

DualGuard MPPI: Safe and Performant Optimal Control by Combining Sampling-Based MPC and Hamilton-Jacobi Reachability

Javier Borquez ^{ID}, Graduate Student Member, IEEE, Luke Raus, Yusuf Umut Ciftci, Graduate Student Member, IEEE, and Somil Bansal ^{ID}

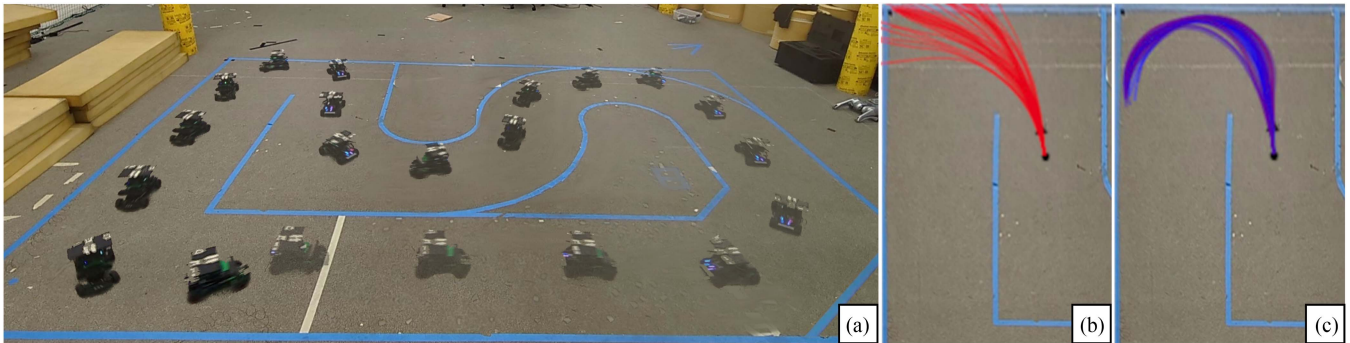


Fig. 1. We propose DualGuard MPPI - a framework to solve optimal control problems subjected to hard safety constraints. Our method integrates safety filtering during the sampling process in MPPI to generate safe rollouts, ensuring safe executions while improving sample efficiency. An output least restrictive filter is used to ensure safe executions on the system, despite potential multimodality in the sampling process. (a) We apply the proposed framework to an RC car experiment where the vehicle completes laps without leaving the track (breaching safety), while trying to stay centered in the lane and going as fast as possible. (b) Sampled trajectories in classical MPPI generate only high-cost unsafe executions near a tight corner, which results in breaching the boundary of the track. (c) DualGuard MPPI safe rollouts generate only collision-free executions with mild costs depending only on performance criteria, resulting in safe and performant behavior. Details on this experiment are provided in Section VI.

Abstract—Designing controllers that are both safe and performant is inherently challenging. This co-optimization can be formulated as a constrained optimal control problem, where the cost function represents the performance criterion and safety is specified as a constraint. While sampling-based methods, such as Model Predictive Path Integral (MPPI) control, have shown great promise in tackling complex optimal control problems, they often struggle to enforce safety constraints. To address this limitation, we propose DualGuard-MPPI, a novel framework for solving safety-constrained optimal control problems. Our approach integrates Hamilton-Jacobi reachability analysis within the MPPI sampling process to ensure that all generated samples are provably safe for the system. On the one hand, this integration allows DualGuard-MPPI to enforce strict safety constraints; at the same time, it facilitates a more effective exploration of the environment with the same number of samples, reducing the effective sampling variance

and leading to better performance optimization. Through several simulations and hardware experiments, we demonstrate that the proposed approach achieves much higher performance compared to existing MPPI methods, without compromising safety.

Index Terms—Robot safety, collision avoidance, motion and path planning.

I. INTRODUCTION

CO-OPTIMIZING safety and performance is a critical challenge in the design of controllers for autonomous systems, especially in safety-critical domains such as autonomous vehicles, UAVs, and robotics. In such settings, controllers must ensure that performance objectives are met while guaranteeing that safety constraints are respected. This can be framed as a constrained optimal control problem, where the goal is to satisfy safety constraints throughout the trajectory while optimizing performance objectives.

Several methods have been proposed to address this co-optimization challenge. Dynamic programming (DP) offers a rigorous solution to constrained optimization [1], [2]. Still, it is computationally intractable for many real-time applications due to the “curse of dimensionality”. On the other hand, MPC is more feasible for real-time applications, leveraging its ability to generate optimized control sequences over a receding horizon [3], [4]. Among these, Model Predictive Path Integral (MPPI) control is a sampling-based MPC method that has gained

Received 31 January 2025; accepted 29 April 2025. Date of publication 9 May 2025; date of current version 2 June 2025. This article was recommended for publication by Associate Editor F. Shi and Editor O. Stasse upon evaluation of the reviewers’ comments. This work was supported in part by the University of Santiago de Chile, in part by NSF CAREER Program under Award 2240163 and in part by DARPA ANSR Program. (corresponding author: Javier Borquez.)

Javier Borquez and Yusuf Umut Ciftci are with the University of Southern California, Los Angeles, CA 90089 USA (e-mail: javierbo@usc.edu).

Luke Raus is with the Olin College of Engineering, Needham, MA 02492 USA.

Somil Bansal is with Stanford University, Stanford, CA 94305 USA (e-mail: somil@stanford.edu).

Digital Object Identifier 10.1109/LRA.2025.3568686

popularity due to its flexibility in handling complex dynamics, uncertainties, and non-linear systems [5]. MPPI leverages stochastic sampling to optimize control trajectories, offering a scalable and efficient way to generate control actions. However, ensuring safety within MPPI has proven challenging, as the basic framework does not account for hard safety constraints. Consequently, various extensions have been proposed to incorporate safety, each solving the co-optimization problem differently.

Classical MPPI-based approaches encourage safety by penalizing unsafe sampled trajectories, which are then mostly ignored in the sample aggregation process. However, this approach cannot guarantee safety and its satisfaction depends on the penalty function. Another drawback is that computation is wasted on ignored samples, reducing the effective number of samples used for optimization. To enforce safety, approaches such as safety filtering and barrier functions have emerged, where constraints are enforced as a post-optimization correction [6], [7]. Such approaches allow to solve the MPPI problem first and apply safety filtering afterward, or combine these two approaches [8], [9], [10]. However, these approaches tend to ignore the long-term effect of safety actions on performance. Other procedures use probabilistic methods to address the safety-performance co-optimization, such as modifying the sampling distribution to favor safer trajectories [11], using stochastic safety certificates to steer away from unsafe regions [12], or improving safety indirectly by enhancing adaptability and robustness to uncertainty [13], [14].

