BR-MPPI: Barrier Rate guided MPPI for Enforcing Multiple Inequality Constraints with Learned Signed Distance Field

Hardik Parwana[1], Taekyung Kim[1], Kehan Long[2], Bardh Hoxha[3], Hideki Okamoto[3], Georgios Fainekos[3], and Dimitra Panagou[1]

***Abstract*— Model Predictive Path Integral (MPPI) controller is used to solve unconstrained optimal control problems and Control Barrier Function (CBF) is a tool to impose strict inequality constraints, a.k.a, barrier constraints. In this work, we propose an integration of these two methods that employ *CBF-like* conditions to guide the control sampling procedure of MPPI. CBFs provide an inequality constraint restricting the rate of change of barrier functions by a class-$\mathcal{K}$ function of the barrier itself. We instead impose the CBF condition as an *equality constraint* by choosing a parametric linear class-$\mathcal{K}$ function and treating this parameter as a state in an augmented system. The time derivative of this parameter acts as an additional control input that is designed by MPPI. A cost function is further designed to reignite Nagumo's theorem at the boundary of the safe set by promoting specific values of class-$\mathcal{K}$ parameter to enforce safety. Our problem formulation results in an MPPI subject to multiple state and control-dependent equality constraints which are non-trivial to satisfy with randomly sampled control inputs. We therefore also introduce state transformations and control projection operations, inspired by the literature on path planning for manifolds, to resolve the aforementioned issue.**

We show empirically through simulations and experiments on quadrotor that our proposed algorithm exhibits better sampled efficiency and enhanced capability to operate closer to the safe set boundary over vanilla MPPI

I. Introduction

Autonomous systems have increased presence in interactive components have been seeing an increasing use of learning based techniques in real-time. These applications range from real-time active learning for estimating unknown quantities to simply using offline learned components like dynamics function or distance fields. Autonomous systems have been seeing an increasing use in several applications. The ability to plan kinodynamic paths in real-time and operate in constrained environments, especially near constrained set boundaries, is quintessential for performing the aforementioned tasks in the real world. Two popular methods that fall under the paradigm mentioned above are Model Predictive Path Integral (MPPI) control[1] and Control Barrier Function (CBF) based quadratic program (QP) controllers[2]. MPPI and CBF have demonstrated their capabilities in generating safe paths for several complex systems. however, they suffer from several shortcomings. The most notable being that MPPI cannot guarantee inequality constraint satisfaction and CBF-QP controllers are myopic and often lead to suboptimal trajectories. In this work, we introduce a method to integrate both methods by using CBF-inspired conditions to guide the MPPI trajectory sampling procedure.

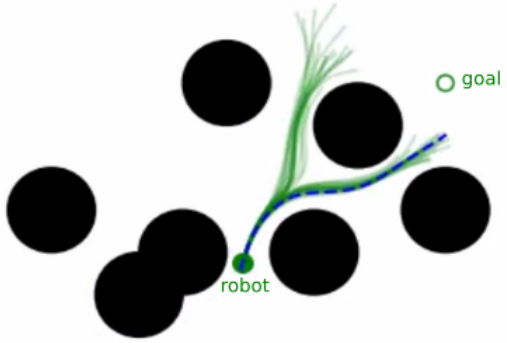


Fig. 1: The sampled paths (green) and the weighted average path (blue) of our proposed MPPI for a robot navigating among black obstacles. As shown, BR-MPPI plans paths in an augmented state space that produces multimodal paths in the original state space.

The Vanilla MPPI[1] solves unconstrained optimal control problems by sampling several robot control input trajectories, assigning costs to the resulting state trajectories based on user-given cost function, and then returning a weighted average based on the assigned costs. MPPI is used to impose inequality constraints in a soft manner by adding a term in the cost function that penalizes the violation of constraints. This leaves MPPI susceptible to constraint violation, especially when few samples are used[3]. Furthermore, the weights of constraint penalty in the cost function may require extensive tuning to construct paths that can navigate close to constraint boundaries (for example close to obstacles) without violating them. Several works have been proposed to improve MPPI's performance[4], [5], [6], [7]. In this work, we limit our discussion to methods that employ CBFs for achieving better constraint maintenance performance.

One of the simplest ways to employ CBF is to filter the output of MPPI by a CBF-QP optimization problem as done in [3]. Another way, also proposed in [3], is to first compute

[1]Robotics Department, University of Michigan, Ann Arbor, MI 48109, USA {hardiksp, taekyung, dpanagou}@umich.edu

[2]Contextual Robotics Institute, University of California San Diego, La Jolla, CA 92093, USA. k3long@ucsd.edu.

[3]Toyota Motor North America, Research & Development, Ann Arbor, MI 48105, USA. <first_name.last_name>@toyota.com

This work was primarily conducted during the summer internships of Hardik Parwana and Kehan Long at Toyota Motor North America R & D.

trust regions, defined as the mean and covariance of the control sampling distribution, using CBF-based optimization and pass it to MPPI. However, this requires an SDP to be solved at every time instant of each sampled trajectory, a computational burden that is traded off by selecting fewer samples but overall still leads to better real-time performance. A cost function is designed to penalize CBF inequality constraint violation for deterministic dynamical systems in [8] and for stochastic dynamical systems in [9]. The resulting output trajectory is then passed to another layer of gradient-based nonlinear optimization that outputs, subject to convergence, a trajectory that achieves hard satisfaction of CBF inequality constraint.

We introduce a method that integrates CBF like conditions with the MPPI sampling procedure but unlike [3] does not require solving an optimization with inequality constraints. CBFs bound the rate of change of barrier functions by a class-$\mathcal{K}$ function of the barrier itself in an inequality constraint. We instead impose the CBF condition as an *equality constraint* by choosing a parametric linear class-$\mathcal{K}$ function and treating this parameter as a state in an augmented system. The time derivative of this parameter acts as an additional control input that is designed by MPPI. A cost function is designed to reignite Nagumo's theorem at the boundary of a safe set by promoting specific values of class-$\mathcal{K}$ parameter to enforce safety. Our MPPI leads to multi-modal trajectory distribution in the original state space of the robot as shown in Fig. 1 which is uncharacteristic of vanilla MPPI.

The aforementioned procedure poses a challenge: the equality constraints introduce manifolds, and randomly chosen control inputs by MPPI are not guaranteed to lie on this manifold. We therefore also introduce a state-dependent projection operation for control inputs to restrict robot state motion along these manifolds. This projection operation admits an analytical formula under control, affine dynamics, and can be implemented efficiently. The work most similar to ours is [10] where they extend the MPPI state space to include the barrier value. However, they require infinity-going barriers and further, since the barrier value is completely specified by the robot state, its dynamics in the augmented state space is what results from the natural flow of the robot state. In contrast, we do not require infinity-going barriers and include the class-$\mathcal{K}$ parameter as a state rather than the barrier itself. Further, we also impose a specific dynamics model on the barrier derivative rather than letting it flow naturally along system dynamics.

