

$$\dot{v}_x = \frac{f_{Fx}\cos\delta - f_{Fy}\sin\delta + f_{Rx}}{m} + v_y\dot{\psi}, \tag{A.4a}$$

$$\dot{v}_y = \frac{f_{Fx}\sin\delta + f_{Fy}\cos\delta + f_{Ry}}{m} - v_x\dot{\psi}, \tag{A.4b}$$

$$\ddot{\psi} = \frac{\left(f_{Fy}\cos\delta + f_{Fx}\sin\delta\right)\ell_F - f_{Ry}\ell_R}{I_z}, \tag{A.4c}$$

$$\dot{\omega}_F = -\frac{r_F}{I_{\omega F}} f_{Fx}, \tag{A.4d}$$

$$\dot{\omega}_R = \Theta(\omega_R, T), \tag{A.4e}$$

$$\dot{e}_\psi = \dot{\psi} - \frac{v_x\cos e_\psi - v_y\sin e_\psi}{1 - \rho(s)e_y}\rho(s), \tag{A.4f}$$

$$\dot{e}_y = v_x\sin e_\psi + v_y\cos e_\psi, \tag{A.4g}$$

$$\dot{s} = \frac{v_x\cos e_\psi - v_y\sin e_\psi}{1 - \rho(s)e_y}, \tag{A.4h}$$

where m and I_z are the vehicle's mass and moment of inertia about the vertical axis, respectively. The radius of the front wheel is r_F, and the moment of inertia of the front wheel about the front axle is $I_{\omega F}$. The curvature of the track centerline

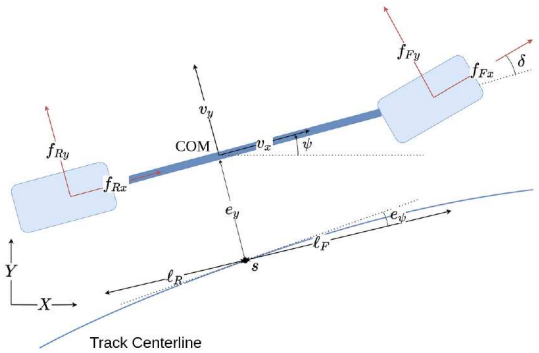


Fig. 18: Schematic of the dynamic bicycle model.

at position s is $\rho(s)$. The front and rear tire frictional forces are denoted by $f_{Fx}, f_{Fy}, f_{Rx}, f_{Ry}$, where the subscripts F, R indicate front and rear tires and x, y indicate longitudinal and lateral directions. These frictional forces are computed using the ellipse model in [73],

$$f_{ix} = f_{iz}\mu_{ix}, \tag{A.5a}$$
$$f_{iy} = f_{iz}\mu_{iy} + \Phi_i(\delta, \alpha_i), \tag{A.5b}$$

where $i = F, R$ indicates front or rear wheels, f_{iz} is the normal force acting on the tire. The tire friction coefficients μ_{ix}, μ_{iy} are dependent on wheel slip angles and the intrinsic tire parameters.

Inspired by [1], we use a neural network to model the residual error $\Phi_i(\delta, \alpha_i)$, where δ is steering angle, $\alpha_i = \arctan(v_{iy}/v_{ix})$ is wheel slip angle. Φ_i is trained using experimental data collected from the AutoRally. The rear wheel angular acceleration $\dot{\omega}_R$ is given by (A.4e). Modelling $\dot{\omega}_R$ is challenging, because it is determined by a variety of mechanical and physical conditions, including nonlinear tire behavior, dynamic load transfer, road surface irregularities and the complex interaction between these components. To this end, we use a data-driven approach and utilize a neural network $\Theta(\cdot, \cdot)$ trained on experimental data to model the rear wheel angular acceleration $\dot{\omega}_R$.

The nonlinear, continuous-time system (A.1) is discretized and converted to a discrete-time system using Euler integration,

$$x_{k+1} = f(x_k, u_k) = x_k + F(x_k, u_k)\Delta t, \tag{A.6}$$

where $\Delta t = t_{k+1} - t_k$ is the discretization time interval. We employ an Unscented Kalman Filter (UKF) on the discrete-time system (A.6) to identify the vehicle and tire parameters using the approach in [87]. The resulting parameters of the vehicle model include the vehicle mass $m = 22$ kg, moment of inertia $I_z = 1.1$ kg $\cdot$ m^2, $l_F = 0.34$ m, $l_R = 0.23$ m, $I_{\omega F} = 0.10$ kg $\cdot$ m^2 and front wheel radius $r_F = 0.095$ m. The parameters in Pacejka's Magic Formula used by the tire friction elipse model [73] are computed as $\text{tire}_\text{B} = 4.1$, $\text{tire}_\text{C} = 0.95$, $\text{tire}_\text{D} = 1.1$.

A3 Drone Dynamics Modelling

We model the two-dimensional drone dynamics using the following equations,

$$\dot{v}_x = -\frac{F}{m}\sin\theta, \tag{A.7a}$$
$$\dot{v}_z = \frac{F}{m}\cos\theta - g, \tag{A.7b}$$
$$\ddot{\theta} = \frac{\tau}{I}, \tag{A.7c}$$
$$\tag{A.7d}$$

where the total thrust F and the torque τ are given by,

$$F = F_1 + F_2, \quad \tau = \ell(F_2 - F_1), \tag{A.8}$$

where ℓ is the length from the center of the drone to the center of each rotor, and F_1, F_2 are thrust forces generated by the left and right rotors, respectively. The thrust forces are computed considering the ground effect [20], given by,

$$F_1 = F_{in1}/\big(1 - \rho(\frac{r}{4z_{r1}})\big), \tag{A.9}$$
$$F_2 = F_{in2}/\big(1 - \rho(\frac{r}{4z_{r2}})\big), \tag{A.10}$$

where F_{in1}, F_{in2} are the input thrust commands for the left and right rotors, and ρ is an intrinsic property of the drone. The radius of the rotors is r and the heights from the rotors to the ground are z_{r1} and z_{r2}, which are given by,

$$z_{r1} = z - \ell \sin\theta, \tag{A.11}$$

$$z_{r2} = z + \ell \sin\theta, \tag{A.12}$$

where θ is the pitch angle.

APPENDIX B
PROOFS FOR VARIATIONAL INFERENCE MPC

B1 Derivations of the Variational Inference Updates

Let $q_{\mathbf{v}}$ be the density function of a Gaussian distribution with mean $\mathbf{v}$ and covariance Σ, i.e.,

$$q_{\mathbf{v}}(\mathbf{u}) = Z_q^{-1} \exp\left(-\frac{1}{2}(\mathbf{u}-\mathbf{v})^\top \Sigma^{-1}(\mathbf{u}-\mathbf{v})\right), \tag{B.1}$$

where the normalization constant Z_q is independent of the mean $\mathbf{v}$. We wish to solve the variational inference problem

$$\min_{\mathbf{v}} \quad \mathbb{D}_{\mathrm{KL}}\big(p(\mathbf{u} \mid o=1) \parallel q_{\mathbf{v}}(\mathbf{u})\big). \tag{B.2}$$

Expanding and ignoring terms unrelated to $\mathbf{v}$, we get

$$\mathbb{D}_{\mathrm{KL}}\big(p(\mathbf{u} \mid o=1) \parallel q_{\mathbf{v}}(\mathbf{u})\big) = \int p(\mathbf{u} \mid o=1) \log\left(\frac{p(\mathbf{u} \mid o=1)}{q_{\mathbf{v}}(\mathbf{u})}\right) \mathrm{d}\mathbf{u} \tag{B.3}$$

$$= \int p(\mathbf{u} \mid o=1) \log\left(p(\mathbf{u} \mid o=1)\right) \mathrm{d}\mathbf{u} - \int p(\mathbf{u} \mid o=1) \log(q_{\mathbf{v}}(\mathbf{u})) \, \mathrm{d}\mathbf{u} \tag{B.4}$$

$$= -\int p(\mathbf{u} \mid o=1) \log(q_{\mathbf{v}}(\mathbf{u})) \, \mathrm{d}\mathbf{u} + O(1) \tag{B.5}$$

$$= \int p(\mathbf{u} \mid o=1) \left(\frac{1}{2}(\mathbf{u}-\mathbf{v})^\top \Sigma^{-1}(\mathbf{u}-\mathbf{v}) + \log Z_q\right) \mathrm{d}\mathbf{u} + O(1) \tag{B.6}$$

$$= \mathbb{E}_{p(\mathbf{u}|o=1)}\left[\frac{1}{2}(\mathbf{u}-\mathbf{v})^\top \Sigma^{-1}(\mathbf{u}-\mathbf{v}) + \log Z_q\right] + O(1) \tag{B.7}$$

$$= \mathbb{E}_{p(\mathbf{u}|o=1)}\left[\frac{1}{2}(\mathbf{u}-\mathbf{v})^\top \Sigma^{-1}(\mathbf{u}-\mathbf{v})\right] + O(1). \tag{B.8}$$