Beyond MPPI literature, control barrier function methods, allow for the co-optimization of safety and performance by solving constrained optimization problems [7], [15]. However, these approaches tend to be myopic, optimizing safety and performance locally without capturing long-term effects, and often rely on Lyapunov-based objectives that limit the expressiveness of performance goals. Filter-aware motion planning has also been explored to mitigate this performance degradation caused by safety interventions, where instead of treating safety reactively, these approaches explicitly account for future interventions. For instance, shielding-aware planning proactively balances performance with the risk of emergency maneuvers triggered by low-probability agent behaviors [16], [17]. Similarly, in reinforcement learning settings, integrating safety filters during training rather than solely during execution has been shown to improve performance and reduce undesired safety interventions [18], [19].

In this work, we introduce DualGuard-MPPI, a novel MPPI algorithm that incorporates safety constraints directly into the MPPI algorithm using Hamilton-Jacobi (HJ) reachability. Our key idea is to incorporate two safety filtering stages in the MPPI algorithm: first, we perform safety filtering along the sampled control perturbations, leading to provably safe rollouts of the system trajectories, which are then used to optimize the performance objective. Second, we apply safety filtering on the selected control sequence before it is applied to the system to prevent multimodal combinations that may compromise safety. In this context, multimodality refers to distinct groups of safe trajectories, where different evasive maneuvers—such as swerving left or right to avoid an obstacle—can both be viable but should not be arbitrarily mixed.

The proposed approach imposes safety constraints throughout the trajectory optimization and control application stages, ensuring that the system operates within safe bounds at all times, without requiring any safety penalty tuning. Moreover, since

all generated samples are provably safe, they all contribute to performance optimization, effectively reducing the MPPI sampling variance and leading to a better performance, as compared to vanilla MPPI algorithm. In summary, the key contributions of this work are:

- An MPPI framework that co-optimizes safety and performance while ensuring safety at all times;
- Elimination of safety penalty tuning within MPPI and improvement in sampling efficiency by guaranteeing that all sampled control sequences are safe.
- Extensive simulations and hardware experiments demonstrating the superior performance and robustness of the proposed approach without compromising safety.

II. PROBLEM STATEMENT

Consider an autonomous system with state $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^n$ that evolves according to dynamics $\dot{\mathbf{x}} = f(\mathbf{x}, u, d)$, where $u \in \mathcal{U}$ and $d \in \mathcal{D}$ are the control and disturbance of the system, respectively. d can represent potential model uncertainties or an actual, adversarial exogenous input to the system. We assume the dynamics are uniformly continuous in u and d , bounded, and Lipschitz continuous in \mathbf{x} for fixed u and d . Finally, let $\xi_{\mathbf{x},t}^{u,d}(\tau)$ denote the system state at time τ , starting from the state \mathbf{x} at time t under control signal $\mathbf{u}(\cdot)$ and disturbance signal $\mathbf{d}(\cdot)$ while following the dynamics. A control signal $\mathbf{u}(\cdot)$ is defined as a measurable function mapping from the time horizon to the set of admissible controls \mathcal{U} , and a disturbance signal is similarly defined. We additionally assume that the control and disturbance signals $\mathbf{u}(\cdot)$ and $\mathbf{d}(\cdot)$ are piecewise continuous in t . This assumption ensures that the system trajectory $\xi_{\mathbf{x},t}^{u,d}$ exists and is unique and continuous for all initial states [20], [21].

In addition, we are given a failure set $\mathcal{F} \subset X$ that the system must avoid at all times (e.g., obstacles for a navigation robot). The safety constraint is encoded via a Lipschitz continuous function $l(\cdot) : \mathcal{F} = \{\mathbf{x} : l(\mathbf{x}) \leq 0\}$. We aim to design a controller that optimally balances the system's safety constraints and performance objectives. The performance objective is given by minimizing a cost function S over the system trajectory, given by:

$$S(\xi) = \phi(\mathbf{x}(t_f), t_f) + \int_0^{t_f} \mathcal{L}(\mathbf{x}(t), u(t), t) dt \quad (1)$$

where ϕ and \mathcal{L} represent the terminal and running cost respectively, and t_f the task completion time. Specifically, we aim to ensure that the control actions u drive the system towards minimizing S while rigorously maintaining system safety by preventing the state \mathbf{x} from entering \mathcal{F} even for the worst case disturbances d . This takes the form of the following constrained optimal control problem:

$$\begin{aligned} u^*(\cdot) &= \underset{u(\cdot)}{\operatorname{argmin}} S \left(\xi_{\mathbf{x},t}^{u,d}, u(\cdot) \right) \\ \text{s.t. } \dot{\mathbf{x}}(t) &= f(\mathbf{x}(t), u(t), d(t)), \\ l(\mathbf{x}(t)) &> 0, u(t) \in \mathcal{U}, d(t) \in \mathcal{D}, \forall t \in [t, t_f] \end{aligned} \quad (2)$$

the optimization problem defined in (2) in general is non-convex and can be difficult to solve. In this work, we propose a novel MPPI method to solve this problem.

III. BACKGROUND

A. Model Predictive Path Integral (MPPI)

Original MPPI [5] aims to solve a stochastic optimal control problem using a zeroth-order sampling-based MPC scheme for undisturbed deterministic general nonlinear systems. Specifically, at each state, MPPI samples K random control sequences around a nominal control sequence for a time horizon $\hat{T} \leq t_f$ discretized over H evenly spaced steps. The control input for the k th sequence at j th time step is given by: $u^k(x, j) = u(x, j) + \delta_j^k$, where $u(x, j)$ is a nominal control sequence and δ_j^k a randomly sampled control perturbation. Next, the state trajectory ξ_j^k and the cost to go $S(\xi_j^k)$ corresponding to each control sequence is computed. It is shown that the optimal control sequence can be approximated by the following update law [8]:

$$u(\mathbf{x}, j)^* \approx u(\mathbf{x}, j) + \frac{\sum_{k=1}^K \exp[-(1/\lambda)S(\xi_j^k)]\delta_j^k}{\sum_{k=1}^K \exp[-(1/\lambda)S(\xi_j^k)]} \quad (3)$$

where $\lambda \in \mathbb{R}^+$ is the temperature parameter that weighs the contributions from different control sequences. The first control in the optimal sequence is applied to the system and the entire process is repeated at the next time step.

