# PA-MPPI: Perception-Aware Model Predictive Path Integral Control for Quadrotor Navigation in Unknown Environments

Yifan Zhai, Rudolf Reiter, Davide Scaramuzza

***Abstract*—Quadrotor navigation in unknown environments is critical for practical missions such as search-and-rescue. Solving it requires addressing three key challenges: the non-convexity of free space due to obstacles, quadrotor-specific dynamics and objectives, and the need for exploration of unknown regions to find a path to the goal. Recently, the Model Predictive Path Integral (MPPI) method has emerged as a promising solution that solves the first two challenges. By leveraging sampling-based optimization, it can effectively handle non-convex free space while directly optimizing over the full quadrotor dynamics, enabling the inclusion of quadrotor-specific costs such as energy consumption. However, its performance in unknown environments is limited, as it lacks the ability to explore unknown regions when blocked by large obstacles. To solve this issue, we introduce Perception-Aware MPPI (PA-MPPI). Here, perception-awareness is defined as adapting the trajectory online based on perception objectives. Specifically, when the goal is occluded, PA-MPPI's perception cost biases trajectories that can perceive unknown regions. This expands the mapped traversable space and increases the likelihood of finding alternative paths to the goal. Through hardware experiments, we demonstrate that PA-MPPI, running at 50 Hz with our efficient perception and mapping module, performs up to 100% better than the baseline in our challenging settings where the state-of-the-art MPPI fails. In addition, we demonstrate that PA-MPPI can be used as a safe and robust action policy for navigation foundation models, which often provide goal poses that are not directly reachable.**

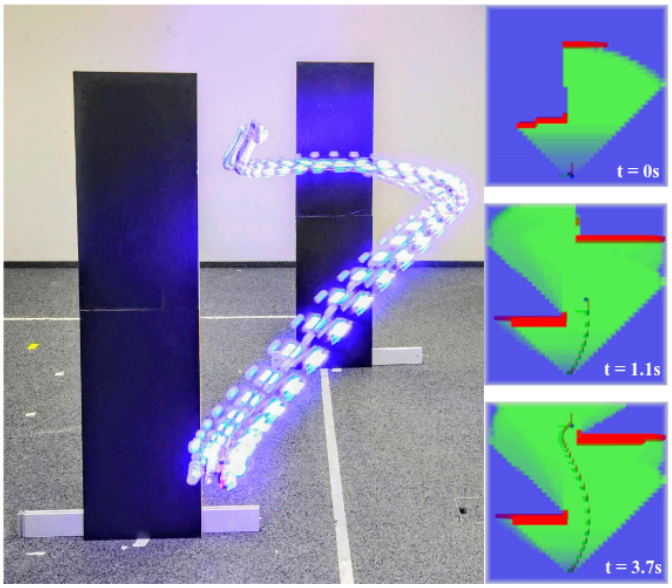



Fig. 1: The Perception-Aware MPPI (PA-MPPI) controller navigating to the goal pose while avoiding obstacles in a previously unknown environment. The controller simultaneously controls the quadrotor at 50 Hz and optimizes the perception objective based on an online-updated 3D map constructed from onboard observations.

## I. Introduction

Enabling quadrotors to autonomously navigate to a goal in unknown environments is critical for practical missions such as search-and-rescue, infrastructure inspection, and exploration [1]–[3]. It is also relevant for the safe deployment of navigation foundation models, which provide navigation waypoints or goals in previously unseen environments [4], [5]. Achieving this capability requires addressing three key challenges: (i) non-convex constraints: cluttered environments create non-convex free space, which complicates gradient-based optimization; (ii) quadrotor-specific dynamics and costs: planned trajectories must satisfy dynamic feasibility while optimizing costs such as effort and energy; and (iii) mapping in unknown environments: since the environment must be mapped online using onboard perception, successful navigation must incorporate exploration and mapping of unknown regions to find a feasible path to the goal.

A common approach is a hierarchical planner–controller architecture [6]–[9], where a global planner computes trajectories subsequently tracked by a local controller. However, this separation often results in conservative or dynamically infeasible plans, as quadrotor-specific dynamics and costs are not considered during the planning stage [10].

Model Predictive Path Integral (MPPI) control has recently emerged as a promising alternative. By employing sampling-based optimization, MPPI can navigate highly non-convex and nonsmooth free space in the presence of obstacles while directly optimizing control inputs on quadrotor-specific dynamics and costs [11], [12]. Nevertheless, the application of MPPI in navigating complex, unknown environments remains unexplored. Our experiments show that standard MPPI tends to get stuck at large obstacles and fails to reach the goal, as it fails to expand traversable regions by actively exploring and mapping unknown areas.