Taking the derivative with respect to $\mathbf{v}$ and setting to zero, we get

$$\mathbb{E}_{p(\mathbf{u}|o=1)}\left[\Sigma^{-1}(\mathbf{u}-\mathbf{v}^*)\right] = 0. \tag{B.9}$$

Rearranging, yields

$$\mathbf{v}^* = \mathbb{E}_{p(\mathbf{u}|o=1)}\left[\mathbf{u}\right]. \tag{B.10}$$

B2 Derivations of the Self Normalized Importance Sampling Estimator

We next derive the self-normalized importance sampling (SNIS) estimator $\hat{\mathbf{v}}$ in (9) of $\mathbf{v}^*$ in (8). Note that, since $Z = \int \exp(-J(\mathbf{u}))p_0(\mathbf{u}) \, \mathrm{d}\mathbf{u}$ is unknown, we cannot compute the weights $\omega(\mathbf{u})$ in (8) directly. Hence, the regular importance sampled Monte Carlo estimator

$$\hat{\mathbf{v}} = \frac{1}{N}\sum_{i=1}^{N} \omega(\mathbf{u}^i)\mathbf{u}^i, \tag{B.11}$$

is not computable. Instead, we can use the SNIS estimator [58], which can be derived as follows. First, note that Z can be written as the expectation of $\exp(-J(\mathbf{u}))$, i.e.,

$$Z = \mathbb{E}_{p_0(\mathbf{u})}\left[\exp(-J(\mathbf{u}))\right]. \tag{B.12}$$

Hence, one idea is to reuse the samples $\mathbf{u}^i \sim r$ to compute an estimate $\hat{Z}$ of Z, that is,

$$\hat{Z} = \frac{1}{N}\sum_{i=1}^{N} \exp(-J(\mathbf{u}^i))p_0(\mathbf{u}^i)/r(\mathbf{u}^i), \tag{B.13}$$

where $\hat{Z}$ is a "normal" importance sampled Monte Carlo estimator of Z. If we use the estimate $\hat{Z}$ to compute the weights $\hat{\omega}(\mathbf{u})$ in the normal importance sampled Monte Carlo estimator $\hat{\mathbf{v}}$ (B.11), we obtain the SNIS estimator as in (9) and (8)

$$\hat{\mathbf{v}} = \frac{1}{N}\sum_{i=1}^{N}\hat{\omega}(\mathbf{u}^i)\mathbf{u}^i, \tag{B.14}$$

$$= \frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{\hat{Z}}\frac{\exp(-J(\mathbf{u}^i))p_0(\mathbf{u}^i)}{r(\mathbf{u}^i)}\right)\mathbf{u}^i, \tag{B.15}$$

$$= \frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{\frac{1}{N}\sum_{j=1}^{N}\exp(-J(\mathbf{u}^j))p_0(\mathbf{u}^j)/r(\mathbf{u}^j)}\frac{\exp(-J(\mathbf{u}^i))p_0(\mathbf{u}^i)}{r(\mathbf{u}^i)}\right)\mathbf{u}^i, \tag{B.16}$$

$$= \frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{\frac{1}{N}\sum_{j=1}^{N}Z^{-1}\exp(-J(\mathbf{u}^j))p_0(\mathbf{u}^j)/r(\mathbf{u}^j)}\frac{Z^{-1}\exp(-J(\mathbf{u}^i))p_0(\mathbf{u}^i)}{r(\mathbf{u}^i)}\right)\mathbf{u}^i, \tag{B.17}$$

$$= \frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{\frac{1}{N}\sum_{j=1}^{N}Z^{-1}\exp(-J(\mathbf{u}^j))p_0(\mathbf{u}^j)/r(\mathbf{u}^j)}\omega(\mathbf{u}^i)\right)\mathbf{u}^i, \tag{B.18}$$

$$= \frac{1}{N}\sum_{i=1}^{N}\left(\frac{\omega(\mathbf{u}^i)}{\frac{1}{N}\sum_{j=1}^{N}\omega(\mathbf{u}^j)}\right)\mathbf{u}^i, \tag{B.19}$$

$$= \sum_{i=1}^{N}\left(\frac{\omega(\mathbf{u}^i)}{\sum_{j=1}^{N}\omega(\mathbf{u}^j)}\right)\mathbf{u}^i, \tag{B.20}$$

$$= \sum_{i=1}^{N}\tilde{\omega}^i\mathbf{u}^i, \tag{B.21}$$

where,

$$\tilde{\omega}^i = \frac{\omega(\mathbf{u}^i)}{\sum_{j=1}^{N}\omega(\mathbf{u}^j)}. \tag{B.22}$$

Unfortunately, the use of the estimated $\hat{Z}$ in the weights $\omega(\mathbf{u})$ causes the SNIS estimator to be biased [58].

B3 MPPI as a Special Case of the Variational Inference Update

Theorem 6. *When $p_0(\mathbf{u}) = q_{\mathbf{0}}(\mathbf{u}) := q_{\mathbf{v}=\mathbf{0}}(\mathbf{u})$ and $r(\mathbf{u}) = q_{\bar{\mathbf{v}}}(\mathbf{u})$ for some previous estimate of the optimal control sequence $\bar{\mathbf{v}}$, the equation for the weights $\omega(\mathbf{u})$ in the variational inference MPC update (8) reduces to that of MPPI.*

Proof: Let $q_{\mathbf{v}}$ be the density function for a Gaussian distribution with mean $\mathbf{v}$ and covariance Σ, that is,

$$q_{\mathbf{v}}(\mathbf{u}) = Z_q^{-1}\exp\left(-\frac{1}{2}(\mathbf{u}-\mathbf{v})^\top\Sigma^{-1}(\mathbf{u}-\mathbf{v})\right), \tag{B.23}$$

By choosing $p_0(\mathbf{u}) = q_{\mathbf{0}}(\mathbf{u})$ and $r(\mathbf{u}) = q_{\bar{\mathbf{v}}}(\mathbf{u})$, we have that

$$\frac{p_0(\mathbf{u})}{r(\mathbf{u})} = \frac{Z_q^{-1}\exp\left(-\frac{1}{2}\mathbf{u}^\top\Sigma^{-1}\mathbf{u}\right)}{Z_q^{-1}\exp\left(-\frac{1}{2}(\mathbf{u}-\mathbf{v})^\top\Sigma^{-1}(\mathbf{u}-\mathbf{v})\right)} \tag{B.24}$$

$$= \exp(-\frac{1}{2}\mathbf{u}^\top\Sigma^{-1}\mathbf{u} + \frac{1}{2}(\mathbf{u}-\mathbf{v})^\top\Sigma^{-1}(\mathbf{u}-\mathbf{v})) \tag{B.25}$$

$$= \exp(-\frac{1}{2}\mathbf{u}^\top\Sigma^{-1}\mathbf{u} + \frac{1}{2}\mathbf{u}^\top\Sigma^{-1}\mathbf{u} - \mathbf{u}^\top\Sigma^{-1}\mathbf{v} + \frac{1}{2}\mathbf{v}^\top\Sigma^{-1}\mathbf{v}) \tag{B.26}$$

$$= \exp(-\mathbf{u}^\top\Sigma^{-1}\mathbf{v} + \frac{1}{2}\mathbf{v}^\top\Sigma^{-1}\mathbf{v}). \tag{B.27}$$

Hence,

$$\omega(\mathbf{u}) = \frac{Z^{-1}\exp(-J(\mathbf{u}))p_0(\mathbf{u})}{r(\mathbf{u})} \tag{B.28}$$

$$= Z^{-1}\exp(-[J(\mathbf{u}) + \mathbf{u}^\top\Sigma^{-1}\mathbf{v}])\exp(\frac{1}{2}\mathbf{v}^\top\Sigma^{-1}\mathbf{v}) \tag{B.29}$$

$$\propto \exp(-[J(\mathbf{u}) + \mathbf{u}^\top\Sigma^{-1}\mathbf{v}]), \tag{B.30}$$

where the last line follows since terms not dependent on $\mathbf{u}$ can be canceled out between the numerator and denominator in (10). ■

APPENDIX C
NCBF TRAINING DETAILS

The definition of the avoid set is given by (2),

$$\mathcal{A} = \{x | h(x) > 0\},$$

where the avoidance heuristic $h(x)$ is used to define the avoid set. Furthermore, the definition of the policy value function $V^{h,\pi}(x_0)$ given by (19), and the ensuing training losses (24), (26) are all dependent on $h(x)$. A suitable selection of the avoidance heuristic $h(x)$ must ensure that its corresponding avoidance set $\mathcal{A}$ encompasses all system states that overlap with any obstacles. Additionally, the heuristic should improve the quality of information captured by the loss function (26), thereby making it more effective for neural network training. To this end, a reasonable choice of the avoidance heuristic utilizes coordinates in (A.2),

$$h_0(x) = w_I^2 - e_y^2, \tag{C.1}$$

where $w_I = 1.5$ m is half of the track width, and e_y is the lateral deviation of the vehicle CoM to the track centerline. The visualization of $h_0(x)$ is demonstrated by the orange curve in Fig. 19.