To the best of our knowledge, this is the first work to consider the following: 1) an MPPI with multiple state-and-control dependent equality constraints, 2) an application of results to MPPI from the well-established theory of set invariance to enforce constraint maintenance in a higher-dimensional lifted state space that is constructed to monitor not only constraint values but also its rate of change, and finally, 3) the first application of MPPI to design or adapt the class-$\mathcal{K}$ function parameters of CBFs. While the adaptation of class-$\mathcal{K}$ function parameter for improving the performance of CBF-QP controllers is an active area of research[11], [12], [13], [14], [15], it has not been studied in the context of MPPI.

II. Preliminaries

A. Notation

The set of integers, real numbers, and positive real numbers are denoted as $\mathbb{Z}$, $\mathbb{R}$, and $\mathbb{R}^+$ respectively. The time derivative of x is denoted by $\dot{x}$. Given $x \in \mathbb{R}^n$, $||x||_Q \coloneqq \sqrt{x^T Q x}$ is called the weighted Q norm for a positive definite matrix Q. $\langle a, b \rangle = a^T b$ represents the inner product between $a, b \in \mathbb{R}^n$. The interior and boundary of a set $\mathcal{C}$ is denoted by $\text{Int}(C)$ and ∂C. A continuous function $\alpha : [0, a) \to [0, \infty)$ is a class-$\mathcal{K}$ function if it is strictly increasing and $\alpha(0) = 0$. Furthermore, if $a = \infty$ and $\lim_{r\to\infty} \alpha(r) = \infty$, then it is called class-$\mathcal{K}_\infty$. Both $\frac{\partial}{\partial x}$ and ∇_x denote gradient and will be used interchangeably depending on the complexity of expressions for easy understanding.

B. System Description

Consider the following discrete-time dynamics with state $x_t \in \mathbb{R}^n$, control input $u_{x_t} \in \mathbb{R}^m$

$$x_{t+1} = x_t + F(x_t, u_{x_t}) \tag{1}$$

where $F : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is the dynamics function. The following control-affine dynamics form, a special case of (1) will also be referred to later in Section III-C.1

$$x_{t+1} = x_t + f(x_t) + g(x_t)u_{x_t} \tag{2}$$

where $f : \mathbb{R}^n \to \mathbb{R}^n$ and $\mathbb{R}^n \to \mathbb{R}^{n\times m}$ are continuous functions. Let $\phi : \mathbb{R}^n \to \mathbb{R}^p$ project the robot state x_t to its position $\phi(x_t) \in \mathbb{R}^p$, we also denote the robot body as a set valued function $\mathcal{B}(x_t) \subset \mathbb{R}^p$, and its surface as $\partial\mathcal{B}(x_t)$.

The state x_t is required to lie in safe sets $\mathcal{S}_i, i \in \{1, .., N\}$ that are defined to be 0-superlevel sets of a continuously differentiable constraint functions $h_i : \mathbb{R}^n \to \mathbb{R}$

$$\mathcal{S}_i \triangleq \{x \in \mathcal{X} : h_i(x) \geq 0\} \tag{3}$$

$$\partial\mathcal{S}_i \triangleq \{x \in \mathcal{X} : h_i(x) = 0\} \tag{4}$$

$$\text{Int}(\mathcal{S}_i) \triangleq \{x \in \mathcal{X} : h_i(x) > 0\} \tag{5}$$

C. Control Barrier Functions

CBFs are a common tool to enforce safety constraints in a controller. If h_i is a zeroing-barrier function for a class-$\mathcal{K}_\infty$ function ν_i, then the following condition in discrete-time on h_i, called discrete-time CBF condition, is sufficient for the forward invariance of the set $\mathcal{S}_i$:

$$\sup_{u\in\mathbb{R}^m} \left[h_i(F(x_t, u_t)) - h_i(x_t)\right] \geq \nu_i(h_i(x_t)), \forall x \in \mathcal{S}_i \tag{6}$$

In practice, (6) is imposed even if h_i is only a constraint function (a.k.a candidate CBF) and is not guaranteed to satisfy (6) for $x \in \mathcal{S}_i$ as finding a valid CBF is still an ongoing topic of research. For simplicity, we only consider a special case of linear class-$\mathcal{K}$ function $\nu_i(x) = \alpha_i x, \alpha_i \in \mathbb{R}^+$ in the remainder of the paper. Intuitively, the CBF condition puts an upper bound on the rate at which the state x_t is allowed to approach safe set boundary (that is,

it allows $h(x_{t+1}) < h_{x_t}$ when $h(x_t) > 0$ and requires $h(x_{t+1}) \geq h(x_t)$ when $h(x_t) = 0$. The *rate* is described by the parameter α. Owing to the difficulty of finding valid CBF for different performance characteristics of the system, (aggressive or conservative for example) α is usually tuned offline or online.

D. Model Predictive path Integral Control

We provide a brief description of the MPPI algorithm. For a time horizon H, consider the state trajectory $\mathbf{x} = [x_t^T, ..., x_{t+H}^T]^T$, mean control input sequence $\mathbf{v} = [v_t^T, .., v_{t+H}^T]^T, v_\tau \in \mathbb{R}^m$ and injected Gaussian noise $\mathbf{w} = [w_t^T, .., w_{t+H}^T]^T$ where $w_\tau \sim N(0, \Sigma_w)$ where Σ_w is the noise covariance, often chosen by the user. Let the disturbed control input sequence be $\mathbf{u} = [u_t, ..., u_{t+H}] = \mathbf{v} + \boldsymbol{w}$. MPPI then solves the following problem

$$\min_{\mathbf{v}} \quad J(\mathbf{v}) = \mathbb{E}\left[\sum_{\tau=t}^{t+H-1} Q(x_\tau, u_\tau) + \left(\frac{\lambda}{2} v_\tau^T \Sigma_w^{-1} v_\tau\right)\right] \tag{7a}$$

$$\text{s.t.} \quad x_{\tau+1} = F(x_\tau, v_\tau + w_\tau) \tag{7b}$$

$$w_\tau \sim \mathcal{N}(0, \Sigma_w) \tag{7c}$$

where $Q : \mathbb{R}^n \times \mathbb{R}^m$ is a cost function designed to promote user-defined tasks such as convergence to the target state or collision avoidance with an obstacle. In implementation, MPPI adopts a sampling-based approach to solving (7). Given a sample size K, we sample K control input perturbation sequences $\boldsymbol{w}^k, k \in \{1, .., K\}$ where each sequence is of size H, that is $\boldsymbol{w}^k = [w_t^k, ..., w_{t+H}^k]$. The control input sequence is then computed as $\mathbf{u}^k = \mathbf{v} + \boldsymbol{w}^k$. For each sequence k, we compute the states $x_\tau^k, \tau \in [t, .., t+H]$ using dynamics (1).