Running example (Safe Planar Navigation): As a running example we consider a Dubins' car trying to reach a goal in a cluttered environment. The system can be represented by the following three dimensional dynamics:

$$\dot{\mathbf{x}} = [\dot{x} \quad \dot{y} \quad \dot{\theta}] = [V \cos(\theta) \quad V \sin(\theta) \quad u] \quad (4)$$

With x, y position of the car's center, θ its orientation, $V = 2$ m/s its fixed speed and $u \in [-3, 3]$ rad/s its angular speed input. The failure set in this case is a $10 \text{ m} \times 10 \text{ m}$ square enclosure, randomly cluttered with 40 circular obstacles ranging in diameter from 0.35 m to 3.5 m, as shown in Fig. 2 (zoomed in) and in Fig. 5 (full scale). The cost function S penalizes the distance to the goal and obstacle penetration (to encourage safety).

Dotted lines in Fig. 2 represent possible rollouts of the system following a randomly perturbed control sequence starting from the current state.

The rollouts that fail to avoid obstacles achieve a higher cost (shown in red). Thus, the update law in (3) prioritizes control perturbations that will keep the car safe (shown in a blue-to-purple gradient, with blue indicating lower costs). However, it could be possible that all samples lead to safety violations. Moreover, if the number of safe samples is small, it could lead to a high variance in the performance of MPPI, necessitating the need to account for safety constraints rigorously.

B. Hamilton-Jacobi Reachability

One way to guarantee safe operation of autonomous continuous-time dynamical systems is through Hamilton-Jacobi (HJ) reachability analysis. This approach involves computing the Backward Reachable Tube (BRT) of the failure set \mathcal{F} . The BRT captures the states from which the system is not able to

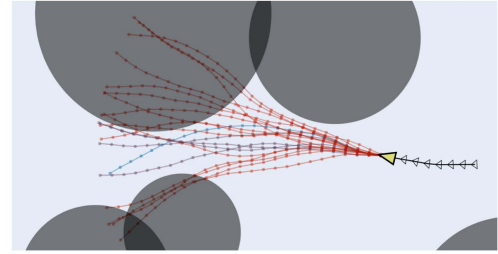


Fig. 2. Visualization of sampling step in the MPPI algorithm.

avoid entering \mathcal{F} within some time horizon T , despite the best control effort.

In HJ reachability, the BRT computation is formulated as a zero-sum game between control and disturbance. Specifically, a cost function is defined to capture the minimum distance to \mathcal{F} over time:

$$J(\mathbf{x}, t, u(\cdot), d(\cdot)) = \min_{\tau \in [t, T]} l(\mathbf{x}(\tau)) \quad (5)$$

The goal is to capture this minimum distance for optimal system trajectories. Thus, we compute the optimal control that maximizes this distance (drives the system away from the failure set) and the worst-case disturbance signal that minimizes the distance. The value function corresponding to this robust optimal control problem is:

$$V(\mathbf{x}, t) = \inf_{\gamma \in \Gamma(t)} \sup_{u(\cdot)} \{J(\mathbf{x}, t, u(\cdot), \gamma[u(\cdot)])\}, \quad (6)$$

where $\Gamma(t)$ defines the set of non-anticipative strategies for the disturbance [22]. The value function in (6) can be computed using dynamic programming, which results in the following final value Hamilton-Jacobi-Isaacs Variational Inequality (HJI-VI) [22], [23], [24]:

$$\begin{aligned} \min\{D_t V(\mathbf{x}, t) + H(\mathbf{x}, t, \nabla V(\mathbf{x}, t)), l(\mathbf{x}) - V(\mathbf{x}, t)\} &= 0 \\ V(\mathbf{x}, T) &= l(\mathbf{x}), \quad \text{for } t \in [0, T] \end{aligned} \quad (7)$$

D_t and ∇ represent the time and spatial gradients of the value function. H is the Hamiltonian, which optimizes over the inner product between the spatial gradients of the value function and the dynamics:

$$H(\mathbf{x}, t, \nabla V(\mathbf{x}, t)) = \min_{u \in \mathcal{U}} \max_{d \in \mathcal{D}} \nabla V(\mathbf{x}, t) \cdot f(\mathbf{x}, u, d) \quad (8)$$

Once the value function is obtained, the BRT is given as the sub-zero level set of the value function:

$$\mathcal{V}(t) = \{\mathbf{x} : V(\mathbf{x}, t) \leq 0\} \quad (9)$$

The corresponding optimal safe control can be derived as:

$$u_{safe}^*(\mathbf{x}, t) = \operatorname{argmax}_{u \in \mathcal{U}} \min_{d \in \mathcal{D}} \nabla V(\mathbf{x}, t) \cdot f(\mathbf{x}, u, d) \quad (10)$$

In safety-ensuring applications, we want to guarantee that the system does not enter \mathcal{F} at any time. To achieve this, we use a time-converged BRT, as the set of unsafe states typically stops expanding after some time. Since our objective is safety, we will rely exclusively on the converged value function $V(\mathbf{x})$ throughout this letter. This allows us to synthesize safety controllers using an expression identical to (10) but without time dependency.

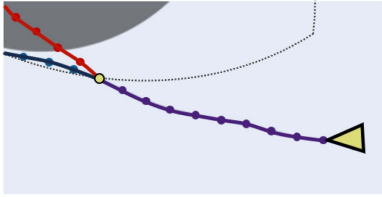


Fig. 3. Trajectory correction by LRF. System under perturbed controls (purple) reaches the safe set boundary (yellow dot). The red path shows an unsafe execution, while the blue path demonstrates how safety filtering maintains the system outside the failure set.

To compute the value function and obtain the BRT and the optimal controller, we can rely on methods that solve the HJI-VI in (6) numerically [25], [26] or using learning-based methods [27], [28].

C. Least Restrictive Filtering

If the BRT and associated converged value function are known, a *Least Restrictive Filter* (LRF) can be deployed to guarantee the safe operation of the system. The LRF is constructed as follows:

$$u(\mathbf{x}, t) = \begin{cases} u_{\text{nom}}(\mathbf{x}, t) & V(\mathbf{x}) > 0 \\ u_{\text{safe}}^*(\mathbf{x}) & V(\mathbf{x}) = 0 \end{cases} \quad (11)$$

Here, $u_{\text{nom}}(\mathbf{x}, t)$ corresponds to an arbitrary nominal controller that might optimize a performance criterion without enforcing safety constraints. This control is used when the value function $V(\mathbf{x})$ is positive, as the system is not at risk of breaching safety. Whenever the system reaches the boundary of the BRT ($V(\mathbf{x}) = 0$), it switches to $u_{\text{safe}}^*(x)$ (10) as this optimal control is guaranteed to maintain or increase $V(\mathbf{x})$ keeping the system from entering the unsafe states determined by the BRT, thereby enforcing safety at all times. We refer the reader to [6] for details on this filtering technique and proof of the safety guarantees.