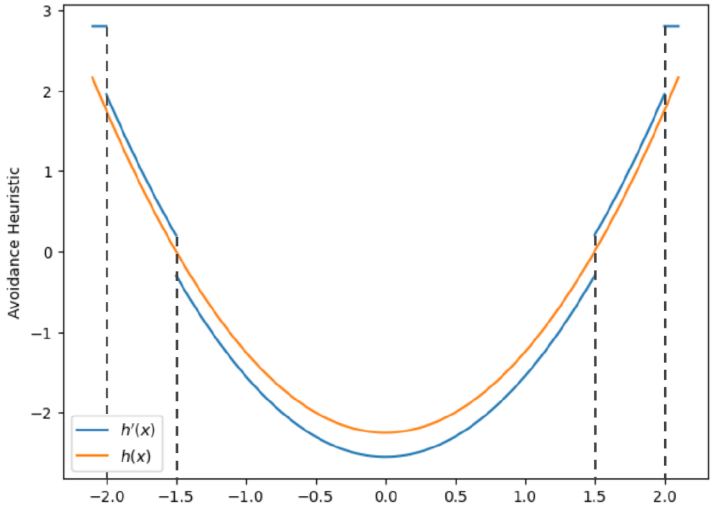


Fig. 19: Avoidance heuristic visualization. The orange curve shows the original avoidance heuristic h_0, while the blue curve demonstrates the modified avoidance heuristic h used for training the NCBF.

However, the neural network approximation $V_\theta^{h_0,\pi}(x)$ of the policy value function $V^{h_0,\pi}$ given by (19) may not accurately distinguish the unsafe states in the avoid set $\mathcal{A}_0 = \{x | h_0(x) > 0\}$ from the rest of the state space, due to the fact that trained neural networks can suffer from insufficient accuracy and precision [22, 40]. This is demonstrated by the orange line and its error margins in Fig.20. To alleviate this problem, we introduce discontinuity to $h_0(x)$ for enhanced tolerance of policy value function modelling errors while maintaining the same avoid set (2), such that,

$$\mathcal{A} = \{x | h(x) > 0\} = \{x | h_0(x) > 0\}. \tag{C.2}$$

where our choice of the avoidance heuristic is given by,

$$h(x) = \begin{cases} h_0(x) - 0.3, & \text{if } e_y < w_I, \\ h_0(x) + 0.2, & \text{if } w_I \leq e_y \leq w_O, \\ 2.8, & \text{if } w_O \leq e_y, \end{cases} \tag{C.3}$$

where w_O is the"crash width". If $w_O < e_y$, the vehicle crashes and needs to be reset. If $w_I \leq e_y \leq w_O$, the AutoRally collides with the soft drainage pipes but can still continue driving. As shown by the blue curve in Fig. 19, $h(x)$ is designed to reduce the error of the avoid set of the resulting NCBF by introducing a discontinuity around zero. This is further demonstrated in Fig. 20.

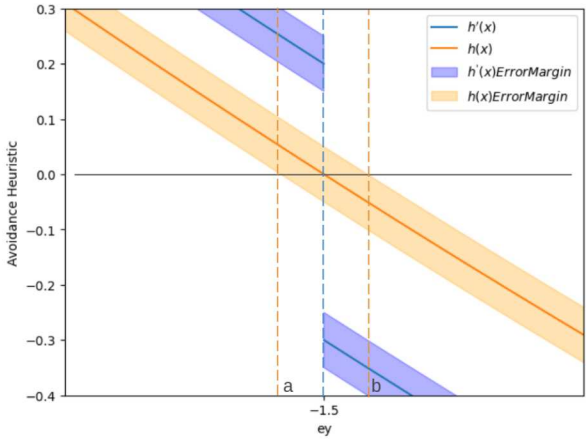


Fig. 20: Improving accuracy and precision of NCBF modeled avoid set boundary by using the modified heuristic $h(x)$. When using the original avoidance heuristic $h_0(x)$ to supervise the NCBF training process, the resulting modeled avoid set boundary can be anywhere within $e_y \in (a, b)$ due to model errors. Instead, using $h(x)$ to supervise the training process results in a modeled avoid set with an accurate boundary shown by the dashed blue line, despite model errors.

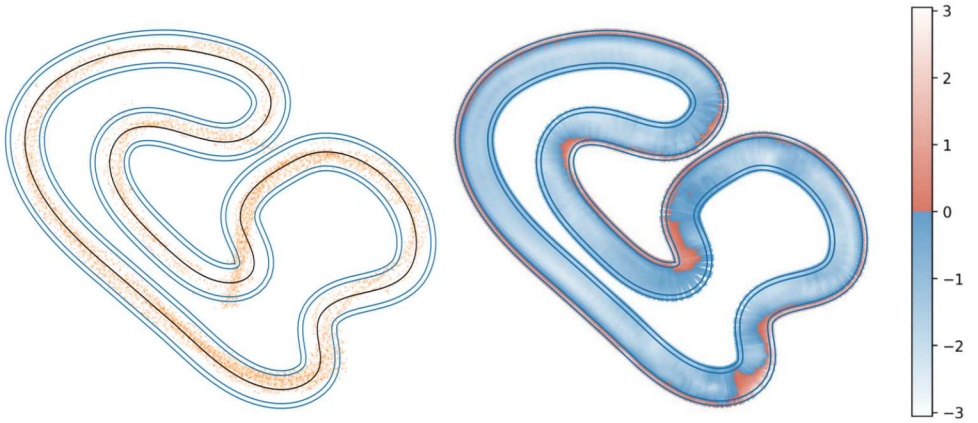


Fig. 21: NCBF Training. The left figure shows collected system states (yellow dots) in AutoRally simulations. The right figure visualizes the resulting Neural CBF. The NCBF $B(x)$ takes the 8-dimensional state x as input. The red color indicates an unsafe region where $B(x) > 0$, while the blue color indicates a safe region where $B(x) \leq 0$.

The orange dots in the left plot in Fig. 21 represent the vehicle states collected using the training data. The dots outside of the track show that the autonomous car went off the track at several turns. These data are used to train a neural DCBF, leading to a visualization as demonstrated by the plot on the right in Fig. 21,in which the NCBF $B(x)$ is mapped onto the 2D Cartesian plane by utilizing mean values from the training dataset for the remaining state dimensions. The red color shows the area where $B(x) > 0$, and as a result, the neural CBF slows down the Autorally when it approaches sharp turns to ensure safety.

From Fig. 22, we see that the learned NCBF $B(x)$ has a smaller safe set (negative level set) than the avoid set heuristic $h(x)$, as it turns positive earlier than $h(x)$ when the vehicle approaches the track boundary. When used as a safety filter, $B(x)$ can slow down the autonomous vehicle much earlier, making its negative level set control-invariant.

APPENDIX D
PROOFS FOR RESAMPLING-BASED ROLLOUTS (RBR) SUPERVISED BY A CBF

D1 Proof of Theorem 1

Proof: Condition (14b) implies that $B(x_k) \leq (\mathrm{Id} - \alpha) \circ B(x_{k-1})$, where $\circ$ denotes function composition and Id denotes the identity function [2]. Since $B(x_1) \leq (\mathrm{Id} - \alpha) \circ B(x_0)$, it follows that,

$$B(x_k) \leq (\mathrm{Id} - \alpha)^k \circ B(x_0). \tag{D.1}$$

Since $(\mathrm{Id} - \alpha)$ is a class-κ function for $a \in (0, 1)$, it follows from $B(x_0) \leq 0$ that $B(x_k) \leq 0$. Hence, the set $\mathcal{S}$ is forward invariant for system (1). ■

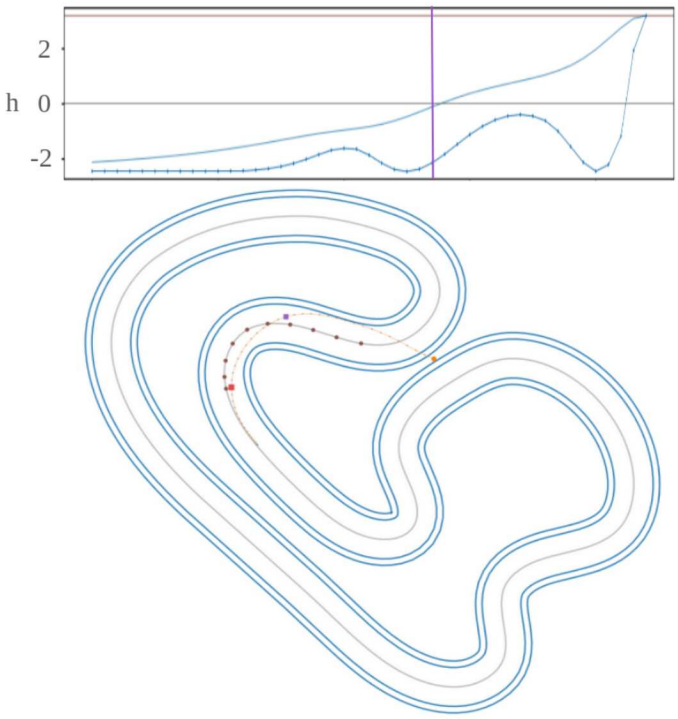


Fig. 22: A simulation example of the value change of the learned Neural CBF $B(x)$ compared to the avoid set heuristic $h(x)$. The orange trajectory produced by the standard MPPI results in a collision with the track boundary. The square purple dot shows the point where $B(x)$ shifts from negative to positive values, corresponding to the purple vertical line in the curve plot above, which shows the value change of the heuristics $B(x)$, $h(x)$ along the MPPI trajectory. The dots along the track centerline are observations collected starting at the square red dot, used to augment the training data.

