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

Javier Borquez[1], Luke Raus[2], Yusuf Umut Ciftci[1], and Somil Bansal[3]

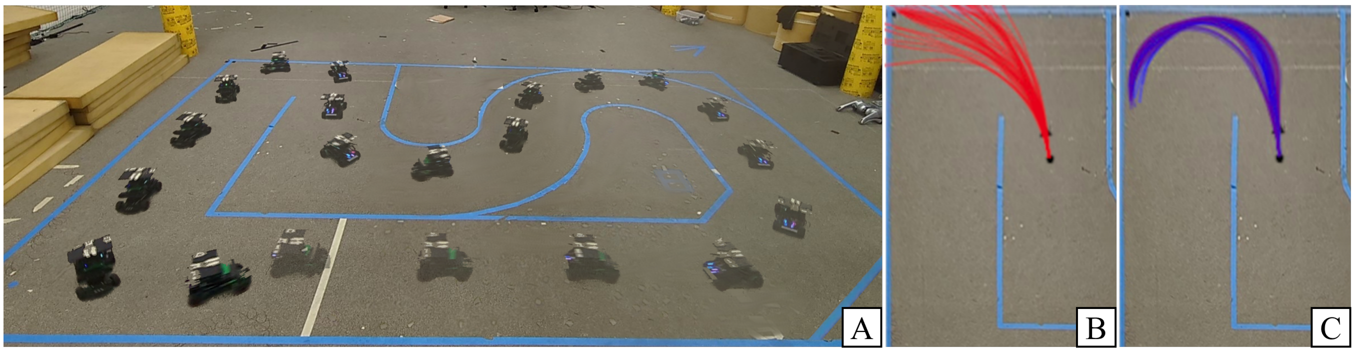

Fig. 1. DualGuard MPPI solves optimal control problems with hard safety constraints by integrating safety filtering during sampling, ensuring safe and efficient rollouts. An output least-restrictive filter handles potential multimodality during execution. (A) In an RC car experiment, the vehicle completes laps safely while maximizing speed and staying centered. (B) Classical MPPI produces unsafe, high-cost trajectories near tight corners, breaching track boundaries. (C) DualGuard MPPI generates only safe, performance-driven rollouts. See Section V for details.

***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. This integration allows DualGuard-MPPI to enforce strict safety constraints; at the same time, it facilitates a more effective exploration with the same number of samples, leading to better performance optimization. Through simulation and hardware experiments, we demonstrate that DualGuard achieves higher performance compared to existing MPPI methods, without compromising safety.**

I. Introduction

Co-optimizing safety and performance is essential for autonomous systems operating in safety-critical environments. While dynamic programming provides rigorous solutions to constrained control [1, 2], its computational demands limit real-time use. In contrast, sampling-based MPC methods like Model Predictive Path Integral (MPPI) control [3] offer scalability for complex, uncertain, and nonlinear systems, but struggle to enforce hard safety constraints.

Existing MPPI extensions address safety by penalizing unsafe trajectories or applying post-hoc filtering [4, 5], but these approaches cannot guarantee constraint satisfaction or neglect cost optimization for safety guarantees. Barrier function methods [6, 7] and probabilistic safety strategies [8, 9] offer alternatives, yet often neglect the long-term impact of safety actions or rely on limited performance formulations. Recent work has explored integrating safety filters during planning or training to proactively reduce undesired interventions [10, 11].

This work introduces *DualGuard-MPPI*, a novel MPPI algorithm that enforces safety constraints using Hamilton-Jacobi (HJ) reachability during both sampling and execution. Unsafe control perturbations are filtered upfront to generate only provably safe rollouts, and multimodal safe trajectories are managed to prevent unsafe combinations.

Contributions:

- A sampling-based MPC framework that guarantees safety throughout planning and execution using HJ reachability.
- Elimination of safety penalty tuning and improved sample efficiency by filtering unsafe trajectories.
- Demonstrated superior performance and robustness in simulation and hardware experiments.