Running example (*Safe Planar Navigation*):

We can use the HJI-VI in (6) to calculate the value function and associated BRT, over which we can implement a LRF to guarantee the system's safety. In this case considering the dynamics in (4) and the environment definition, we numerically compute the value function using the LevelSetToolbox [25].

In Fig. 3, the purple trajectory represents the system's response to a perturbed control sequence, reaching a critical state at the boundary of the safe set (yellow dot). The boundary of the safe set at this state is illustrated with a dotted line. The red trajectory shows the continuation of the system using the perturbed controls, which leads to the failure set. By contrast, the blue trajectory demonstrates how the unsafe execution can be made safe. This is achieved by applying a LRF step to each control action in the sequence before execution.

IV. DUALGUARD MPPI

As discussed in Section III-A, MPPI addresses the optimal control problem in (2) using a sampling-based method. While MPPI may encourage the satisfaction of safety constraints in (2) via penalizing safety violations in the cost function, ensuring

Algorithm 1: DualGuard MPPI.

Given: $V(\mathbf{x}), u_{\text{safe}}^*(\mathbf{x}), S(\xi, u(\cdot)), \lambda, K, H$
Input: $[u_0, u_1, \dots, u_H]$
while task not completed **do**

```

    Sample control perturbations  $\delta_j^k$ ;
     $\mathbf{x}_0 \leftarrow \text{StateMeasurement}()$ ;
    for  $k \leftarrow 0$  to  $K$  do
        for  $j \leftarrow 0$  to  $H$  do
            if  $V(\mathbf{x}_j) > 0$  then
                 $\Delta_j^k = \delta_j^k$ ; (Safe Rollouts)
            else
                 $\Delta_j^k = u_{\text{safe}}^*(\mathbf{x}_j) - u_j$ ;
                 $\mathbf{x}_{j+1} \leftarrow \mathbf{x}_j + f(\mathbf{x}_j, u_j + \Delta_j^k)\Delta t$ ;
             $S^k = S([\mathbf{x}_0, \dots], [u_0, \dots] + [\Delta_0^k, \dots])$ ;
        for  $j \leftarrow 0$  to  $H$  do
             $u_j^* \leftarrow u_j + \frac{\sum_{k=1}^K \exp(-(1/\lambda)S^k)\Delta_j^k}{\sum_{k=1}^K \exp(-(1/\lambda)S^k)}$ ; (Update Rule)
        if  $V(\mathbf{x}_0) > 0$  then
             $u_0^{**} = u_0^*$ ; (Output Filter)
        else
             $u_0^{**} = u_{\text{safe}}^*(\mathbf{x}_0)$ ;
         $\text{ExecuteControl}(u_0^{**})$ ;
         $[u_0, \dots, u_{H-1}, u_H] \leftarrow [u_1^*, \dots, u_H^*, u_H^*]$ ;
         $\text{TaskCompletionStatus}()$ ;
    
```

safety constraints remains challenging. Moreover, by construction, the high-cost safety breaching sequences are mostly ignored during the optimal control sequence computation, wasting computational resources that could have been used to refine the system performance further. While the safety filtering mechanism discussed in III-C can provide a safety layer after MPPI to enforce the safety constraint, this approach is myopic in nature and might lead to performance impairment in favor of safety.

To overcome these challenges, we propose DualGuard MPPI, a two-layered safety filtering approach. Offline, we use HJ reachability to compute a safety value function considering the disturbed system dynamics. By incorporating disturbances at this stage, we ensure safety against worst-case scenarios. In the first step of our algorithm, safety filtering is incorporated directly into the original MPPI framework, where control sequence rollouts sampled using the undisturbed dynamics are rendered safe by a LRF step. Disturbances are not included during MPPI propagation to streamline the sampling process, since safety constraints are already handled by the LRF. This ensures that all rollouts remain within the safe set and can fully contribute to performance optimization. To ensure that the resultant optimal control sequence also satisfies the safety constraint, we filter the output optimal control sequence by a safety filter as well. In addition to ensuring the safety constraints, the proposed framework leads to an increased sample efficiency as the safe rollouts keep all sampled trajectories safe and relevant to the performance objective, thereby avoiding “sample wastage” due to safety constraint violation. The proposed algorithm is presented in Algorithm 1. Details on the proposed filtering stages are discussed next.

A. Generating Safe Rollouts

As in classical MPPI, we consider K sequences of random perturbations δ_j^k , that modify a nominal control sequence u_j . The perturbed sequences are applied to the deterministic dynamic

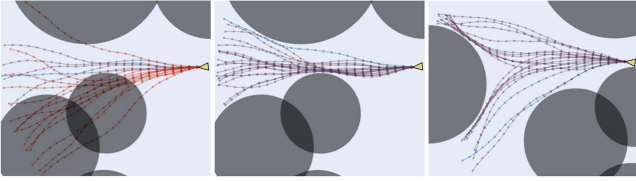


Fig. 4. Unmodified rollouts (Left). Safe rollouts (center). Possible multi-modality on safe rollouts encourages the use of output least restrictive filtering (right).

model of the system while being filtered at each time step along the horizon, using the LRF in (11), resulting in rollouts that are guaranteed to maintain safety. The cost-to-go S^k for the filtered trajectory is calculated over the safety-filtered control perturbation sequence Δ_j^k .

The nominal control sequence u_j , the filtered control perturbations Δ_j^k and the cost over the safe rollouts S^k are used to calculate the optimal control sequence u_j^* using the update rule defined in (3). As we weigh over controls that only produce safe trajectories, all K sequences contribute to the performance optimization, reducing the variance of MPPI algorithm and leading to better performance.

B. Output Least Restrictive Filtering

Even though the optimal control sequence u_j^* was obtained by weighing control perturbations that individually resulted in safe trajectories, the safety guarantees may not hold for u_j^* . For this reason, before applying the first control in the sequence u_0^* to the system, we perform one last LRF step to guarantee the safe operation of the complete system. The rest of the sequence is used as the nominal control for the next time step, and the entire process is repeated.

One motivation for this last filtering stage is scenarios where the safe rollouts present multimodality. We can imagine a vehicle trying to swerve around an obstacle by turning right or left; even if both maneuvers result in safe behaviors, a weighting over rollouts split between these modalities could result in maintaining a straight trajectory, causing a collision. Such issues are resolved by this additional filtering stage which picks one of the safe modes.

Running example (Safe Planar Navigation):

Considering our running example we observe the proposed steps of DualGuard MPPI in Fig. 4. First, we visualize the unmodified rollout step in the left panel. Next, using the same control perturbations, we show how the safe rollout step renders all the samples safe. Sampled trajectories are presented in a blue-to-red scale, corresponding to the associated low-to-high costs.