D2 Proof of Theorem 3

Proof: Note that

$$\mathsf{f}(\tilde{\mathsf{b}} \mid \mathsf{a}, \mathsf{b}) = \mathbb{1}_{\mathsf{b}\in\mathsf{S}}\delta(\tilde{\mathsf{b}} - \mathsf{b}) + \mathbb{1}_{\mathsf{b}\notin\mathsf{S}}\delta(\tilde{\mathsf{b}} - \mathsf{a}). \tag{D.2}$$

Let $\tilde{\mathsf{f}}$ denote the marginal density of $\tilde{\mathsf{b}}$. Then, using the independence of a and b,

$$\tilde{\mathsf{f}}(\tilde{\mathsf{b}}) = \int \left(\int \mathsf{f}(\tilde{\mathsf{b}} \mid \mathsf{a}, \mathsf{b}) \mathsf{f}(\mathsf{a} \mid \mathsf{a} \in \mathsf{S}) \, \mathrm{d}\mathsf{a} \right) \mathsf{f}(\mathsf{b}) \, \mathrm{d}\mathsf{b} \tag{D.3}$$

Simplifying the inner integral first using properties of the Dirac delta function gives

$$\int \mathsf{f}(\tilde{\mathsf{b}} \mid \mathsf{a}, \mathsf{b}) \mathsf{f}(\mathsf{a} \mid \mathsf{a} \in \mathsf{S}) \, \mathrm{d}\mathsf{a} \tag{D.4}$$

$$= \mathbb{1}_{\mathsf{b}\in\mathsf{S}}\delta(\tilde{\mathsf{b}} - \mathsf{b}) + \mathbb{1}_{\mathsf{b}\notin\mathsf{S}} \int \delta(\tilde{\mathsf{b}} - \mathsf{a}) \mathsf{f}(\mathsf{a} \mid \mathsf{a} \in \mathsf{S}) \, \mathrm{d}\mathsf{a} \tag{D.5}$$

$$= \mathbb{1}_{\mathsf{b}\in\mathsf{S}}\delta(\tilde{\mathsf{b}} - \mathsf{b}) + \mathbb{1}_{\mathsf{b}\notin\mathsf{S}} \mathsf{f}(\tilde{\mathsf{b}} \mid \tilde{\mathsf{b}} \in \mathsf{S}). \tag{D.6}$$

Hence,

$$\tilde{\mathsf{f}}(\tilde{\mathsf{b}}) = \int \left(\mathbb{1}_{\mathsf{b}\in\mathsf{S}}\delta(\tilde{\mathsf{b}} - \mathsf{b}) + \mathbb{1}_{\mathsf{b}\notin\mathsf{S}} \mathsf{f}(\tilde{\mathsf{b}} \mid \tilde{\mathsf{b}} \in \mathsf{S}) \right) \mathsf{f}(\mathsf{b}) \, \mathrm{d}\mathsf{b} \tag{D.7}$$

$$= \mathsf{f}(\tilde{\mathsf{b}}) \mathbb{1}_{\tilde{\mathsf{b}}\in\mathsf{S}} + \mathsf{f}(\tilde{\mathsf{b}} \mid \tilde{\mathsf{b}} \in \mathsf{S}) \int \mathbb{1}_{\mathsf{b}\notin\mathsf{S}} \mathsf{f}(\mathsf{b}) \, \mathrm{d}\mathsf{b} \tag{D.8}$$

$$= \mathsf{f}(\tilde{\mathsf{b}} \mid \tilde{\mathsf{b}} \in \mathsf{S}) P(\mathsf{b} \in \mathsf{S}) + \mathsf{f}(\tilde{\mathsf{b}} \mid \tilde{\mathsf{b}} \in \mathsf{S}) P(\mathsf{b} \notin \mathsf{S}) \tag{D.9}$$

$$= \mathsf{f}(\tilde{\mathsf{b}} \mid \tilde{\mathsf{b}} \in \mathsf{S}). \tag{D.10}$$

■

D3 Proof of Corollary 1

Proof: Applying Theorem 3, x, a and $\tilde{\mathsf{b}}$ have the same distribution,

$$\mathsf{x} \overset{d}{=} \mathsf{a} \overset{d}{=} \tilde{\mathsf{b}}. \tag{D.11}$$

Hence,

$$\begin{aligned}
\mathbb{E}[\frac{1}{2}w(\mathsf{a}) + \frac{1}{2}w(\tilde{\mathsf{b}})] &= \frac{1}{2}\mathbb{E}[w(\mathsf{a})] + \frac{1}{2}\mathbb{E}[w(\tilde{\mathsf{b}})], && \text{(D.12)}\\
&= \frac{1}{2}\mathbb{E}[w(\mathsf{x})] + \frac{1}{2}\mathbb{E}[w(\mathsf{x})], && \text{(D.13)}\\
&= \mathbb{E}[w(\mathsf{x})]. && \text{(D.14)}
\end{aligned}$$

■

D4 Proof of Theorem 4

We prove the two claims separately.

Lemma 7. *The variance of the Monte Carlo estimator of the optimal control law is*

$$\mathrm{Var}[\hat{v}_k] = \frac{1}{N}\left(\frac{1}{3}2^K - \frac{1}{4}\right), \qquad k = 1, \dots, K. \tag{D.15}$$

Proof: First, note that the optimal control $\mathbf{v}^*$ is given by

$$\begin{aligned}
v_k^* &= \int_{[-1,1]^K} p(\mathbf{u} \mid o = 1) u_k \, \mathrm{d}\mathbf{u} && \text{(D.16)}\\
&= \int_{[-1,1]^K} \mathbb{1}_{\mathbf{u}\in[0,1]} u_k \, \mathrm{d}\mathbf{u} && \text{(D.17)}\\
&= \int_{[0,1]^K} u_k \, \mathrm{d}\mathbf{u} && \text{(D.18)}\\
&= \frac{1}{2}. && \text{(D.19)}
\end{aligned}$$

Using the formula for computing the variance of an importance sampled Monte Carlo estimator [58], we then have that

$$\begin{aligned}
\mathrm{Var}[\hat{v}_k] &= \frac{1}{N}\int_{[-1,1]^K} \frac{(u_k p(\mathbf{u} \mid o = 1) - v_k^* r(\mathbf{u}))^2}{r(\mathbf{u})} \, \mathrm{d}\mathbf{u} && \text{(D.20)}\\
&= \frac{1}{N}\int_{[-1,1]^K} \frac{(u_k p(\mathbf{u} \mid o = 1) - \frac{1}{2}2^{-K})^2}{2^{-K}} \, \mathrm{d}\mathbf{u} && \text{(D.21)}\\
&= \frac{1}{N}2^K\left(\int_{[0,1]^K} (u_k)^2 \, \mathrm{d}\mathbf{u} - 2^{-K}\int_{[0,1]^K} u_k \, \mathrm{d}\mathbf{u} + \frac{1}{4}(2^{-K})^2 \int_{[-1,1]^K} 1 \, \mathrm{d}\mathbf{u}\right) && \text{(D.22)}\\
&= \frac{1}{N}2^K\left(\frac{1}{3} - \frac{1}{2}2^{-K} + \frac{1}{4}2^{-K}\right) && \text{(D.23)}\\
&= \frac{1}{N}\left(\frac{1}{3}2^K - \frac{1}{4}\right). && \text{(D.24)}
\end{aligned}$$

■

Proof of second claim. We now prove the second claim. When performing resampling-based rollouts, *only* if u_k^i lies in $[-1, 0]$ for all $k = 0, \dots, K-2$ does resampling not occur, and the output $\tilde{u}_k^i \in [-1, 0]$ for $k = 0, \dots, K-2$. Otherwise, we have that $\tilde{u}_k^i \in [0, 1]$ for $k = 0, \dots, K-2$. The last control $u_{K-1}^i = \tilde{u}_{K-1}^i$ is never resampled. Hence, the probability that all N samples lie in $[-1, 0]^{K-1} \times [-1, 1]$ is equal to $2^{-N(K-1)}$.