II. Problem Statement

Consider a system with state $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^n$ governed by $\dot{\mathbf{x}} = f(\mathbf{x}, u, d)$, where $u \in \mathcal{U}$ and $d \in \mathcal{D}$ are control and disturbance inputs. The disturbance d may represent model uncertainty or adversarial input. We assume f is uniformly continuous in u and d, bounded, and Lipschitz continuous in $\mathbf{x}$ for fixed inputs.

Let $\xi_{\mathbf{x},t}^{\mathbf{u},\mathbf{d}}(\tau)$ denote the system state at time τ, starting from $\mathbf{x}$ at time t under control $\mathbf{u}(\cdot)$ and disturbance $\mathbf{d}(\cdot)$. Both signals are measurable functions of time into $\mathcal{U}$ and $\mathcal{D}$, and assumed piecewise continuous to ensure well-posedness of the trajectory [12, 13].

A failure set $\mathcal{F} \subset X$—e.g., obstacle regions—is defined via a Lipschitz function l such that $\mathcal{F} = \{\mathbf{x} : l(\mathbf{x}) \leq 0\}$. The goal is to minimize a cost S over the trajectory while keeping $\mathbf{x}$ out of $\mathcal{F}$ under worst-case d. The cost is:

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

We thus formulate the constrained optimal control problem as:

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

Where (2) is generally non-convex and hard to solve. We propose a novel MPPI-based method to address this challenge.

III. Background

A. Model Predictive Path Integral (MPPI)

MPPI [3] is a zeroth-order, sampling-based MPC method for stochastic optimal control of nonlinear systems. At each state, MPPI samples K random control sequences around a nominal sequence over a horizon $\hat{T} \leq t_f$, discretized into H steps. Each sample follows: $u^k(x, j) = u(x, j) + \delta_j^k$, where $u(x, j)$ is the nominal control and δ_j^k is a random perturbation.

The corresponding trajectory ξ_j^k and cost $S(\xi_j^k)$ are computed for each sequence. The optimal control is approximated using the weighted average:

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

$\lambda > 0$ the temperature parameter controls the influence of each sample. Only the first control in the updated sequence is applied, and the process repeats at the next step [14].

B. Hamilton-Jacobi Reachability

Hamilton-Jacobi (HJ) reachability guarantees safety by computing the Backward Reachable Tube (BRT) of a failure set $\mathcal{F}$, i.e., the set of states from which the system cannot avoid entering $\mathcal{F}$ within a time horizon T, even under optimal control. This is framed as a zero-sum game between control and disturbance, with cost defined as the closest a trajectory ever was to $\mathcal{F}$:

$$J(\mathbf{x}, t, u(\cdot), d(\cdot)) = \min_{\tau \in [t, T]} l(\mathbf{x}(\tau)) \tag{4}$$

The value function V for this optimal control problem is:

$$V(\mathbf{x}, t) = \inf_{\gamma \in \Gamma(t)} \sup_{u(\cdot)} \{J(\mathbf{x}, t, u(\cdot), \gamma[u](\cdot))\} \tag{5}$$

where $\Gamma(t)$ is the set of non-anticipative disturbance strategies [15]. Using dynamic programming, this yields the HJI Variational Inequality (HJI-VI) [15, 16, 17]:

$$\begin{gathered} \min\{D_t V(\mathbf{x}, t) + H(\mathbf{x}, t, \nabla V), l(\mathbf{x}) - V(\mathbf{x}, t)\} = 0 \\ V(\mathbf{x}, T) = l(\mathbf{x}), \quad t \in [0, T] \end{gathered} \tag{6}$$

Here, D_t and ∇ are the time and spatial gradients, and the Hamiltonian is:

$$H(\mathbf{x}, t, \nabla V) = \min_{u \in \mathcal{U}} \max_{d \in \mathcal{D}} \nabla V \cdot f(\mathbf{x}, u, d) \tag{7}$$

The BRT is the sub-zero level set of V:

$$\mathcal{V}(t) = \{\mathbf{x} : V(\mathbf{x}, t) \leq 0\} \tag{8}$$

The corresponding optimal safe controller is:

$$u_{safe}^*(\mathbf{x}, t) = \underset{u \in \mathcal{U}}{\operatorname{argmax}} \min_{d \in \mathcal{D}} \nabla V \cdot f(\mathbf{x}, u, d) \tag{9}$$

To ensure safety at all times, we use the time-converged BRT, relying on the stationary value function $V(\mathbf{x})$, removing time dependence from (9). The value function can be computed via numerical solvers [18, 19] or learning-based methods [20, 21].