The right panel of Fig. 4 shows how the safe rollouts might present themselves in a multimodal fashion for some obstacle configurations; this indicates a possible failure mode where the control sequence given by the update rule in (3) could drive the vehicle directly into the obstacles, leading to safety violations. The proposed output filtering stage safeguards against this possibility.

V. SIMULATION STUDIES

We evaluate the performance of our proposed method on simulation studies and a real hardware testbed. For this, we compare the proposed approach against five baseline methods. Each baseline builds on the MPPI control approach but incorporates distinct modifications to enhance safety.

Obstacle Penalty (Obs. penalty): A classical MPPI approach to collision avoidance by adding a high penalty for entering the obstacles in the cost function, incentivizing the system to avoid these penalized regions.

BRT Penalty (BRT penalty): Advanced MPPI methods penalize sampled trajectories that enter a safe zone defined as augmented obstacles, Discrete Barrier States (DBaS), or Control Barrier Functions (CBF) [29], [30], [31]. To represent this technique, we implement a baseline that penalizes trajectories that enter the obstacle set BRT. This preemptive approach penalizes trajectories that have entered states from which collision is unavoidable, even if they have not yet entered the obstacle set within the sampling horizon.

Obstacle Penalty with Output LRF (Obs. pen. + LRF): Extension of the Obstacle Penalty baseline by adding a LRF step. This corresponds to a naive safety filtering approach where the LRF is used to correct any control that would result in a safety violation just before it is applied.

BRT Penalty with Output LRF (BRT pen. + LRF): Similar to the previous baseline, this extends the BRT Penalty baseline by adding least restrictive filtering safety guarantees.

Shield Model Predictive Path Integral (Shield MPPI): We implement the approach described in [9], where a cost related to the CBF condition is considered for rollouts, followed by an approximate CBF-based repair step over the updated control to promote safety. To ensure a fair comparison, we use the HJ value function as a barrier function so that all methods work on the maximal safe set [32].

Evaluation Metrics: To quantitatively assess the performance of each method, we define the following evaluation metrics. These metrics will be measured over batches of simulation runs for each method.

- Failure: Percentage of episodes that result in a safety violation (e.g., a collision).
- RelCost: Average cost of the trajectories and standard error, normalized to our method's mean cost. We only consider common safe trajectories between the compared method and ours. This allows for a fair comparison and ensures that the RelCost is not affected by the high failure penalties for some of the baselines.

Furthermore, when deemed relevant, we define additional performance-related metrics specific to each case study.

A. Safe Planar Navigation

We assess the running-example safe planar navigation with dynamics defined in (4). In each episode, the system must navigate from an initial state \mathbf{x}_0 to a goal region of radius 0.1 m in the xy plane centered at the goal state \mathbf{x}_g , within a time horizon of $T = 20$ s. The cost function is defined as the squared distance from the goal, a control effort penalty, and a method dependent safety penalty $P(\mathbf{x})$:

$$S = \sum_{t=0}^T (\mathbf{x}_t - \mathbf{x}_g)^T Q (\mathbf{x}_t - \mathbf{x}_g) + m(u_t)^2 + P(\mathbf{x}_t) \quad (12)$$

TABLE I
 SAFE PLANAR NAVIGATION RESULTS OVER 100 EPISODES

K	Method	Success	Timeout	Failure	RelCost
1000	Obs. penalty	49	1	50	1.19 ± 0.14*
	BRT penalty	72	21	7	2.05 ± 0.17*
	Obs. pen. + LRF	80	20	0	1.44 ± 0.10*
	BRT pen. + LRF	74	26	0	2.61 ± 0.29*
	Shield MPPI	83	17	0	1.70 ± 0.14*
	DualGuard (Ours)	99	1	0	1.00 ± 0.07
250	Obs. penalty	31	0	69	0.99 ± 0.10
	BRT penalty	58	15	27	2.28 ± 0.23*
	Obs. pen. + LRF	81	19	0	1.80 ± 0.14*
	BRT pen. + LRF	69	31	0	5.46 ± 0.59*
	Shield MPPI	78	21	1	1.85 ± 0.14*
	DualGuard (Ours)	98	2	0	1.00 ± 0.08
60	Obs. penalty	11	0	89	1.07 ± 0.16
	BRT penalty	43	21	36	3.70 ± 0.49*
	Obs. pen. + LRF	70	30	0	2.45 ± 0.18*
	BRT pen. + LRF	55	45	0	6.48 ± 0.60*
	Shield MPPI	64	36	0	2.82 ± 0.20*
	DualGuard (Ours)	96	4	0	1.00 ± 0.07

(*) Indicates statistically significant differences ($p < 0.05$) compared to our method.

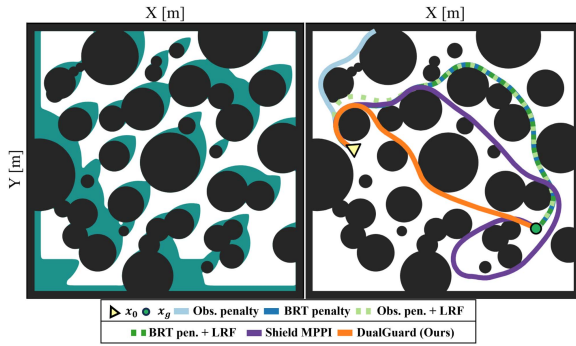


Fig. 5. Safe planar navigation environment, with obstacles in gray and a slice of the BRT for a southwestern heading is in teal.

We test each controller in the same set of 100 episodes defined by pairs $(\mathbf{x}_0, \mathbf{x}_g)$. These initial-goal state pairs were selected randomly but constrained to be outside the BRT, near the environment boundary, and separated by at least 5m for diversity. The results are summarized in Table I. In addition to the metrics previously defined, we also compute the Success rate, defined as the percentage of episodes that reach the goal region safely within the time horizon, and Timeout, defined as the percentage of episodes that fail to reach the target within the given time horizon.

Examining the results for $K = 1000$, we see that DualGuard achieves zero failure rate and highest success rate. While all methods with an output LRF avoid collisions as expected, our method also times-out in fewer episodes, indicating a better performance. The safety-performance co-optimization capabilities of the proposed method are also evident from its lowest RelCost across the safe methods, this difference was further validated to be statistically significant through paired t-tests against DualGuard-MPPI.