The cost S_k of each trajectory $k \in \{1, .., K\}$ is evaluated based on (7a) as follows

$$S_k = \Phi(x_{t+H}^k) \sum_{\tau=t}^{t+H-1} Q(x_\tau^k) + \gamma (v_\tau^k)^T \Sigma_g^{-1} (v_\tau^k + g_\tau^k)$$

where $\gamma \in [0, \lambda]$. The weight w_k for k^{th} trajectory is determined as

$$w_k = \exp\left(-\frac{1}{\lambda}(S_k - \beta)\right) \tag{8}$$

where $\beta = \min_{k \in \{1,..,K\}} S_k$. The optimal control sequence is then computed as

$$\mathbf{v}^+ = \sum_{k=1}^{K} w_k \mathbf{u}^k / \sum_{k=1}^{K} w_k. \tag{9}$$

The cost functions are generally user-designed for specific objectives. For instance, we introduce cost metrics here to quantify progress toward achieving two types of user-specified tasks: convergence to a goal and collision avoidance.

E. Signed Distance Function

Signed distance functions (SDFs) are widely used in robotics for accurate modeling of environmental geometries [16], [17], [18]. Following recent developments [19], [20], [21], we employ SDFs to model the robot's own shape. This approach offers enhanced expressiveness for complex robot geometries and facilitates faster computations of distances to the environments with point-cloud observations. In particular, for a robot body $\mathcal{B}$, we define the robot's SDF $\varphi : \mathbb{R}^p \mapsto \mathbb{R}$:

$$\varphi(\mathbf{p}) := \begin{cases} -d(\mathbf{p}, \partial\mathcal{B}), & \mathbf{p} \in \mathcal{B}, \\ d(\mathbf{p}, \partial\mathcal{B}), & \mathbf{p} \notin \mathcal{B}, \end{cases} \tag{10}$$

where d denotes the Euclidean distance between a point and a set, and $\mathbf{p} \in \mathbb{R}^p$ is a workspace point.

III. Methodology

We discuss our approach to training the robot SDF model.

A. Training the Robot SDF Model

To accurately model the robot's shape using an SDF, we begin by preparing a training dataset $\mathcal{D}$ derived from the robot's geometric description (e.g., URDF). This dataset consists of workspace points and their associated signed distances to the robot body. Specifically, the dataset includes samples of the form $(\mathbf{p}, d)$, where $\mathbf{p} \in \mathbb{R}^p$ is a point in the workspace and d is the signed distance from $\mathbf{p}$ to the robot's surface $\partial\mathcal{B}$.

We define a loss function to train the SDF model $\hat{\varphi}(\mathbf{p}; \boldsymbol{\theta})$, parameterized by $\boldsymbol{\theta}$, as follows:

$$\ell(\boldsymbol{\theta}; \mathcal{D}) = \ell^D(\boldsymbol{\theta}; \mathcal{D}) + \lambda_E \, \ell^E(\boldsymbol{\theta}; \mathcal{D}), \tag{11}$$

where ℓ^D is the distance loss, ℓ^E is the Eikonal loss, and λ_E is a tunable parameter that balances the two terms.

The distance loss ℓ_i^D measures the mean squared error between the predicted signed distances and the ground truth distances $\ell^D(\boldsymbol{\theta}; \mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{\mathbf{p},d) \in \mathcal{D}} \left(\hat{\varphi}(\mathbf{p}; \boldsymbol{\theta}_i) - d\right)^2$.

The Eikonal loss ℓ^E enforces the property that the gradient of the SDF with respect to the input point $\mathbf{p}$ has unit norm almost everywhere, ensuring that the learned SDF behaves like a true distance function:

$$\ell^E(\boldsymbol{\theta}; \mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{q},\mathbf{p}) \in \mathcal{D}} \left(\|\nabla_{\mathbf{p}} \hat{\varphi}(\mathbf{p}; \boldsymbol{\theta})\| - 1\right)^2. \tag{12}$$

By minimizing the combined loss function in (11), we train the SDF model to accurately represent the robot's geometry.

B. Safe MPPI Formulation

In this work, instead of trying to impose the inequality constraint (6), we impose the following equality constraint

$$h_i(F(x_t, u_{x_t})) - h_i(x_t) = -\alpha_{i,t} h_i(x_t), \forall i \in \{1, .., N\} \tag{13}$$

and allow the parameter α_i to change with time (hence the notation $\alpha_{i,t}$ in (13)). We achieve this by introducing a pseudo-parameter state $\tilde{\alpha}_{i,t}$ and and additional control inputs $u_{\tilde{\alpha}_{i,t}}$ in the following augmented state-space system

$$\underbrace{\begin{bmatrix} x_{t+1} \\ \tilde{\alpha}_{t+1} \end{bmatrix}}_{z_{t+1}} = \underbrace{\begin{bmatrix} x_t \\ \tilde{\alpha}_t \end{bmatrix}}_{z_t} + \begin{bmatrix} F(x_t, u_{x_t}) \\ u_{\tilde{\alpha}_{1,t}} \end{bmatrix} \tag{14}$$

where $z_t = [x_t^T \ \tilde{\alpha}_t^T] \in \mathbb{R}^{n+N}$ with $\tilde{\alpha}_t^T = [\tilde{\alpha}_{1,t}, .., \tilde{\alpha}_{N,t}]^T$ and $u_t = [u_{x_t}^T \ u_{\tilde{\alpha}_t}^T]^T, u_{\tilde{\alpha}_t} = [u_{\tilde{\alpha}_{1,t}}, .., u_{\tilde{\alpha}_{N,t}}]^T$ will be referred to as the augmented state and control input. $\tilde{\alpha}_t$ will be used in Section III-C to compute the α_t that will be imposed in the equality constraint (13) at time t. Specifically, we will design parameter dynamics such that $\alpha_t = \tilde{\alpha}_{t+1}$.

Note that we allow $\tilde{\alpha}_t$ and $\tilde{\alpha}$ to vary in $\mathbb{R}^N$, that is, $\alpha_{i,t}$ can attain positive as well as negative values which is necessary for the robot to move away (for collision avoidance) or towards the obstacle (in transient phase while making progress towards goal state). Later in Section III-D, we introduce a cost function that promotes constraint satisfaction.

Remark 1. We introduce $\tilde{\alpha}_t$, instead of α_t, as the new state variable for the following reason. α_t along with u_t must be such that they satisfy (13) for $i \in \{1, .., N\}$ simultaneously. The compatibility cannot be enforced if α_t were a state variable that is guided by random control inputs sampled by MPPI. In the ensuing, we provide more details of our procedure to ensure compatibility. Also, note that α_t cannot be treated as control input (instead of a state variable) for the same reason specified above. Compatibility between (13) for $i \in \{1, .., N\}$ would impose constraints on the choice of u_t and α_t thereby precluding random Gaussian sampling that is inherent to MPPI.

Remark 2. The notion of designing dynamics for parameter α is similar to existing works that design $\dot{\alpha}$ in discrete-time [11] and continuous-time [12], [13].