For all timesteps except the last, we write the joint probability density function (over all N samples) as

$$p(\tilde{u}_k^1, \dots, \tilde{u}_k^N) = \begin{cases} 2^{-N}, & \text{if } \tilde{u}_k^i \in [-1, 0] \text{ for all } i = 1, \dots, N, \\ 1 - 2^{-N}, & \text{if } \tilde{u}_k^i \in [0, 1] \text{ for all } i = 1, \dots, N, \\ 0, & \text{otherwise,} \end{cases} \qquad \text{for } k = 0, \dots, K-2, \tag{D.25}$$

and

$$p(\tilde{u}^i_{K-1}) = 1, \qquad p(\tilde{u}^1_{K-1}, \dots, \tilde{u}^N_{K-1}) = \prod_{i=1}^{N} p(\tilde{u}^i_{K-1}) = 1. \tag{D.26}$$

For convenience, let $t \coloneqq 2^{-N}$ such that $P(\tilde{u}^{1:N}_k \in [0,1]) = 1 - t$ for $k < K-1$, and

$$P(\tilde{u}^{1:N}_{0:K-2} \in [0,1]) = \prod_{k=0}^{K-2} P(\tilde{u}^{1:N}_k \in [0,1]) = (1-t)^{K-1}. \tag{D.27}$$

Next, consider the Monte Carlo estimator $\hat{v}_k$ using the resampled controls $\tilde{u}^i_k$, defined by

$$\hat{v}_k = \frac{1}{N} \sum_{i=1}^{N} \frac{\mathbb{1}_{\tilde{u}^i_{0:K-1} \in [0,1]}}{\frac{1}{2}(1-t)^{K-1}} \tilde{u}^i_k. \tag{D.28}$$

We will compute the expectation of $\hat{v}_k$ using the law of total expectation. To this end, we have that

$$\mathbb{E}_{\tilde{u}^{1:N}_{0:K-1}}[\hat{v}_k] = \frac{1}{N} \mathbb{E}_{\tilde{u}^{1:N}_{0:K-1}} \left[\sum_{i=1}^{N} \frac{\mathbb{1}_{\tilde{u}^i_{0:K-1} \in [0,1]}}{\frac{1}{2}(1-t)^{K-1}} \tilde{u}^i_k \right] \tag{D.29}$$

$$= \frac{1}{N} \Big(P(\tilde{u}^{1:N}_{0:K-2} \geq 0) \mathbb{E}_{\tilde{u}^{1:N}_{0:K-1}} \left[\sum_{i=1}^{N} \frac{\mathbb{1}_{\tilde{u}^i_{0:K-1} \in [0,1]}}{\frac{1}{2}(1-t)^{K-1}} \tilde{u}^i_k \;\middle|\; \tilde{u}^{1:N}_{0:K-2} \geq 0 \right] \tag{D.30}$$

$$+ P(\tilde{u}^{1:N}_{0:K-2} < 0) \mathbb{E}_{\tilde{u}^{1:N}_{0:K-1}} \left[\sum_{i=1}^{N} \frac{\mathbb{1}_{\tilde{u}^i_{0:K-1} \in [0,1]}}{\frac{1}{2}(1-t)^{K-1}} \tilde{u}^i_k \;\middle|\; \tilde{u}^{1:N}_{0:K-2} < 0 \right] \Big)$$

$$= \frac{1}{N} \left((1-t)^{K-1} \mathbb{E}_{\tilde{u}^{1:N}_{0:K-1}} \left[\sum_{i=1}^{N} \frac{\mathbb{1}_{\tilde{u}^i_{0:K-1} \in [0,1]}}{\frac{1}{2}(1-t)^{K-1}} \tilde{u}^i_k \;\middle|\; \tilde{u}^{1:N}_{0:K-2} \geq 0 \right] \right) \tag{D.31}$$

$$= \frac{2}{N} \left(\mathbb{E}_{\tilde{u}^{1:N}_{0:K-1}} \left[\sum_{i=1}^{N} \mathbb{1}_{\tilde{u}^i_{0:K-1} \in [0,1]} \tilde{u}^i_k \;\middle|\; \tilde{u}^{1:N}_{0:K-2} \geq 0 \right] \right). \tag{D.32}$$

We now split into two cases. When $k = K-1$,

$$\mathbb{E}_{\tilde{u}^{1:N}_{0:K-1}}[\hat{v}_{K-1}] = \frac{2}{N} \left(\mathbb{E}_{\tilde{u}^{1:N}_{K-1}} \left[\mathbb{E}_{\tilde{u}^{1:N}_{0:K-2}} \left[\sum_{i=1}^{N} \mathbb{1}_{\tilde{u}^i_{0:K-1} \in [0,1]} \tilde{u}^i_{K-1} \;\middle|\; \tilde{u}^{1:N}_{0:K-2} \geq 0 \right] \right] \right) \tag{D.33}$$

$$= \frac{2}{N} \left(\mathbb{E}_{\tilde{u}^{1:N}_{K-1}} \left[\sum_{i=1}^{N} \mathbb{1}_{\tilde{u}^i_{K-1} \in [0,1]} \tilde{u}^i_{K-1} \right] \right) \tag{D.34}$$

$$= \frac{2}{N} \left(\sum_{i=1}^{N} \mathbb{E}_{\tilde{u}^i_{K-1}} \left[\mathbb{1}_{\tilde{u}^i_{K-1} \in [0,1]} \tilde{u}^i_{K-1} \right] \right) \tag{D.35}$$

$$= \frac{2}{N} \left(\sum_{i=1}^{N} \frac{1}{4} \right) \tag{D.36}$$

$$= \frac{1}{2}. \tag{D.37}$$

Similarly, when $k < K-1$,

$$\mathbb{E}_{\tilde{u}^{1:N}_{0:K-1}}[\hat{v}_k] = \frac{2}{N} \left(\mathbb{E}_{\tilde{u}^{1:N}_{K-1}} \left[\mathbb{E}_{\tilde{u}^{1:N}_{0:K-2}} \left[\sum_{i=1}^{N} \mathbb{1}_{\tilde{u}^i_{K-1} \in [0,1]} \mathbb{1}_{\tilde{u}^i_{0:K-2} \in [0,1]} \tilde{u}^i_k \;\middle|\; \tilde{u}^{1:N}_{0:K-2} \geq 0 \right] \right] \right) \tag{D.38}$$

$$= \frac{1}{N} \left(\mathbb{E}_{\tilde{u}^{1:N}_{0:K-2}} \left[\sum_{i=1}^{N} \tilde{u}^i_k \;\middle|\; \tilde{u}^{1:N}_{0:K-2} \geq 0 \right] \right) \tag{D.39}$$

$$= \frac{1}{2}. \tag{D.40}$$

Thus, $\hat{v}_k$ is an unbiased estimator.

Next, we look at the variance. Summarizing the above computation, we have that

$$\mathbb{E}_{\tilde{u}_{0:K-1}^{1:N}}[\hat{v}_k \mid \tilde{u}_{0:K-2}^{1:N} \geq 0] = \frac{1}{2(1-t)^{K-1}}, \qquad \mathbb{E}_{\tilde{u}_{0:K-1}^{1:N}}[\hat{v}_k \mid \tilde{u}_{0:K-2}^{1:N} < 0] = 0. \tag{D.41}$$

Let $c \coloneqq \mathbb{E}_{\tilde{u}_{0:K-1}^{1:N}}[\hat{v}_k \mid \tilde{u}_{0:K-2}^{1:N} \geq 0] = \frac{1}{2(1-t)^{K-1}}$ for convenience. Computing the conditional variances, for $\tilde{u}_{0:K-2}^{1:N} < 0$, we have that

$$\mathrm{Var}_{\tilde{u}_{0:K-1}^{1:N}}[\hat{v}_k \mid \tilde{u}_{0:K-2}^{1:N} < 0] = 0. \tag{D.42}$$