To understand the different controllers' behavior, consider the trajectories shown in Fig. 5. Here, the obstacle penalty method fails early by entering a region in the northeast corner which is impossible to escape and hence avoided by all BRT-aware methods. The other baseline methods execute a reasonable trajectory to the goal, but the proposed method manages to find a lower-cost trajectory by taking a shortcut after rounding the first obstacle. This is because the baseline methods are still fundamentally limited by the exploration capabilities of

MPPI – if the underlying samples are unsafe, they contribute little to the performance optimization. On the other hand, the proposed method generates safe rollouts, allowing it to explore more relevant parts of the environment with the same number of samples, as previously visualized in Fig. 4, leading to a better performance optimization.

A consequence of the superior exploration capabilities of the proposed method is that it can achieve similar success rates with far fewer samples. We summarize the results for different methods as we vary the number of samples in Table I. We note that while all methods degrade with fewer samples as expected, DualGuard remains relatively consistent. With only 60 samples, it succeeds in 96% of the episodes, with the closest contender not going over 70%. This could enable its deployment on resource-constrained systems, which MPPI methods typically struggle with.

B. Autonomous Quadrotor Navigation

Next, we demonstrate our method on a 6-dimensional nonlinear quadrotor system. The system follows the dynamics:

$$\begin{aligned} \dot{\mathbf{x}} &= [\dot{x} \quad \dot{y} \quad \dot{z} \quad \dot{v}_x \quad \dot{v}_y \quad \dot{v}_z]^T \\ &= [v_x \quad v_y \quad v_z \quad a_g \tan u_\theta \quad -a_g \tan u_\phi \quad u_T - a_g]^T \end{aligned} \quad (13)$$

where $a_g = 9.81 \text{ m/s}^2$ is gravity, and the controls $u = [u_T, u_\phi, u_\theta]$ correspond to thrust, roll, and pitch angles. Their ranges are $u_T \in [7.81, 11.81] \text{ m/s}^2$ and $u_\phi, u_\theta \in [-17.18^\circ, 17.18^\circ]$. Zero-mean Gaussian noise is added to simulate actuation errors.

The environment is a $10 \text{ m} \times 10 \text{ m} \times 4 \text{ m}$ room containing 10 spherical obstacles of varying sizes. $V(\mathbf{x})$ and the associated BRT of the system are numerically computed using the LevelSetToolbox [25]. The task of the quadrotor is to reach a goal sphere with radius 0.3 m around the goal state \mathbf{x}_g (shown by the green sphere in Fig. 6), starting from a random initial position \mathbf{x}_0 while avoiding collisions with the obstacles. The system must reach the goal location within a time horizon of $T = 10 \text{ s}$; otherwise, the episode is considered timeout. If the system collides with an obstacle or the environment boundary, the episode is considered unsafe.

The cost function consists of distance to the goal, control effort penalty, and a method dependent safety penalty $P(\mathbf{x})$:

$$\begin{aligned} S &= (\mathbf{x}_t - \mathbf{x}_g)^T Q (\mathbf{x}_t - \mathbf{x}_g) \\ &\quad + \|(1/u_{range})^T (u_t - u_{mean})\|_2^2 + P(\mathbf{x}_t) \end{aligned} \quad (14)$$

where $Q = \text{diag}([20, 20, 20, 0, 0, 0])$, $u_{range} = [2, 0.3, 0.3]$ and $u_{mean} = [9.81, 0, 0]$. For all methods, 500 rollouts are simulated for a horizon of 0.3 s. Each method is tested starting from the same 100 randomly sampled initial positions over the environment with at least 5 m distance from the goal location. The results are summarized in Table II.

In this case, the obstacle penalty method leads to a poor success rate. It is partially due to the added disturbance in the system that makes it challenging to ensure safety. A BRT penalty improves the safety rate from 14% to 30%, but it still does not enforce safety, as shown in Fig. 6. By penalizing decreases in safety and the approximate repair step, Shield MPPI decreases the number of collisions to 21. In contrast, safety is guaranteed for baselines with a LRF. Nevertheless, a significant portion of

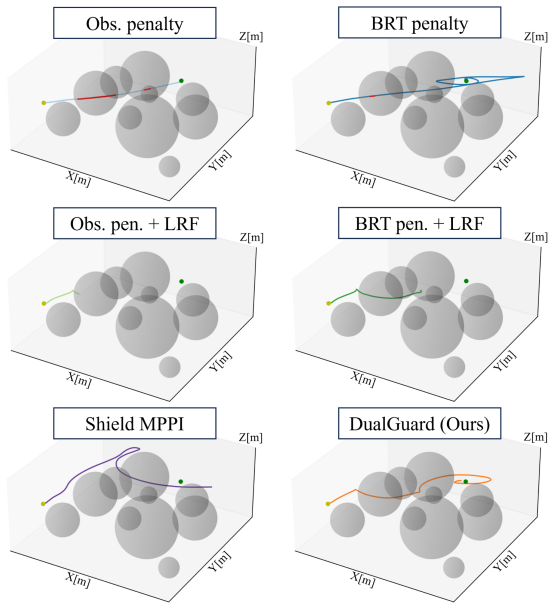


Fig. 6. Quadrotor environment with obstacles (gray), initial position (olive), and goal (dark green). Trajectories for each method are shown in different colors, obstacle intrusions are denoted by red.

TABLE II
QUADROTOR SIMULATION RESULTS

Method	Success	Timeout	Failure	RelCost
Obs. penalty	14	0	86	0.48 ± 0.06*
BRT penalty	30	0	70	0.60 ± 0.09*
Obs. pen. + LRF	68	32	0	1.22 ± 0.06*
BRT pen. + LRF	65	35	0	1.21 ± 0.06*
Shield MPPI	69	10	21	0.91 ± 0.05
DualGuard (Ours)	75	25	0	1.00 ± 0.05

Asterisks (*) indicate statistically significant differences ($p < 0.05$) compared to our method.

the rollouts end up inside obstacles and are unable to contribute to the control synthesis meaningfully. Therefore, methods with safety-filtered actions experience many trajectories that fail to reach the goal within the given task horizon despite maintaining safety; in Fig. 6, we show such timeout executions. By filtering the rollouts, our method incorporates samples that would otherwise be unmeaningful into the optimization. Doing so achieves the best success and safety rates among all methods and achieves better cost on safe trajectories compared to the others that guarantee safety.