Before presenting our final algorithm, we introduce another control space transformation. Let $u'_{x_t} \in \mathbb{R}^m$ and $u'_{\tilde{\alpha}_t} \in \mathbb{R}^N$ be pseudo-inputs that are randomly sampled by MPPI and u_{x_t} and $u_{\tilde{\alpha}}$ are obtained from pseudo-inputs through a projection operation $\mathbb{P} : \mathbb{R}^{n+N} \times \mathbb{R}^m \times \mathbb{R}^N \to \mathbb{R}^m, \mathbb{R}^N$ introduced in the next section. The dynamics under consideration for MPPI is thus the following

$$\begin{bmatrix} x_{t+1} \\ \tilde{\alpha}_{t+1} \end{bmatrix} = \begin{bmatrix} x_t \\ \tilde{\alpha}_t \end{bmatrix} + \begin{bmatrix} F(x_t, u_{x_t}) \\ u_{\tilde{\alpha}_{1,t}} \end{bmatrix}, \tag{15a}$$

$$u_{x_t}, u_{\tilde{\alpha}_t} = \mathbb{P}(z_t, u'_{x_t}, u'_{\alpha_t}) \tag{15b}$$

The projection operator $\mathbb{P}$, as will become clear, will project the randomly sampled control inputs to the set of allowed manifolds described by equality constraints (13)

C. Design of the Projection function

To reiterate, the system of equations (13) might not have a solution for arbitrary values of u_{x_t} and α_t. Therefore the objective of designing $\mathbb{P}_{u,\alpha}$ is to ensure compatibility of (13) for $i \in \{1, .., N\}$.

Let $u'_{x,t}$ and $u'_{\tilde{\alpha},t-1}$ be the randomly sampled control inputs from MPPI. We project them to satisfy the manifold constraints with the following operation

$$\mathbb{P}(z_t, u'_{x_t}, u'_{\tilde{\alpha}_t}) = \tag{16a}$$

$$\arg\min_{v,a} \quad ||v - u'_{x_t}||_{Q_1} + ||a - u'_{\tilde{\alpha}_t}||_{Q_2} \tag{16b}$$

$$\text{s.t.} \quad h_i(F(x_t, v)) - h_i(x_t) = -(\tilde{\alpha}_{i,t} + a_i) h_i(x_t) \quad \forall i \in \{1, .., N\} \tag{16c}$$

Note that $\tilde{\alpha}_{i,t} + a_i = \tilde{\alpha}_{i,t+1}$ and therefore, by design of dynamics in (15b), $\tilde{\alpha}_{t+1} = \alpha_t$.

1) Simplification for Control Affine dynamics: The optimization (16) might be too expensive to solve for generic dynamics function F. In the ensuing, we limit our discussions and implementations to the special case of control affine dynamics in (2) under which (16) accepts an analytical solution. An extension to generic F is left for future work. Under (2) and first-order Taylor series expansion of $h(f(x_t)+g(x_t)u_t)$ about $h(x_t)$, we get

$$h(f(x_t) + g(x_t)u_t) - h(x_t) = \frac{\partial h}{\partial x}\Big|_{x_t} (f(x_t) + g(x_t)u_t) \tag{17}$$

and thus the constraints in (16) are linear in u. The projection thus reduces to the following minimum norm problem subject to equality constraints.

$$\mathbb{P}(z_t, u'_{x_t}, u'_{\tilde{\alpha}_t}) = \tag{18a}$$

$$\arg\min_{v,a} \quad (z - z_{des})^T W (z - z_{des}) \tag{18b}$$

$$\text{s.t.} \quad A_t z = b_t \tag{18c}$$

where

$$z = \begin{bmatrix} v \\ a \end{bmatrix}, \ z_{des} = \begin{bmatrix} u'_{x,t} \\ u'_{\tilde{\alpha}_t} \end{bmatrix}, \ W = diag(Q_1, Q_2)$$

$$A = \begin{bmatrix} \frac{\partial h_1}{\partial x}\Big|_{x_t} g(x_t) & h_1(x_t) \\ \vdots & \\ \frac{\partial h_N}{\partial x}\Big|_{x_t} g(x_t) & h_N(x_t) \end{bmatrix}, \ b = \begin{bmatrix} -\frac{\partial h_1}{\partial x}\Big|_{x_t} f(x_t) \\ \vdots \\ -\frac{\partial h_N}{\partial x}\Big|_{x_t} f(x_t) \end{bmatrix} \tag{19}$$

Equation (18c) is simply an instance of an equality-constrained weighted minimum norm problem[**?**] the solution is given by

$$z = z_{des} + W^{-1} A^T (A W^{-1} A^T)(b - A z_{des}) \tag{20}$$

Remark 3. Sampling inputs while trying to satisfy equality constraints is actively being studied for path planning on manifolds[22]. Several systems for closed-chain systems that inherently require a state constraint to be satisfied fall under this paradigm. Our projection operation is also inspired by such works as each equality constraint in (13) induces a manifold on which the state-action pair should lie. We therefore also share some issues inherent in the discretization of manifold constraints. Our simplification in (18c), although leads to good results in practice, induces numerical errors due to first-order Taylor series expansion of constraint function h. These induced errors may pose problems for highly nonlinear constraint functions and lead to instability of our algorithm. We could keep recomputing tangent space of h_i whenever

discretization errors get larger than a threshold as done in [23] for kinodynamic RRT but such an analysis, and its applicability to real-time control, is left for future work. Note that the dynamics constraint (1) is itself a manifold that is inherently followed by MPPI trajectories. However, to the best of our knowledge, our work is the first to consider multiple state and input-dependent equality constraints in MPPI.

D. Design of MPPI Cost

We decompose the MPPI stage-wise cost Q into two terms Q_c and Q_h. Q_c is designed to promote convergence to the desired state and Q_h is designed to ensure constraint satisfaction. Q_c is designed by the user. In the following, we specify the design of Q_h.

A necessary condition for constraint maintenance is given by Nagumo's theorem[24], [25] and states that whenever $h_i(x_t) = 0$, $\dot{h}_i(x_t) \geq 0$ or equivalently in discrete time $h(x_{t+1}) - h(x_t) = -\alpha_t h(x_t) \geq 0$. Owing to the discrete-time nature of our algorithm, instead of just accounting for constraint set boundary, we consider a buffer zone D_i with buffer length d_i,

$$D_i = \{x \mid 0 \leq h_i(x) \leq d_i\} \tag{21}$$

Inside this buffer zone, we impose the condition that $\alpha_t < 0$ in (13) to promote positive $\dot{h}_i$. The buffer length d_i is a tuning parameter in practice. Note that since $\alpha_{i,t} \in [-\infty, \infty]$), we explore all possible ways in which the robot can navigate around an obstacle, and invoke Nagumo's theorem only near the boundary to ensure constraint satisfaction.

$$Q_h(z_t, z_{t+1}, u) = \sum_{i=1}^{N} \mathbb{1}(x_t \in D_i \cap x_t \in \bar{\mathcal{S}}_i) \frac{\tilde{\alpha}_{i,t+1}}{h_i(x)} \tag{22}$$

The whole algorithm is summarized in Algorithm 1.