C. Least Restrictive Filtering

Given the converged BRT, we define a *Least Restrictive Filter* (LRF) to switch between a nominal and a safety controller:

$$u(\mathbf{x}, t) = \begin{cases} u_{\text{nom}}(\mathbf{x}, t) & V(\mathbf{x}) > 0 \\ u_{\text{safe}}^*(\mathbf{x}) & V(\mathbf{x}) = 0 \end{cases} \tag{10}$$

Here, u_{nom} may optimize performance without safety guarantees. When $V(\mathbf{x}) > 0$, the system is safe, and u_{nom} is applied. When $V(\mathbf{x}) = 0$, the LRF switches to the safe controller (9), which guarantees safety by keeping the system outside the BRT. See [5] for formal guarantees.

IV. DualGuard MPPI

To address the challenge of enforcing safety in sampling-based control, we propose *DualGuard MPPI*, a two-stage safety filtering framework that integrates HJ reachability into MPPI. Offline, we compute a safety value function under worst-case disturbances. Online, we modify the MPPI procedure (Alg. 1) to apply safety filtering both during trajectory sampling and before control execution, as described next:

A. Generating Safe Rollouts

As in standard MPPI, we sample K perturbation sequences δ_j^k around a nominal sequence u_j. These are applied to the deterministic dynamics, but each step is filtered with the LRF (Eq. 10) to ensure safety. The resulting safe rollouts are used to compute costs S^k and filtered control perturbations Δ_j^k.

The updated optimal control u_j^* is computed using (3), now averaging only over safe trajectories. This improves performance by avoiding wasted samples.

B. Output Least Restrictive Filtering

Although u_j^* is built from safe rollouts, the resulting averaged action is not guaranteed to be safe. Therefore, we apply one final LRF step to u_0^* before execution. One example case where this can happen is with multi-modal rollouts arising from multiple safe modes (e.g., swerving left vs. right).

Algorithm 1: DualGuard MPPI

Given: $V(\mathbf{x}), u^*_{safe}(\mathbf{x}), S(\xi, u(\cdot)), \lambda, K, H$
Input: $[u_0, u_1, ..., u_H]$
while *task not completed* **do**
  Sample control perturbations δ^k_j;
  $\mathbf{x}_0 \leftarrow StateMeasurement()$;
  for $k \leftarrow 0$ *to* K **do**
    for $j \leftarrow 0$ *to* H **do**
      if $V(\mathbf{x}_j) > 0$ **then**
        $\Delta^k_j = \delta^k_j$; **(Safe Rollouts)**
      else
        $\Delta^k_j = u^*_{safe}(\mathbf{x}_j) - u_j$;
      $\mathbf{x}_{j+1} \leftarrow \mathbf{x}_j + f(\mathbf{x}_j, u_j + \Delta^k_j)\Delta t$;
    $S^k = S([\mathbf{x}_0, ...], [u_0, ...] + [\Delta^k_0, ...])$;
  for $j \leftarrow 0$ *to* H **do**
    $u^*_j \leftarrow u_j + \frac{\sum_{k=1}^{K} \exp(-(1/\lambda)S^k)\Delta^k_j}{\sum_{k=1}^{K} \exp(-(1/\lambda)S^k)}$; **(Update Rule)**
  if $V(\mathbf{x}_0) > 0$ **then**
    $u^{**}_0 = u^*_0$; **(Output Filter)**
  else
    $u^{**}_0 = u^*_{safe}(\mathbf{x}_0)$;
  $ExecuteControl(u^{**}_0)$;
  $[u_0, ..., u_{H-1}, u_H] \leftarrow [u^*_1, ..., u^*_H, u^*_H]$;
  $TaskCompletionStatus()$;