VI. HARDWARE EXPERIMENTS - RC CAR

Finally, we consider a real-world miniature RC car with dynamics modeled as (15), with $L = 23.5$ cm, controls $u = [V, \delta]$ with ranges $V \in [0.7, 1.4]$ m/s and $\delta \in [-25^\circ, 25^\circ]$, and disturbances $d_x, d_y \in [-0.1, 0.1]$ to account for model mismatches and state estimation error. The vehicle's task is to complete laps on the racetrack shown in Fig. 1.

$$\begin{aligned} \dot{\mathbf{x}} &= [\dot{x} \quad \dot{y} \quad \dot{\theta}] \\ &= [V \cos(\theta) + d_x, \quad V \sin(\theta) + d_y, \quad V \tan(\delta)/L] \end{aligned} \quad (15)$$

We adapt our evaluation metrics to better reflect real-world, single-run applicability. Instead of batch statistics, we measure the *CompTime* for each method – the time taken to generate and evaluate potential samples within a multiple-lap run. This metric reflects how well-suited each technique is for real-time

TABLE III
HARDWARE EXPERIMENTS RESULTS SUMMARY

Method	CompTime (ms)	RelCost	Speed (m/s)
Obs costs	1.8 ± 0.3	fail	1.00 ± 0.05
BRT costs	1.8 ± 0.3	fail	1.01 ± 0.06
Obs costs + LRF	1.7 ± 0.4	1.1874	1.03 ± 0.12
BRT costs + LRF	1.8 ± 0.4	1.1626	1.04 ± 0.12
Shield-MPPI	1.7 ± 0.2	1.1038	1.04 ± 0.08
DualGuard (Ours)	2.5 ± 0.4	1.0000	1.10 ± 0.11

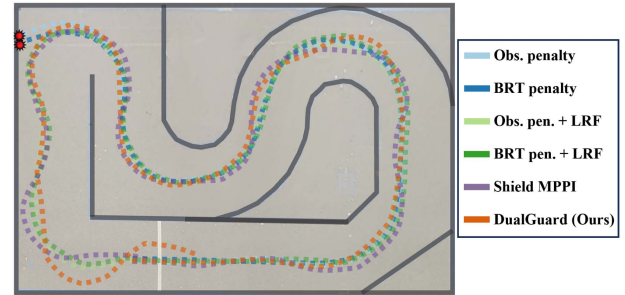


Fig. 7. Top view of the RC car's trajectories under each method.

control. We also report the car's average *Speed* over three laps to measure how aggressive the policy is. The *RelCost* metric remains consistent with the simulations.

As cost function we use (16), where the first term penalizes going slower than $V_{max} = 1.4$ m/s, the second term penalizes the distance from the track's center line, the third term $P(\mathbf{x})$ is a method dependent safety penalty. The value function and associated BRT are numerically computed using [25].

$$S = (V_{max} - V)^2 + K_c(l_{center} - l(x)) + P(\mathbf{x}) \quad (16)$$

The controllers were implemented using JAX [33] on a laptop equipped with an NVIDIA GeForce RTX 4060. We generate 1000 parallel rollouts (with 100 time steps each) in a loop running at 50 Hz. Results are summarized in Table III, and trajectories for the first lap are shown in Fig. 7.

First, we highlight the need for hard safety constraints as methods that only rely on safety penalties fail to clear the top-left tight turn as shown in Fig. 7. Fine-tuning the cost function and MPPI parameters might allow unfiltered methods to complete laps. Still, we want to consider and compare methods that provably allow for safe executions.

DualGuard leads to faster and more performant trajectories than the other safe baselines. Comparison with the LRF baselines shows how the proposed safe rollout step improved the quality of the samples as seen in Fig. 1(b) and (c), leading to better overall performance and higher average speed. Also, the proposed method outperforms Shield-MPPI even after tuning its hyperparameters to the best of our capabilities to maintain safety without an excessive impact on performance.

Computation times are nearly identical across baselines, as each method fundamentally involves calculating performant terms of the cost function and querying the obstacle set or BRT for safety-related penalties. DualGuard introduces an additional LRF step for each sample along the trajectories, resulting in an increase in computational time. Nevertheless, all methods operate well within the 20 ms time budget, leaving ample time for the control loop to handle state estimation, communications, and actuation.

VII. LIMITATIONS AND FUTURE WORK

In this work, we have presented DualGuard-MPPI, a novel MPPI framework for addressing the safety-constrained optimal control problem. By integrating Hamilton-Jacobi reachability analysis with MPPI, our approach ensures strict adherence to safety constraints throughout both the sampling and control execution phases. This combination enables high-performance trajectory optimization without compromising safety, validated through extensive simulation and real-world experiments. DualGuard-MPPI stands out for its capability to eliminate safety-related terms from the cost function, thereby streamlining the optimization process to focus purely on performance objectives.

While these contributions establish DualGuard-MPPI as a robust and scalable framework, certain limitations remain that warrant further exploration. The requirement for a precomputed BRT introduces considerable computational overhead during setup, which may hinder its implementability in rapidly changing environments. Additionally, the reliance on explicitly defined system dynamics may restrict the framework's applicability to systems with highly complex or partially unknown models. Addressing these challenges—such as by leveraging online reachability methods to dynamically update BRTs [34], [35] or using reachability methods for black-box systems [36], [37] could significantly expand the framework's usability. In addition, we will explore the deployment of the proposed approach on other safety-critical robotics applications.