E. Discussion

While applying MPPI on augmented system (14) might be posed as simply a state transformation, we find it to have far-reaching exciting consequences. Here we discuss the improvements over and differences with respect to standard MPPI and CBF approaches.

1) Improvement of MPPI through CBF: First, we use the well established theory of CBFs and set invariance to guide our algorithm in imposing inequality constraints. In essence, our algorithm realizes inequality constraint through two novel approaches: 1) replace inequality constraint with a *set* of equality constraints where the set is specified by the additional state variable $\tilde{\alpha}$, 2) Use the theory of planning on manifolds to impose the equality constraint. Note that this only shifts the burden to appropriately imposing the equality constraints but we realize it through Nagumo's theorem.

Second, we observe stark differences in the nature of sampled trajectories in our approach compared to vanilla MPPI. As seen in Fig. 1, our trajectories can be multimodal resulting in the seemingly disconnected fan of trajectories skipping over the obstacles. We hypothesize the reason being that sampling unimodal trajectories in a higher dimensional system given by (15b) results in multi-modal trajectories in the original lower-dimensional system given by (1). Indeed, in the context of Fig. 1, the trajectories on either side of an obstacle have similar $h_i(\approx 0)$ and $\dot{h}_i$ and therefore are continuous as far as constraint violation is concerned. Since $\tilde{\alpha}$ is related to h, the trajectories are unimodal in the augmented state space. Note that in theory, our algorithm may still have trajectories sampled inside the obstacle. However, it seems to be better at diverging away from such solutions.

2) Improvement of CBF through MPPI: Finding a valid CBF is an active area of research and it is common practice to use candidate CBFs when imposing (6) in CBF-QP controllers instead. While a candidate CBF is not guaranteed to have a feasible solution to CBF-QP for all states, even a valid CBF, although gives feasible solutions, might be too conservative for a given class-$\mathcal{K}$ function. Several works adapt the class-$\mathcal{K}$ function offline or online. Our proposed algorithm could be considered as performing an online adaptation of class-$\mathcal{K}$ to improve performance while still ensuring safety. Moreover, although we use CBF-like conditions, we enforce constraint maintenance only by invoking Nagumo's theorem near the boundary. In the interior of the safe set, we allow $\tilde{\alpha}$ to vary in $\mathbb{R}$. Therefore, our method is less restrictive than using a CBF with a fixed class-$\mathcal{K}$ function. A more thorough analysis of the relationship between solutions found by our proposed MPPI and existing valid CBFs will be subject to future work.

Algorithm 1 BR-MPPI Controller

Require: z_t, H, F, Q_c ▷ current augmented state, time horizon, Dynamics function, user-given convergence cost

Sample $\boldsymbol{w}^k = [w_1^k, .., w_H^k] \sim \mathcal{N}(0, \Sigma_\epsilon), \forall k \in \{1, .., K\}$
$\mathbf{u}^k \leftarrow \mathbf{v} + \boldsymbol{\epsilon}^k$
for $k = 1$ to K **do** ▷ Loop over samples
  $S_k = 0$ ▷ Initialize Cost
  for $\tau = 0$ to H **do** ▷ Loop over time horizon
    $u'_{x_t}, u'_{\tilde{\alpha}_t} \leftarrow \boldsymbol{u}_\tau^k$
    Compute $u_{x_t}, u_{\tilde{\alpha}_t}$ using $\mathbb{P}$ in (16) or (20)
    Compute next state $z_{t+\tau+1}$ using (15b)
    Compute Q_h using (22)
    $S_k \leftarrow S_k + Q_c(x_{t+\tau}^k) + Q_h + \gamma(v_{t+\tau}^m)^T \Sigma_\epsilon^{-1}(u_{t+\tau}^m)$
  end for
end for
$w^m \leftarrow$ (8) ▷ Compute weights
$\mathbf{v}^+ \leftarrow$(9) **return** $\mathbf{v}^+$ ▷ Computed weighted average control trajectory

IV. Simulation Results

Videos for all the results in Section IV and V including hardware experiments can be accessed at https://youtube.com/playlist?list=PLJxod9x5m_V1o5wq-DWdUMaVKUAECyOoI&si=7IVb1emEMKVJaaS9.

We use Algorithm 1 to navigate the robot in an obstacle environment shown. The robot follows one of the following

three dynamics models: single integrator (SI), double integrator (DI), and extended unicycle (EU).

$$\text{SI: } \dot{p}_x = u_1,\ \dot{p}_y = u_2 \tag{23}$$

$$\text{DI: } \ddot{p}_x = u_1,\ \ddot{p}_y = u_2 \tag{24}$$

$$\text{EU: } \dot{p}_x = v\cos\theta,\ \dot{p}_y = v\sin\theta,\ \dot{\theta} = u_1,\ \dot{v} = u_2 \tag{25}$$

where $p_x, p_y \in \mathbb{R}$ are positions, $v \in \mathbb{R}$ is forward velocity, θ is the heading direction and u_1, u_2 are the control inputs respectively. We further compare the vanilla MPPI and our algorithm for its capability to navigate for different robot shapes without changing parameters involved in MPPI (that is, without fine-tuning MPPI for every new robot shape, and dynamics). Fig. 2 shows snapshots of paths resulting from vanilla and proposed MPPI for a point mass EU robot. We observe that the proposed algorithm is able to pass through narrow spaces whereas vanilla MPPI takes a detour. Increasing samples of vanilla MPPI to 20,000 prompts it to find similar paths as our algorithm thereby showing that our algorithm has better sample efficiency in this case study. Next, we use both algorithms for a hexagonal-shaped SI robot and visualize the resulting paths in Fig. 3. We observe that vanilla MPPI is too conservative, and the robot is stuck close to the initial position. Our algorithm, on the other hand can find a path leading to the goal. Simulation videos for different robot dynamics can be found on our YouTube playlist.

V. Experiment Results

We demonstrate our algorithm for the navigation of a quadrotor transporting an object in the presence of obstacles. The robot platform has a rectangular footprint of size 0.28 x 0.56 m for which we learn the signed distance filed as detailed in Section III-A. The quadrotor follows the following double integrator dynamics in position and heading angle.

$$\ddot{p}_x = u_1, \ddot{p}_x = u_2, \ddot{\theta} = u_3 \tag{26}$$

where u_1, u_2 are linear acceleration and u_3 is angular acceleration control input. The experiment setup is shown in Fig. 4, and the objective for the robot is to reach the goal location while avoiding obstacles. The quadrotor is fitted with metal straws to render it a rectangular footprint. The snapshots of a simulated run of the experiment with BR-MPPI are shown in Fig. 5. The vanilla MPPI is unable to enter the corridor, and the snapshots are skipped due to lack of space. Each obstacle is converted into a point cloud with 30 points. We use the same parameters as in Section IV and observe that in this highly constrained environment, the vanilla MPPI fails to finish the task, whereas BR-MPPI finds a path. A hardware evaluation of the algorithm is also performed, and video is included in the YouTube playlist.