Illustrative Example ***(Safe Planar Navigation)*****:** We illustrate the proposed DualGuard-MPPI method using a Dubins' car navigating toward a goal in a cluttered environment. The car's dynamics are:

$$\dot{\mathbf{x}} = \begin{bmatrix}\dot{x} & \dot{y} & \dot{\theta}\end{bmatrix} = \begin{bmatrix}V\cos(\theta) & V\sin(\theta) & u\end{bmatrix} \quad (11)$$

where (x, y) is the position, θ is the heading, V is a constant speed, and u is the control input. We assume a cost function that penalizes obstacle penetration.

Figure 2 (left) shows the standard MPPI sampling step: dotted lines are rollouts from perturbed control sequences. Unsafe trajectories incur high costs (red), while safer ones are preferred by the update law in Eq. (3) (blue to purple). However, MPPI may still fail if most samples violate safety, or if the number of safe samples is too small, increasing variance and reducing performance.

Figure 2 (center) illustrates the effect of the *Safe Rollouts* step, where we apply stepwise LRF filtering, which ensures all samples remain within the safe set. These filtered rollouts are used for cost evaluation and control update, improving safety and sample efficiency.

Figure 2 (right) shows a key motivation for the *Output Least Restrictive Filtering* step: even if all rollouts are safe, multimodal behaviors—e.g., turning left or right to avoid an obstacle—can lead to unsafe averaging. Output LRF filtering ensures that only one consistent safe action is executed.

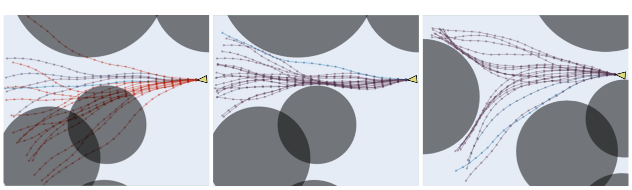

Fig. 2. Unfiltered rollouts (Left). Safe rollouts via LRF (Center). Multimodal safe samples motivate output filtering (Right).

V. Hardware Experiments

We compare against five MPPI-based baselines with varying safety mechanisms:

Obs. Penalty: Classical MPPI with high costs for entering obstacle regions to encourage avoidance.

BRT Penalty: Penalizes trajectories that enter the BRT of obstacles—states from which collisions become inevitable—representing preemptive safety [22, 23, 24].

Obs. penalty + LRF: Adds an output LRF to the obstacle-penalty baseline, filtering unsafe controls just before execution.

BRT penalty + LRF: Adds an output LRF to the BRT-penalty baseline, filtering unsafe controls just before execution.

Shield MPPI: Implements the method in [4], combining CBF-inspired cost terms with a CBF-based repair step. We use the HJ value function as the barrier to ensure all methods operate within the same maximal safe set [25].

Evaluation Metrics: For hardware experiments, we use metrics tailored to real-time, single-run scenarios:

- **Failure:** Whether a safety violation (e.g., collision) occurs during a run.
- **RelCost:** Average trajectory cost and standard error, normalized to our method, evaluated over shared safe trajectories.
- **CompTime:** Time to generate and evaluate control samples, reflecting real-time feasibility.
- **Speed:** Average speed over three laps, indicating policy aggressiveness.
- **LapTime:** Average lap completion time over three laps.

A comprehensive comparison of these baselines, including simulation-based batch statistics and significance testing, is available in the full version of this work published in the IEEE Robotics and Automation Letters (RAL) 10.1109/LRA.2025.3568686. In this section, we focus on real-world performance using hardware experiments and single-run metrics that better reflect practical constraints such as safety, responsiveness, and policy aggressiveness.