For $\tilde{u}_{0:K-2}^{1:N} \geq 0$ we have,

$$\mathrm{Var}_{\tilde{u}_{0:K-1}^{1:N}}[\hat{v}_k \mid \tilde{u}_{0:K-2}^{1:N} \geq 0] = \mathbb{E}_{\tilde{u}_{0:K-1}^{1:N}}[\hat{v}_k^2 \mid \tilde{u}_{0:K-2}^{1:N} \geq 0] - c^2 \tag{D.43}$$

$$= \mathbb{E}_{\tilde{u}_{0:K-1}^{1:N}}\left[\left(\frac{1}{N}\sum_{i=1}^{N}\frac{\mathbb{1}_{\tilde{u}_{0:K-1}^{i}\in[0,1]}}{\frac{1}{2}(1-t)^{K-1}}\tilde{u}_k^i\right)^2 \mid \tilde{u}_{0:K-2}^{1:N} \geq 0\right] - c^2 \tag{D.44}$$

$$= \left(\frac{1}{N}\frac{1}{\frac{1}{2}(1-t)^{K-1}}\right)^2 \underbrace{\mathbb{E}_{\tilde{u}_{0:K-1}^{1:N}}\left[\left(\sum_{i=1}^{N}\mathbb{1}_{\tilde{u}_{0:K-1}^{i}\in[0,1]}\tilde{u}_k^i\right)^2 \,\middle|\, \tilde{u}_{0:K-2}^{1:N} \geq 0\right]}_{\coloneqq \textcircled{\star}} - c^2 \tag{D.45}$$

Expanding $\textcircled{\star}$, yields

$$\textcircled{\star} = \mathbb{E}_{\tilde{u}_{0:K-1}^{1:N}}\left[\sum_{i=1}^{N}\mathbb{1}_{\tilde{u}_{0:K-1}^{i}\in[0,1]}(\tilde{u}_k^i)^2 + \sum_{i=1}^{N}\sum_{j=1,j\neq i}^{N}\mathbb{1}_{\tilde{u}_{0:K-1}^{i}\in[0,1]}\mathbb{1}_{\tilde{u}_{0:K-1}^{j}\in[0,1]}\tilde{u}_k^i\tilde{u}_k^j \,\middle|\, \tilde{u}_{0:K-2}^{1:N} \geq 0\right]. \tag{D.46}$$

We now split into two cases.

Case 1: $k = K-1$. Looking at the first term of $\textcircled{\star}$ in (D.46),

$$\mathbb{E}_{\tilde{u}_{0:K-1}^{1:N}}\left[\sum_{i=1}^{N}\mathbb{1}_{\tilde{u}_{0:K-1}^{i}\in[0,1]}(\tilde{u}_{K-1}^i)^2 \mid \tilde{u}_{0:K-2}^{1:N} \geq 0\right] = \mathbb{E}_{\tilde{u}_{0:K-1}^{1:N}}\left[\sum_{i=1}^{N}\mathbb{E}_{\tilde{u}_{K-1}^{i}}[\mathbb{1}_{\tilde{u}_{K-1}^{i}\in[0,1]}(\tilde{u}_{K-1}^i)^2] \mid \tilde{u}_{0:K-2}^{1:N} \geq 0\right] \tag{D.47}$$

$$= \sum_{i=1}^{N}\mathbb{E}_{\tilde{u}_{K-1}^{i}}[\mathbb{1}_{\tilde{u}_{K-1}^{i}\in[0,1]}(\tilde{u}_{K-1}^i)^2] \tag{D.48}$$

$$= \frac{N}{6}. \tag{D.49}$$

For the second term in (D.46), we have

$$\mathbb{E}_{\tilde{u}_{0:K-1}^{1:N}}\left[\sum_{i=1}^{N}\sum_{j=1,j\neq i}^{N}\mathbb{1}_{\tilde{u}_{0:K-1}^{i}\in[0,1]}\mathbb{1}_{\tilde{u}_{0:K-1}^{j}\in[0,1]}\tilde{u}_{K-1}^i\tilde{u}_{K-1}^j \mid \tilde{u}_{0:K-2}^{1:N} \geq 0\right] \tag{D.50}$$

$$= \sum_{i=1}^{N}\sum_{j=1,j\neq i}^{N}\mathbb{E}_{\tilde{u}_{K-1}^{i,j}}\left[\mathbb{1}_{\tilde{u}_{K-1}^{i}\in[0,1]}\mathbb{1}_{\tilde{u}_{K-1}^{j}\in[0,1]}\tilde{u}_{K-1}^i\tilde{u}_{K-1}^j\right] \tag{D.51}$$

$$= \sum_{i=1}^{N}\sum_{j=1,j\neq i}^{N}\mathbb{E}_{\tilde{u}_{K-1}^{i,j}}\left[\mathbb{1}_{\tilde{u}_{K-1}^{i}\in[0,1]}\tilde{u}_{K-1}^i\right]^2 \tag{D.52}$$

$$= \sum_{i=1}^{N}\sum_{j=1,j\neq i}^{N}\frac{1}{4} \tag{D.53}$$

$$= \frac{N(N-1)}{4}. \tag{D.54}$$

Substituting the two terms (D.49) and (D.54) into (D.45), yields

$$\mathrm{Var}_{\tilde{u}_{0:K-1}^{1:N}}[\hat{v}_{K-1} \mid \tilde{u}_{0:K-2}^{1:N} \geq 0] = \left(\frac{1}{N}\frac{1}{\frac{1}{2}(1-t)^{K-1}}\right)^2 \left(\frac{N}{6} + \frac{N(N-1)}{4}\right) - c^2 \tag{D.55}$$

$$\leq \left(\frac{1}{N}\frac{1}{\frac{1}{2}(1-t)^{K-1}}\right)^2 \frac{N^2}{4} - c^2 \tag{D.56}$$

$$= \left(\frac{1}{(1-t)^2}\right)^{K-1} - \frac{1}{4}\left(\frac{1}{(1-t)^2}\right)^{K-1}, \tag{D.57}$$

$$= \frac{3}{4}\left(\frac{1}{(1-t)^2}\right)^{K-1}. \tag{D.58}$$

Case 2: $k < K-1$. Looking at the first term of $\textcircled{\star}$ in (D.46),

$$\mathbb{E}_{\tilde{u}_{0:K-1}^{1:N}}\left[\sum_{i=1}^{N} \mathbb{1}_{\tilde{u}_{0:K-1}^{i} \in [0,1]} (\tilde{u}_k^i)^2 \mid \tilde{u}_{0:K-2}^{1:N} \geq 0\right] = \mathbb{E}_{\tilde{u}_{0:K-1}^{1:N}}\left[\sum_{i=1}^{N} \mathbb{E}_{\tilde{u}_{K-1}^{i}}[\mathbb{1}_{\tilde{u}_{K-1}^{i} \in [0,1]}](\tilde{u}_k^i)^2 \mid \tilde{u}_{0:K-2}^{1:N} \geq 0\right] \tag{D.59}$$

$$= \frac{1}{2}\mathbb{E}_{\tilde{u}_{0:K-1}^{1:N}}\left[\sum_{i=1}^{N} (\tilde{u}_k^i)^2 \mid \tilde{u}_{0:K-2}^{1:N} \geq 0\right]. \tag{D.60}$$

From Theorem 3, the marginal of any resampled controls will all be uniform over $[0,1]$. Hence,

$$\frac{1}{2}\mathbb{E}_{\tilde{u}_{0:K-1}^{1:N}}\left[\sum_{i=1}^{N} (\tilde{u}_k^i)^2 \mid \tilde{u}_{0:K-2}^{1:N} \geq 0\right] = \frac{1}{2}\sum_{i=1}^{N} \mathbb{E}_{\tilde{u}_k^i}[(\tilde{u}_k^i)^2 \mid \tilde{u}_k^i \geq 0] = \frac{N}{6}. \tag{D.61}$$

For the second term in (D.46),

$$\mathbb{E}_{\tilde{u}_{0:K-1}^{1:N}}\left[\sum_{i=1}^{N}\sum_{j=1,j\neq i}^{N} \mathbb{1}_{\tilde{u}_{0:K-1}^{i} \in [0,1]} \mathbb{1}_{\tilde{u}_{0:K-1}^{j} \in [0,1]} \tilde{u}_k^i \tilde{u}_k^j \mid \tilde{u}_{0:K-2}^{1:N} \geq 0\right] \tag{D.62}$$

$$= \mathbb{E}_{\tilde{u}_{0:K-2}^{1:N}}\left[\sum_{i=1}^{N}\sum_{j=1,j\neq i}^{N} \mathbb{E}_{\tilde{u}_{K-1}^{1:N}}\left[\mathbb{1}_{\tilde{u}_{K-1}^{i} \in [0,1]} \mathbb{1}_{\tilde{u}_{K-1}^{j} \in [0,1]}\right] \tilde{u}_k^i \tilde{u}_k^j \mid \tilde{u}_{0:K-2}^{1:N} \geq 0\right] \tag{D.63}$$

$$= \frac{1}{4}\mathbb{E}_{\tilde{u}_{0:K-2}^{1:N}}\left[\sum_{i=1}^{N}\sum_{j=1,j\neq i}^{N} \tilde{u}_k^i \tilde{u}_k^j \mid \tilde{u}_{0:K-2}^{1:N} \geq 0\right]. \tag{D.64}$$