We reiterate that by including α as a state in the system and designing $\dot{\alpha}$, we are forcing the system to follow smoother trajectories with respect to how they move toward to away from obstacles (constraint boundaries). This leads our algorithm to better explore the space close to obstacles. A more principled theoretical analysis of the differences that result due to our reformulation of vanilla MPPI to augmented state space will be explored in future work.

VI. Conclusion and Future Work

We presented a new algorithm that uses CBF-like conditions to guide the trajectory sampling procedure. Some of the future extensions of our work include the incorporation of higher-order CBFs where control input does not appear in the first derivative of the barrier function and the derivation of guarantees of inequality constraint maintenance. We will also perform hardware evaluation on a variety of systems.

References

[1] 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*, vol. 34, no. 6, pp. 1603–1622, 2018.

[2] 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*, vol. 62, no. 8, pp. 3861–3876, 2016.

[3] C. Tao, H. Kim, H. Yoon, N. Hovakimyan, and P. Voulgaris, "Control barrier function augmentation in sampling-based control algorithm for sample efficiency," in *2022 American Control Conference (ACC)*. IEEE, 2022, pp. 3488–3493.

[4] M. S. Gandhi, B. Vlahov, J. Gibson, G. Williams, and E. A. Theodorou, "Robust model predictive path integral control: Analysis and performance guarantees," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1423–1430, 2021.

[5] J. Yin, Z. Zhang, E. Theodorou, and P. Tsiotras, "Trajectory distribution control for model predictive path integral control using covariance steering," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 1478–1484.

[6] E. Trevisan and J. Alonso-Mora, "Biased-mppi: Informing sampling-based model predictive control by fusing ancillary controllers," *IEEE Robotics and Automation Letters*, 2024.

[7] J. Carius, R. Ranftl, F. Farshidian, and M. Hutter, "Constrained stochastic optimal control with learned importance sampling: A path integral approach," *The International Journal of Robotics Research*, vol. 41, no. 2, pp. 189–209, 2022.

[8] 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 Robotics and Automation Letters*, 2023.

[9] P. Tsiotras, K. Berntorp, *et al.*, "Chance-constrained information-theoretic stochastic model predictive control with safety shielding," *arXiv preprint arXiv:2408.00494*, 2024.

[10] M. Gandhi, H. Almubarak, Y. Aoyama, and E. Theodorou, "Safety in augmented importance sampling: Performance bounds for robust mppi," *arXiv preprint arXiv:2204.05963*, 2022.

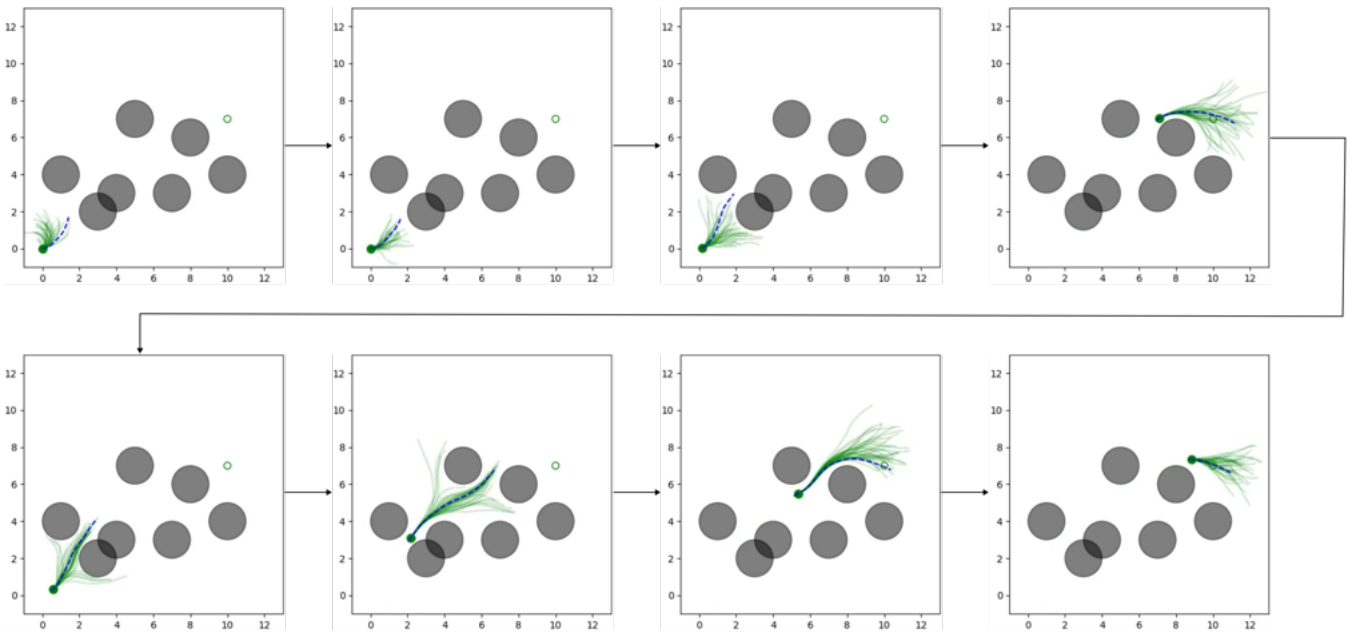
[11] H. Parwana and D. Panagou, "Recursive feasibility guided optimal parameter adaptation of differential convex optimization policies for safety-critical systems," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 6807–6813.

[12] H. Parwana, A. Mustafa, and D. Panagou, "Trust-based rate-tunable control barrier functions for non-cooperative multi-agent systems," in *2022 IEEE 61st Conference on Decision and Control (CDC)*. IEEE, 2022, pp. 2222–2229.