A. RC Car Experiments

We consider a real-world miniature RC car with dynamics modeled as (12), with $L = 23.5$ cm, controls $u = [V, \delta]$ with ranges $V \in [0.7, 1.4]$ m/s and $\delta \in [-25°, 25°]$, 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.

$$\dot{\mathbf{x}} = [\dot{x} \ \dot{y} \ \dot{\theta}] = [V\cos(\theta) + d_x, \ V\sin(\theta) + d_y, \ V\tan(\delta)/L] \quad (12)$$

As cost function we use (13), where the first term penalizes going slower than $V_{\text{max}} = 1.4$ 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 [18].

$$S = (V_{max} - V)^2 + K_c(l_{center} - l(x)) + P(\mathbf{x}) \quad (13)$$

TABLE I. Hardware experiments results summary.

Method	CompTime (ms)	RelCost	Speed (m/s)	LapTime (s)
Obs costs	1.8 ± 0.3	fail	1.00 ± 0.05	fail
BRT costs	1.8 ± 0.3	fail	1.01 ± 0.06	fail
Obs costs + LRF	1.7 ± 0.4	1.1874	1.03 ± 0.12	16.54
BRT costs + LRF	1.8 ± 0.4	1.1626	1.04 ± 0.12	16.37
Shield-MPPI	1.7 ± 0.2	1.1038	1.04 ± 0.08	16.21
DualGuard (Ours)	2.5 ± 0.4	**1.0000**	**1.10 ± 0.11**	**15.06**

The controllers were implemented using JAX [26] 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 $50Hz$. Results are summarized in Table I, and trajectories for the first lap are shown in Fig 3.

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 3. 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 samples quality as seen in Fig. 1(B)(C), leading to better overall performance, higher average speed and shorter lap times. 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 $20ms$ time budget, leaving ample time for the control loop to handle state estimation, communications, and actuation.

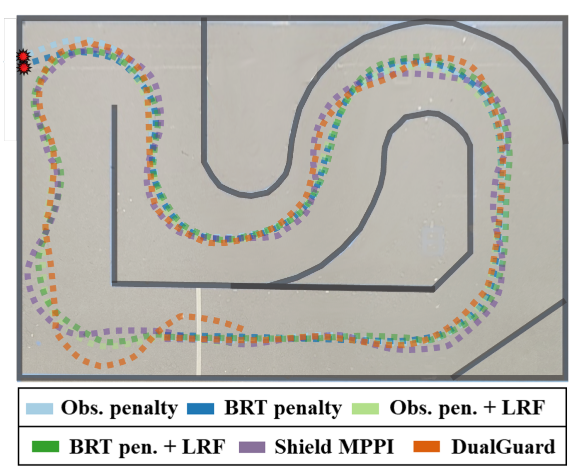

Fig. 3. Top view of the RC car's trajectories under each method.

VI. 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 [27, 28] or using reachability methods for black-box systems [29, 30] 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] Altarovici, Albert, Bokanowski, Olivier, and Zidani, Hasnaa, "A general hamilton-jacobi framework for non-linear state-constrained control problems," *ESAIM: COCV*, 2013.

[2] H. Wang, A. Dhande, and S. Bansal, "Cooptimizing safety and performance with a control-constrained formulation," *IEEE Control Systems Letters*, vol. 8, pp. 2739–2744, 2024.

[3] 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 Transactions on Robotics*, 2018.

[4] J. Yin, C. Dawson, C. Fan, and P. Tsiotras, "Shield mppi: A computationally efficient robust mpc method using control barrier functions," *IEEE Robotics and Automation Letters*, 2023.

[5] J. Borquez, K. Chakraborty, H. Wang, and S. Bansal, "On safety and liveness filtering using hamilton–jacobi reachability analysis," *IEEE Transactions on Robotics*, vol. 40, pp. 4235–4251, 2024.

[6] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, 2016.

[7] J. Zeng, Z. Li, and K. Sreenath, "Enhancing feasibility and safety of nonlinear model predictive control with discrete-time control barrier functions," in *IEEE Conference on Decision and Control*, 2021.