REFERENCES

- [1] A. Altarovici, O. Bokanowski, and H. Zidani, "A general Hamilton-Jacobi framework for non-linear state-constrained control problems," *ESAIM: Control, Optimisation Calculus Variations*, vol. 19, pp. 337–357, 2013.
- [2] H. Wang, A. Dhande, and S. Bansal, "Cooptimizing safety and performance with a control-constrained formulation," *IEEE Control Syst. Lett.*, vol. 8, pp. 2739–2744, 2024.
- [3] M. Schwenzer, M. Ay, T. Bergs, and D. Abel, "Review on model predictive control: An engineering perspective," *Int. J. Adv. Manuf. Technol.*, vol. 117, pp. 1327–1349, 2021.
- [4] F. Borrelli, A. Bemporad, and M. Morari, *Predictive Control for Linear and Hybrid Systems*. New York, NY, USA: Cambridge Univ. Press, 2017.
- [5] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Information-theoretic model predictive control: Theory and applications to autonomous driving," *IEEE Trans. Robot.*, vol. 34, no. 6, pp. 1603–1622, Dec. 2018.
- [6] J. Borquez, K. Chakraborty, H. Wang, and S. Bansal, "On safety and liveness filtering using Hamilton-Jacobi reachability analysis," *IEEE Trans. Robot.*, vol. 40, pp. 4235–4251, 2024.
- [7] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Trans. Autom. Control*, vol. 62, no. 8, pp. 3861–3876, Aug. 2017.
- [8] G. Williams, A. Aldrich, and E. A. Theodorou, "Model predictive path integral control: From theory to parallel computation," *J. Guid., Control, Dyn.*, vol. 40, pp. 344–357, 2017.
- [9] J. Yin, C. Dawson, C. Fan, and P. Tsiotras, "Shield Model Predictive Path Integral: A computationally efficient robust MPC method using control barrier functions," *IEEE Robot. Automat. Lett.*, vol. 8, no. 11, pp. 7106–7113, Nov. 2023.
- [10] K. P. Wabersich and M. N. Zeilinger, "A predictive safety filter for learning-based control of constrained nonlinear dynamical systems," *Automatica*, vol. 129, 2021, Art. no. 109597.
- [11] M. Gandhi, H. Almubarak, and E. Theodorou, "Safe importance sampling in model predictive path integral control," 2023, *arXiv:2303.03441*.
- [12] M. Pereira, Z. Wang, I. Exarchos, and E. Theodorou, "Safe optimal control using stochastic barrier functions and deep forward-backward SDEs," in *Proc. Conf. Robot Learn.*, 2021, pp. 1783–1801.
- [13] J. Higgins, N. Mohammad, and N. Bezzo, "A model predictive path integral method for fast, proactive, and uncertainty-aware UAV planning in cluttered environments," in *Proc. 2023 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2023, pp. 830–837.
- [14] I. S. Mohamed, J. Xu, G. S. Sukhatme, and L. Liu, "Toward efficient MPPI trajectory generation with unscented guidance: U-MPPI control strategy," *IEEE Trans. Robot.*, vol. 41, pp. 1172–1192, 2025.
- [15] J. Zeng, Z. Li, and K. Sreenath, "Enhancing feasibility and safety of nonlinear model predictive control with discrete-time control barrier functions," in *Proc. IEEE Conf. Decis. Control*, 2021, pp. 6137–6144.
- [16] H. Hu, K. Nakamura, and J. F. Fisac, "SHARP: Shielding-aware robust planning for safe and efficient human-robot interaction," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 5591–5598, Apr. 2022.
- [17] H. Hu, D. Isele, S. Bae, and J. F. Fisac, "Active uncertainty reduction for safe and efficient interaction planning: A shielding-aware dual control approach," *Int. J. Robot. Res.*, vol. 43, pp. 1382–1408, 2024.
- [18] J. Thumm, G. Pelat, and M. Althoff, "Reducing safety interventions in provably safe reinforcement learning," in *Proc. 2023 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2023, pp. 7515–7522.
- [19] F. P. Bejarano, L. Brunke, and A. P. Schoellig, "Safety filtering while training: Improving the performance and sample efficiency of reinforcement learning agents," *IEEE Robot. Automat. Lett.*, vol. 10, no. 1, pp. 788–795, Jan. 2025.
- [20] E. A. Coddington and N. Levinson, *Theory of Ordinary Differential Equations*. New York, NY, USA: Tata McGraw-Hill Educ., 1955.
- [21] F. M. Callier and C. A. Desoer, *Linear System Theory*. Berlin, Germany: Springer, 2012.
- [22] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin, "Hamilton-Jacobi reachability: A brief overview and recent advances," in *Proc. IEEE Conf. Decis. Control*, 2017, pp. 2242–2253.
- [23] J. Lygeros, "On reachability and minimum cost optimal control," *Automatica*, vol. 40, no. 6, pp. 917–927, 2004.
- [24] I. Mitchell, A. Bayen, and C. J. Tomlin, "A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games," *IEEE Trans. Autom. Control*, vol. 50, no. 7, pp. 947–957, Jul. 2005.
- [25] I. Mitchell, "A toolbox of level set methods," Tech. Rep. TR-2004-09, 2004. [Online]. Available: <http://www.cs.ubc.ca/mitchell/ToolboxLS/toolboxLS.pdf>
- [26] E. Schmerling and M. Pavone, "HJ reachability: Hamilton-jacobi reachability analysis in jax," 2023. [Online]. Available: https://github.com/StanfordASL/hj_reachability
- [27] S. Bansal and C. J. Tomlin, "DeepReach: A deep learning approach to high-dimensional reachability," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 1817–1824.
- [28] J. F. Fisac, N. F. Lugovoy, V. Rubies-Royo, S. Ghosh, and C. J. Tomlin, "Bridging Hamilton-Jacobi safety analysis and reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2019, pp. 8550–8556.
- [29] Y. Lee, K. H. Choi, and K.-S. Kim, "GPU-enabled parallel trajectory optimization framework for safe motion planning of autonomous vehicles," *IEEE Robot. Automat. Lett.*, vol. 9, no. 11, pp. 10407–10414, Nov. 2024.
- [30] H. Parwana, M. Black, G. Faingkos, B. Hoxha, H. Okamoto, and D. Prokhorov, "Model predictive path integral methods with reach-avoid tasks and control barrier functions," 2024, *arXiv:2407.13693*.
- [31] X. Zhang, J. Lu, Y. Hui, H. Shen, L. Xu, and B. Tian, "RAPA-planner: Robust and efficient motion planning for quadrotors based on parallel RA-MPPI," *IEEE Trans. Ind. Electron.*, early access, Dec. 11, 2024, doi: [10.1109/TIE.2024.3508056](https://doi.org/10.1109/TIE.2024.3508056).
- [32] J. J. Choi, D. Lee, K. Sreenath, C. J. Tomlin, and S. L. Herbert, "Robust control barrier-value functions for safety-critical control," in *Proc. 2021 60th IEEE Conf. Decis. Control*, 2021, pp. 6814–6821.
- [33] J. Bradbury, "JAX: Composable transformations of Python NumPy programs," 2018. [Online]. Available: <http://github.com/google/jax>
- [34] A. Bajcsy, S. Bansal, E. Bronstein, V. Tolani, and C. J. Tomlin, "An efficient reachability-based framework for provably safe autonomous navigation in unknown environments," in *Proc. 2019 IEEE 58th Conf. Decis. Control*, 2019, pp. 1758–1765.
- [35] J. Borquez, K. Nakamura, and S. Bansal, "Parameter-conditioned reachable sets for updating safety assurances online," in *Proc. 2023 IEEE Int. Conf. Robot. Automat.*, 2023, pp. 10553–10559.
- [36] V. K. Chilakamarri, Z. Feng, and S. Bansal, "Reachability analysis for black-box dynamical systems," 2024, *arXiv:2410.07796*. [Online]. Available: <https://arxiv.org/abs/2410.07796>
- [37] K.-C. Hsu, V. Rubies-Royo, C. Tomlin, and J. Fisac, "Safety and liveness guarantees through reach-avoid reinforcement learning," in *Proc. Robot. Sci. Syst.*, Jul. 2021, doi: [10.15607/RSS.2021.XVII.077](https://doi.org/10.15607/RSS.2021.XVII.077).