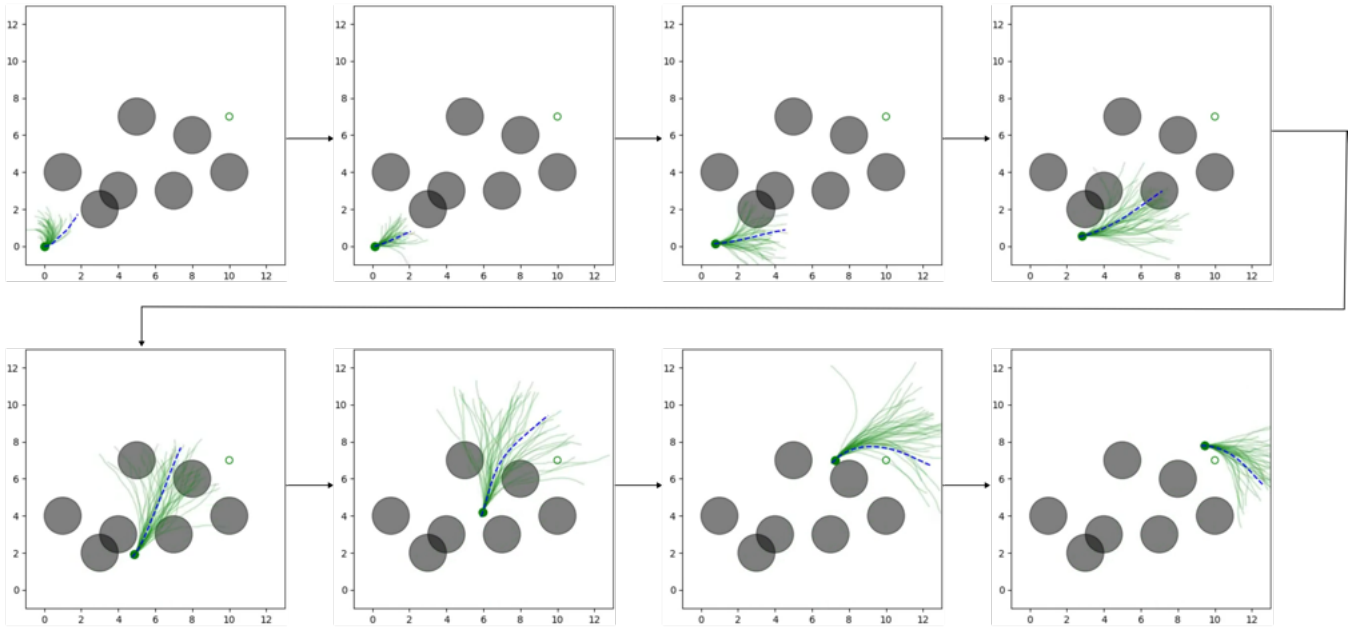
[13] W. Xiao, C. Belta, and C. G. Cassandras, "Adaptive control barrier functions," *IEEE Transactions on Automatic Control*, vol. 67, no. 5, pp. 2267–2281, 2021.

[14] J. Zeng, B. Zhang, Z. Li, and K. Sreenath, "Safety-critical control using optimal-decay control barrier function with guaranteed point-wise feasibility," in *2021 American Control Conference (ACC)*. IEEE, 2021, pp. 3856–3863.

[15] H. Ma, B. Zhang, M. Tomizuka, and K. Sreenath, "Learning differentiable safety-critical control using control barrier functions for generalization to novel environments," in *2022 European Control Conference (ECC)*. IEEE, 2022, pp. 1301–1308.



(a) Proposed Method: BR-MPPI



(b) Vanilla MPPI

Fig. 2: Extended Unicycle point mass robot navigating in an obstacle environment.

[16] L. Han, F. Gao, B. Zhou, and S. Shen, "Fiesta: Fast incremental Euclidean distance fields for online motion planning of aerial robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[17] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3d Euclidean signed distance fields for on-board mav planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 1366–1373.

[18] K. Long, C. Qian, J. Cortés, and N. Atanasov, "Learning barrier functions with memory for robust safe navigation," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4931–4938, 2021.

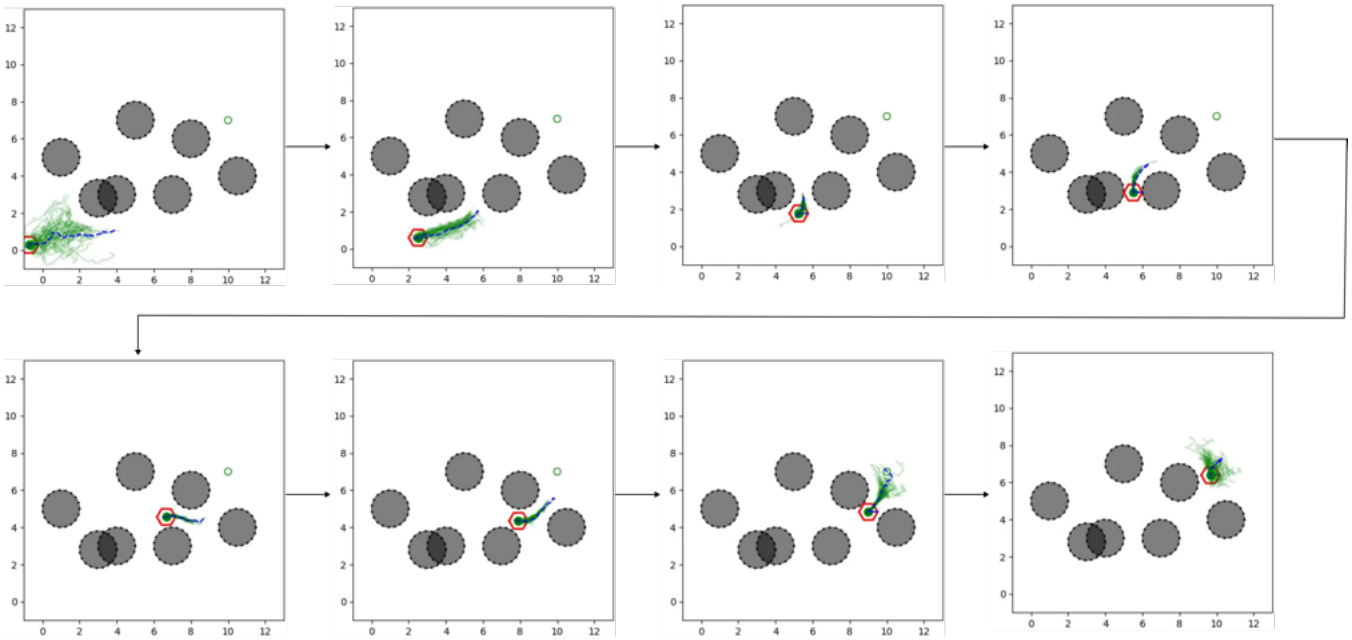
[19] K. Long, Y. Yi, Z. Dai, S. Herbert, J. Cortés, and N. Atanasov, "Sensor-based distributionally robust control for safe robot navigation in dynamic environments," *arXiv preprint arXiv:2405.18251*, 2024.

[20] Y. Li, Y. Zhang, A. Razmjoo, and S. Calinon, "Representing robot geometry as distance fields: Applications to whole-body manipulation," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, 2024, pp. 15 351–15 357.