To make this computation easier, let $\Xi_{ij} = 1$ denote the event that $\tilde{u}_k^i$ and $\tilde{u}_k^j$ for $i \neq j$ were resampled from the same control, i.e., $\tilde{u}_k^i = \tilde{u}_k^j$, and let $\beta = \mathbb{P}(\Xi_{ij} = 1)$. When $\Xi_{ij} = 1$, $\tilde{u}_k^i$ and $\tilde{u}_k^j$ are the same random variable, and therefore it follows that

$$\mathbb{E}_{\tilde{u}_{0:K-2}^{1:N}}\left[\tilde{u}_k^i \tilde{u}_k^j \mid \Xi_{ij} = 1 \mid \tilde{u}_{0:K-2}^{1:N} \geq 0, \Xi_{ij} = 1\right] = \mathbb{E}_{\tilde{u}_k^i}\left[(\tilde{u}_k^i)^2 \mid \tilde{u}_k^i \geq 0\right] = \frac{1}{3}. \tag{D.65}$$

Otherwise, when $\Xi_{ij} = 0$, $\tilde{u}_k^i$ and $\tilde{u}_k^j$ are independent. Hence,

$$\mathbb{E}_{\tilde{u}_{0:K-2}^{1:N}}\left[\tilde{u}_k^i \tilde{u}_k^j \mid \Xi_{ij} = 1 \mid \tilde{u}_{0:K-2}^{1:N} \geq 0, \Xi_{ij} = 0\right] = \mathbb{E}_{\tilde{u}_k^i}\left[(\tilde{u}_k^i) \mid \tilde{u}_k^i \geq 0\right]^2 = \frac{1}{4}. \tag{D.66}$$

Hence, the expectation becomes

$$\text{(D.64)} = \frac{1}{4}\sum_{i=1}^{N}\sum_{j=1,j\neq i}^{N} \beta\frac{1}{3} + (1-\beta)\frac{1}{4} \tag{D.67}$$

$$= \frac{1}{4}N(N-1)\left(\beta\frac{1}{3} + (1-\beta)\frac{1}{4}\right) \tag{D.68}$$

$$\leq \frac{1}{12}N(N-1). \tag{D.69}$$

Combining the two terms (D.61) and (D.69), we thus have that

$$\mathrm{Var}_{\tilde{u}_{0:K-1}^{1:N}}[\hat{v}_k \mid \tilde{u}_{0:K-2}^{1:N} \geq 0] \leq \left(\frac{1}{N}\frac{1}{\frac{1}{2}(1-t)^{K-1}}\right)^2 \left(\frac{N}{6} + \frac{N(N-1)}{12}\right) - c^2 \tag{D.70}$$

$$\leq \left(\frac{1}{N}\frac{1}{\frac{1}{2}(1-t)^{K-1}}\right)^2 \frac{N^2}{6} - c^2 \tag{D.71}$$

$$= \frac{2}{3}\left(\frac{1}{(1-t)^2}\right)^{K-1} - \frac{1}{4}\left(\frac{1}{(1-t)^2}\right)^{K-1} \tag{D.72}$$

$$= \frac{5}{12}\left(\frac{1}{(1-t)^2}\right)^{K-1}. \tag{D.73}$$

Hence, taking both cases into account, we have that

$$\mathrm{Var}_{\tilde{u}_{0:K-1}^{1:N}}[\hat{v}_k \mid \tilde{u}_{0:K-2}^{1:N} \geq 0] \leq \frac{3}{4}\left(\frac{1}{(1-t)^2}\right)^{K-1}. \tag{D.74}$$

Finally, using the law of total variance, we have that

$$\mathrm{Var}[\hat{v}_k] = \underbrace{\mathbb{E}_{\text{sign of } \tilde{u}_{0:K-2}^{1:N}}[\mathrm{Var}[\hat{v}_k \mid \tilde{u}_{0:K-2}^{1:N}]]}_{①} + \underbrace{\mathrm{Var}_{\text{sign of } \tilde{u}_{0:K-2}^{1:N}}[\mathbb{E}[\hat{v}_k \mid \tilde{u}_{0:K-2}^{1:N}]]}_{②}. \tag{D.75}$$

The first term gives

$$① = \mathbb{P}(\tilde{u}_{0:K-2}^{1:N} \geq 0)\,\mathrm{Var}[\hat{v}_k \mid \tilde{u}_{0:K-2}^{1:N} \geq 0] + \mathbb{P}(\tilde{u}_{0:K-2}^{1:N} < 0)\,\mathrm{Var}[\hat{v}_k \mid \tilde{u}_{0:K-2}^{1:N} < 0] \tag{D.76}$$

$$\leq (1-t)^{K-1}\frac{3}{4}\left(\frac{1}{(1-t)^2}\right)^{K-1} + 0 \tag{D.77}$$

$$= \frac{3}{4}\left(\frac{1}{1-t}\right)^{K-1}. \tag{D.78}$$

The second term gives

$$② = \mathbb{P}(\tilde{u}_{0:K-2}^{1:N} \geq 0)(\mathbb{E}[\hat{v}_k \mid \tilde{u}_{0:K-2}^{1:N} \geq 0] - \mathbb{E}[\hat{v}_k])^2 + \mathbb{P}(\tilde{u}_{0:K-2}^{1:N} < 0)(\mathbb{E}[\hat{v}_k \mid \tilde{u}_{0:K-2}^{1:N} < 0] - \mathbb{E}[\hat{v}_k])^2 \tag{D.79}$$

$$= (1-t)^{K-1}\left(\frac{1}{2}\frac{1}{(1-t)^{K-1}} - \frac{1}{2}\right)^2 + (1-(1-t)^{K-1})\frac{1}{4} \tag{D.80}$$

$$= \frac{1}{2}(1-t)^{K-1} - 1 + \frac{1}{2}\frac{1}{(1-t)^{K-1}} + \frac{1}{4} - \frac{1}{4}(1-t)^{K-1} \tag{D.81}$$

$$= \frac{1}{4}(1-t)^{K-1} - \frac{3}{4} + \frac{1}{2}\frac{1}{(1-t)^{K-1}}. \tag{D.82}$$

Hence, combining both terms gives us

$$\mathrm{Var}[\hat{v}_k] \leq \frac{3}{4}\left(\frac{1}{1-t}\right)^{K-1} + \frac{1}{4}(1-t)^{K-1} - \frac{3}{4} + \frac{1}{2}\left(\frac{1}{1-t}\right)^{K-1} \tag{D.83}$$

$$= \frac{5}{4}\left(\frac{1}{1-t}\right)^{K-1} + \frac{1}{4}(1-t)^{K-1} - \frac{3}{4} \tag{D.84}$$

$$= O\left(\left(\frac{1}{1-2^{-N}}\right)^K + (1-2^{-N})^K\right). \tag{D.85}$$

D5 Proof of Theorem 5

Proof: We will prove the equivalent statement that

$$\|\tilde{\mathbf{w}}\|_2^2 - \|\tilde{\mathbf{w}}'\|_2^2 \geq 0. \tag{D.86}$$

To begin, we expand $\tilde{\mathbf{w}}'$ to obtain

$$\|\tilde{\mathbf{w}}'\|_2^2 = \frac{\|\mathbf{w}\|_2^2 + \|\mathbf{c}\|_2^2}{\|\mathbf{w}'\|_1^2}. \tag{D.87}$$

Then,

$$\|\tilde{\mathbf{w}}\|_2^2 - \|\tilde{\mathbf{w}}'\|_2^2 = \frac{\|\mathbf{w}\|_2^2}{\|\mathbf{w}\|_1^2} + \frac{-\|\mathbf{w}\|_2^2 - \|\mathbf{c}\|_2^2}{\|\mathbf{w}'\|_1^2} \tag{D.88}$$

$$= \frac{1}{\|\mathbf{w}'\|_1^2}\left(\|\mathbf{w}'\|_1^2 \frac{\|\mathbf{w}\|_2^2}{\|\mathbf{w}\|_1^2} - \|\mathbf{w}\|_2^2 - \|\mathbf{c}\|_2^2\right). \tag{D.89}$$

Simplifying the first term, yields,

$$\|\mathbf{w}'\|_1^2 \frac{\|\mathbf{w}\|_2^2}{\|\mathbf{w}\|_1^2} = \left(\|\mathbf{w}\|_1^2 + 2\|\mathbf{w}\|_1\|\mathbf{c}\|_1 + \|\mathbf{c}\|_1^2\right)\frac{\|\mathbf{w}\|_2^2}{\|\mathbf{w}\|_1^2} \tag{D.90}$$

$$= \|\mathbf{w}\|_2^2 + 2\frac{\|\mathbf{w}\|_2^2\|\mathbf{c}\|_1}{\|\mathbf{w}\|_1} + \frac{\|\mathbf{w}\|_2^2\|\mathbf{c}\|_1^2}{\|\mathbf{w}\|_1^2}. \tag{D.91}$$