[8] M. Gandhi, H. Almubarak, and E. Theodorou, "Safe importance sampling in model predictive path integral control," *arXiv preprint arXiv:2303.03441*, 2023.

[9] M. Pereira, Z. Wang, I. Exarchos, and E. Theodorou, "Safe optimal control using stochastic barrier functions and deep forward-backward sdes," in *Conference on Robot Learning*. PMLR, 2021.

[10] H. Hu, K. Nakamura, and J. F. Fisac, "Sharp: Shielding-aware robust planning for safe and efficient human-robot interaction," *IEEE Robotics and Automation Letters*, 2022.

[11] J. Thumm, G. Pelat, and M. Althoff, "Reducing safety interventions in provably safe reinforcement learning," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023.

[12] E. A. Coddington and N. Levinson, *Theory of ordinary differential equations*. Tata McGraw-Hill Education, 1955.

[13] F. M. Callier and C. A. Desoer, *Linear system theory*. Springer Science & Business Media, 2012.

[14] G. Williams, A. Aldrich, and E. A. Theodorou, "Model predictive path integral control: From theory to parallel computation," *Journal of Guidance, Control, and Dynamics*, 2017.

[15] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin, "Hamilton-Jacobi Reachability: A brief overview and recent advances," in *IEEE Conference on Decision and Control (CDC)*, 2017.

[16] J. Lygeros, "On reachability and minimum cost optimal control," *Automatica*, vol. 40, no. 6, pp. 917–927, 2004.

[17] I. Mitchell, A. Bayen, and C. J. Tomlin, "A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games," *IEEE Transactions on Automatic Control (TAC)*, 2005.

[18] I. Mitchell, "A toolbox of level set methods," *http://www.cs. ubc. ca/mitchell/ToolboxLS/toolboxLS. pdf, Tech. Rep. TR-2004-09*, 2004.

[19] E. Schmerling and M. Pavone, "hj reachability: Hamilton-jacobi reachability analysis in jax," 2023. [Online]. Available: https://github.com/StanfordASL/hj_reachability

[20] S. Bansal and C. J. Tomlin, "DeepReach: A deep learning approach to high-dimensional reachability," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[21] J. F. Fisac, N. F. Lugovoy, V. Rubies-Royo, S. Ghosh, and C. J. Tomlin, "Bridging hamilton-jacobi safety analysis and reinforcement learning," in *IEEE International Conference on Robotics and Automation*, 2019.

[22] Y. Lee, K. H. Choi, and K.-S. Kim, "Gpu-enabled parallel trajectory optimization framework for safe motion planning of autonomous vehicles," *IEEE Robotics and Automation Letters*, 2024.

[23] H. Parwana, M. Black, G. Fainekos, B. Hoxha, H. Okamoto, and D. Prokhorov, "Model predictive path integral methods with reach-avoid tasks and control barrier functions," *arXiv preprint arXiv:2407.13693*, 2024.

[24] 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 Transactions on Industrial Electronics*, 2024.

[25] J. J. Choi, D. Lee, K. Sreenath, C. J. Tomlin, and S. L. Herbert, "Robust control barrier–value functions for safety-critical control," in *2021 60th IEEE Conference on Decision and Control (CDC)*, 2021.

[26] J. B. et al., "JAX: composable transformations of Python+NumPy programs," 2018. [Online]. Available: http://github.com/google/jax

[27] 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 *2019 IEEE 58th Conference on Decision and Control (CDC)*, 2019, pp. 1758–1765.

[28] J. Borquez, K. Nakamura, and S. Bansal, "Parameter-conditioned reachable sets for updating safety assurances online," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[29] V. K. Chilakamarri, Z. Feng, and S. Bansal, "Reachability analysis for black-box dynamical systems," 2024. [Online]. Available: https://arxiv.org/abs/2410.07796

[30] K.-C. Hsu, V. Rubies-Royo, C. Tomlin, and J. Fisac, "Safety and liveness guarantees through reach-avoid reinforcement learning," in *Robotics: Science and Systems XVII*, ser. RSS2021, 2021.