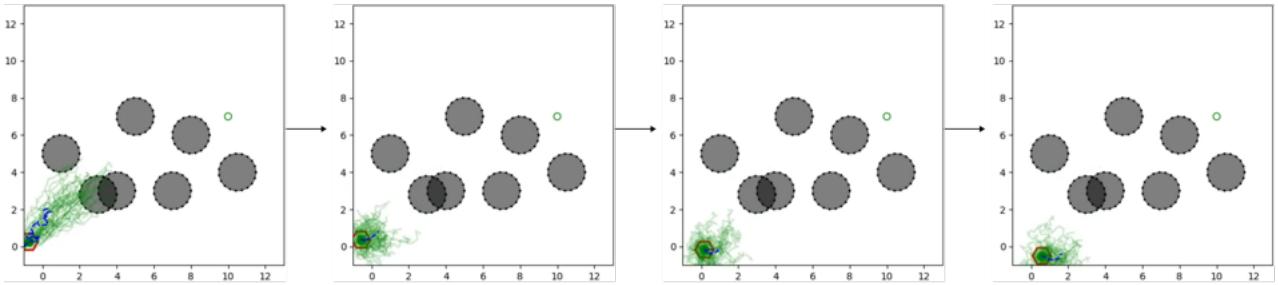
[21] K. Long, H. Parwana, G. Fainekos, B. Hoxha, H. Okamoto, and N. Atanasov, "Neural configuration distance function for continuum robot control," 2024. [Online]. Available: https://arxiv.org/abs/2409.13865

[22] Z. Kingston, M. Moll, and L. E. Kavraki, "Sampling-based methods for motion planning with constraints," *Annual review of control, robotics, and autonomous systems*, vol. 1, no. 1, pp. 159–185, 2018.

[23] R. Bordalba, L. Ros, and J. M. Porta, "A randomized kinodynamic



(a) Proposed Method: BR-MPPI



(b) Vanilla MPPI

Fig. 3: Hexagonal Single Integrator robot navigating in an obstacle environment.

planner for closed-chain robotic systems," *IEEE Transactions on Robotics*, vol. 37, no. 1, pp. 99–115, 2020.

[24] M. Nagumo, "Über die lage der integralkurven gewöhnlicher differentialgleichungen," *Proceedings of the physico-mathematical society of Japan. 3rd Series*, vol. 24, pp. 551–559, 1942.

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

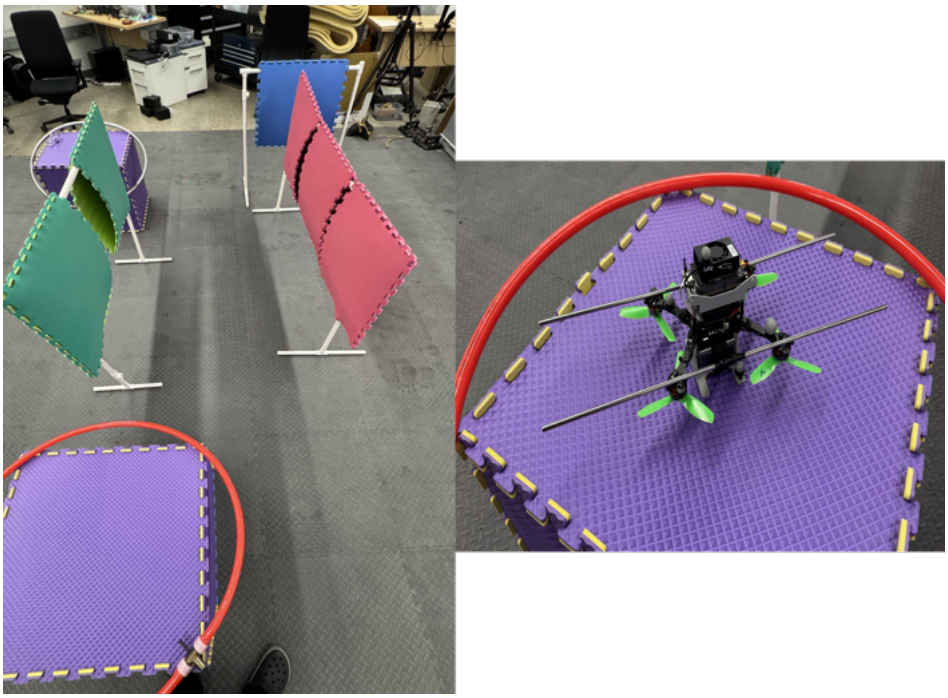


Fig. 4: Experiment Setup. The left figure shows the obstacle environment, and the right figure shows our custom-assembled quad.

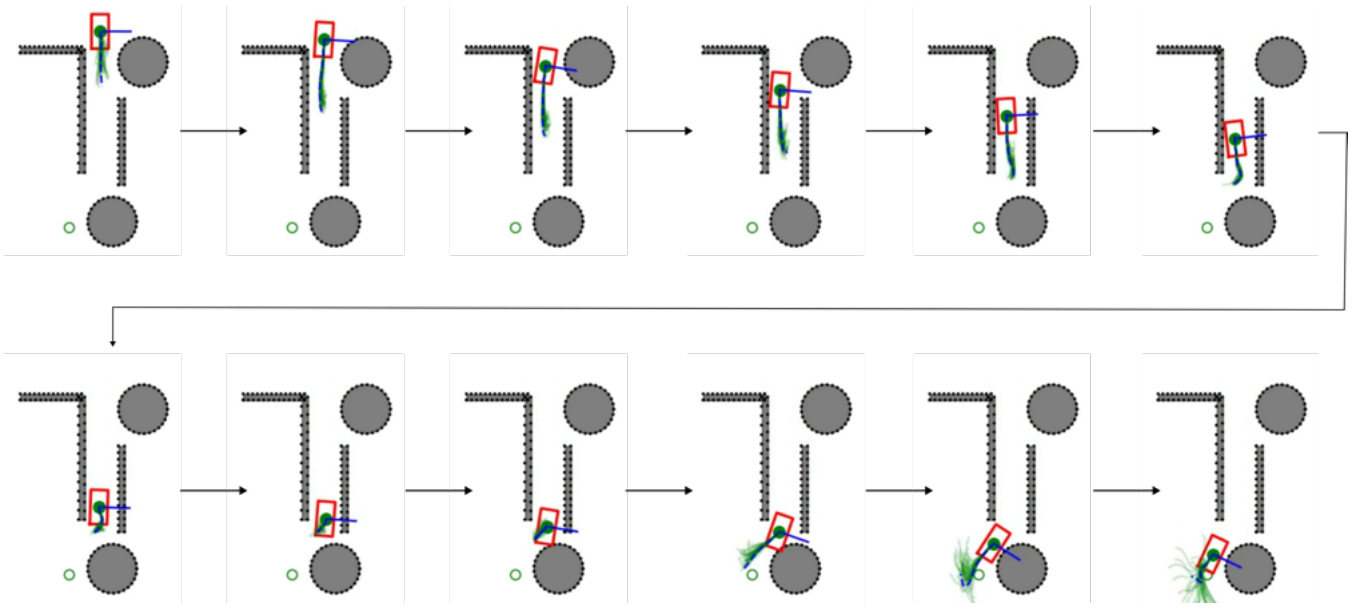


Fig. 5: A simulated run of the experiment setup.