Substituting (D.91) back into the parenthesis (D.89) gives us that

$$\|\mathbf{w}'\|_1^2 \frac{\|\mathbf{w}\|_2^2}{\|\mathbf{w}\|_1^2} - \|\mathbf{w}\|_2^2 - \|\mathbf{c}\|_2^2 = 2\frac{\|\mathbf{w}\|_2^2\|\mathbf{c}\|_1}{\|\mathbf{w}\|_1} + \frac{\|\mathbf{w}\|_2^2\|\mathbf{c}\|_1^2}{\|\mathbf{w}\|_1^2} - \|\mathbf{c}\|_2^2 \tag{D.92}$$

$$\geq 2\frac{\|\mathbf{w}\|_2^2\|\mathbf{c}\|_1}{\|\mathbf{w}\|_1} + \frac{\|\mathbf{w}\|_2^2\|\mathbf{c}\|_1^2}{\|\mathbf{w}\|_1^2} - \|\mathbf{c}\|_1^2 \tag{D.93}$$

$$= \|\mathbf{c}\|_1\left(2\frac{\|\mathbf{w}\|_2^2}{\|\mathbf{w}\|_1} + \frac{\|\mathbf{w}\|_2^2\|\mathbf{c}\|_1}{\|\mathbf{w}\|_1^2} - \|\mathbf{c}\|_1\right). \tag{D.94}$$

Simplifying the parenthesis in (D.94), we obtain

$$2\frac{\|\mathbf{w}\|_2^2}{\|\mathbf{w}\|_1} + \frac{\|\mathbf{w}\|_2^2\|\mathbf{c}\|_1}{\|\mathbf{w}\|_1^2} - \|\mathbf{c}\|_1 = 2\frac{\|\mathbf{w}\|_2^2}{\|\mathbf{w}\|_1} + \|\mathbf{c}\|_1\left(\frac{\|\mathbf{w}\|_2^2}{\|\mathbf{w}\|_1^2} - 1\right) \tag{D.95}$$

$$\geq 2\frac{\|\mathbf{w}\|_2^2}{\|\mathbf{w}\|_1} + \|\mathbf{c}\|_1\left(\frac{1}{N} - 1\right). \tag{D.96}$$

Using our assumption that $\mathbf{c}$ is not "drastically larger" than $\mathbf{w}$ in (38) then gives us the desired result. ■

APPENDIX E
DETAILS ON VIMPC COST DESIGNS

In both simulations and experiments, MPPI variants using DCBF to ensure safety utilized a state-dependent cost function:

$$q(x_k^m) = (x_k^m - x_g)^\intercal Q(x_k^m - x_g), \tag{E.1}$$

whereas the standard MPPI employed the cost,

$$q(x_k^m) = (x_k^m - x_g)^\intercal Q(x_k^m - x_g) + \mathbf{1}(x_k^m), \tag{E.2}$$

where $Q = \text{diag}(q_{v_x}, q_{v_y}, q_{\dot{\psi}}, q_{\omega_F}, q_{\omega_R}, q_{e_\psi}, q_{e_y}, q_s)$ represents the cost weights, and $x_g = \text{diag}(v_g, 0, \ldots, 0)$ specifies the target velocity. The collision cost function is defined as:

$$\mathbf{1}(x_k^m) := \begin{cases} 0, & \text{if } x_k^m \text{ is within the track,} \\ C_{\text{obs}}, & \text{otherwise.} \end{cases} \tag{E.3}$$

Unlike standard MPPI, Shield-MPPI and NS-MPPI incorporate a DCBF constraint violation penalty but do not include an explicit collision cost.

APPENDIX F
CONSTRAINT SATISFACTION ON THE VARIATIONAL DISTRIBUTION

One way of guaranteeing that the variational distribution $q(\mathbf{u})$ satisfies the constraints $x_k \in \bar{\mathcal{X}} := \mathcal{X} \backslash \mathcal{A}$ is by making the strong (but unrealistic) assumption that the set of controls that satisfy the constraints is a convex set. Specifically, for generic state constraints $x \in \bar{\mathcal{X}}$, define the set of safe control trajectories $\mathcal{U}^K_{\text{safe}} \subseteq \mathcal{U}^K$ as

$$\mathcal{U}^K_{\text{safe}}(x_0) = \left\{ \mathbf{u} \in \mathcal{U}^K \mid x_k \in \bar{\mathcal{X}},\ k = 0, 1, \ldots, K \right\}. \tag{F.1}$$

We then have the following lemma.

Lemma 8. *Assume $\mathcal{U}^K_{\text{safe}}(x_0)$ is convex for all states $x_0 \in \bar{\mathcal{X}}$, and suppose that $p(\mathbf{u} \mid o = 1)$ has zero density outside of $\mathcal{U}^K_{\text{safe}}(x_0)$. Then, the mean $\mathbf{v}^*$ (8) of the variational distribution $q_{\mathbf{v}}$ (and the state trajectory resulting from following $\mathbf{v}^*$) will also satisfy the constraints, i.e.,*

$$\mathbf{v}^* \in \mathcal{U}^K_{\text{safe}}(x_0). \tag{F.2}$$

Proof: By definition (6), $\mathbf{v}^*$ is the mean of the optimal distribution $p(\mathbf{u} \mid o = 1)$. Since $p(\mathbf{u} \mid o = 1)$ has zero density outside $\mathcal{U}^K_{\text{safe}}(x_0)$, its support is contained within $\mathcal{U}^K_{\text{safe}}(x_0)$. Since $\mathcal{U}^K_{\text{safe}}(x_0)$ is convex, the integral of $\mathbf{u}$ over $p(\mathbf{u} \mid o = 1)$ will also be contained within $\mathcal{U}^K_{\text{safe}}(x_0)$. Hence, $\mathbf{v}^* \in \mathcal{U}^K_{\text{safe}}(x_0)$. ■

APPENDIX G
RBR PERFORMANCE USING A VALID DCBF

Valid DCBF on DubinsCar. We evaluate a valid DCBF on DubinsCar with $x = [p_x, p_y, \theta]$, $u = [\dot{\theta}]$ (Fig. 23). We also add a high-cost region and a constraint that $|\theta| \leq \pi/2$.

- At $K = 1$, RBR has no effect (no resampling). With $\mathcal{U} = \mathbb{R}$, a valid DCBF allows safe sampling with just $N = 50$, and both S-MPPI and NS-MPPI avoid crashes.
- As estimator variance grows exponentially with K, S-MPPI crashes at larger K. RBR (NS-MPPI) mitigates this, maintaining safety across all horizons.
- Higher K is needed to avoid the high cost region. The cost slowly increases with $K \geq 10$ from estimator variance.

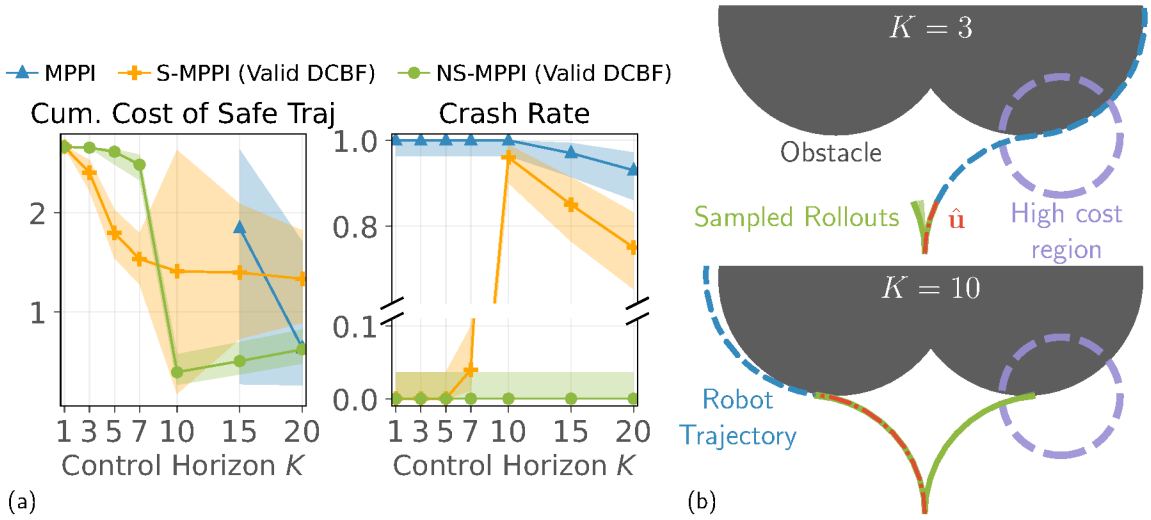


Fig. 23: (a) Cumulative cost of safe trajectories using valid DCBF on DubinsCar with $N = 50$. MPPI is unsafe for $K < 15$ and omitted. (b) Higher K is needed to avoid the high-cost region.