In this work, we introduce the Perception-Aware Model Predictive Path Integral (PA-MPPI) controller, integrated with a custom perception and mapping module, to enable standard MPPI with the capability of exploring unknown regions without direct path to the goal. This is achieved by extending standard MPPI with perception awareness, characterized by adapting control inputs and trajectory optimization based on both current and future perception of the environment. Prior works have demonstrated the effectiveness of perception-aware costs for maintaining visual targets within the onboard

The authors are with the Robotics and Perception Group, University of Zurich Switzerland (http://rpg.ifi.uzh.ch). Contact: dzhai@ifi.uzh.ch

This work was supported by the European Union's Horizon Europe Research and Innovation Programme under grant agreement No. 101120732 (AUTOASSESS) and the European Research Council (ERC) under grant agreement No. 864042 (AGILEFLIGHT).

camera's field of view [2], [13], [14]. PA-MPPI expands on this by introducing an additional perception cost. Using the current map of the environment, this cost evaluates sampled trajectories based on their potential to perceive unknown regions in the goal direction, thereby guiding the optimized trajectory to map unknown regions that lead to the goal.

We validate PA-MPPI in real-world experiments, demonstrating that perception-awareness allows MPPI to navigate through complex, unknown environments where standard MPPI would otherwise get stuck. We show that PA-MPPI successfully addresses all three challenges necessary for autonomous quadrotor goal reaching in unknown settings.

Our contributions are threefold:

- **Novel Perception-Aware Cost for MPPI:** We propose a cost function that exploits the current environment map to guide trajectory optimization toward frontiers that help perceive unknown regions towards the goal, improving navigation success in challenging unknown environments.
- **Integrated Framework:** We present the first perception-aware MPPI framework that simultaneously combines sensing, mapping, and a high-performance MPPI implementation running at 50 Hz for real-time quadrotor control.
- **Hardware-in-the-Loop Validation:** Our experiments show that PA-MPPI outperforms the standard MPPI controller, achieving up to 100% more successful navigations in environments where baseline MPPI fails. We further demonstrate that PA-MPPI enhances the robustness of navigation foundation models when used as action policies.

## II. Related Work

In the following, we group related literature into three sub-sections: (i) navigation with obstacle avoidance in cluttered environments with an emphasis on optimization-based formulations, (ii) perception-aware navigation under sensing and estimation limits, and (iii) algorithms for solving model predictive control (MPC) problems in real time, with a focus on sampling-based MPC.

**Obstacle Avoidance in Cluttered Environments.** In classical software architectures, a high-level planner is responsible for more complex objectives and generates a safe path or trajectory, which is then passed to a fast lower-level controller [10]. Often, planners operate in a discrete or low-dimensional control/state space, for example, using graph search [6]–[8] or rapidly exploring random trees [9]. However, with a reduced state or control space, trajectories may be suboptimal, and the hierarchical decomposition introduces an interaction between the controller and the planner.

Optimization-based approaches often simultaneously plan safe trajectories and control the actuators. However, optimization-based motion planning in the presence of obstacles is challenging due to the induced nonconvexity and nonsmoothness [15]–[19]. A recent approach [17] and, similarly [16], exploit graphs of convex sets to cast collision-free planning as a tight convex program, scaling to high-dimensional environments [17]. The authors in [16], [18], [19] propose efficient algorithms for ellipsoidal or cubic obstacles, and [1] shows a real-world implementation for Unmanned Areal Vehicles (UAVs). However, such derivative-based formulations typically rely on smooth functions, which can be difficult to obtain directly from raw depth observations in complex, highly non-smooth environments.

Beyond optimization-based approaches, learned depth-camera-based policies have shown sufficient navigation performance when trained end-to-end from visual inputs. The authors in [20] demonstrated high-speed flight in real-world environments using policies trained in simulation. More recently, vision transformer-based policies have been explored for end-to-end quadrotor obstacle avoidance using onboard computation [21]. While such approaches improve the closed-loop cost disregarding collisions, they often lack the safety-relevant explicit constraint handling and adaptability to new environments provided by optimization-based methods.

**Perception-Aware Navigation.** A growing number of works integrates perception objectives directly into planning and control, either to acquire task-relevant information or to maintain estimation quality. Perception-aware Model Predictive Control (MPC) aligns the robot's viewpoint with features to maximize visibility during challenging maneuvers [2], [13], [22]. Information-theoretic surrogates such as Fisher Information Fields provide differentiable maps for actively choosing informative viewpoints [23], [24]. In exploration and inspection, planners explicitly reason about localization uncertainty to select safe, observable trajectories [3].

The research direction of our work treat unobserved space as potentially blocked or even hazardous within partial maps, thus requiring its goal-targeted exploration [6], [15], [16]. The high-level planners in [15], [16] enforce perception awareness along an optimistic trajectory by monitoring potentially hazardous regions, while [6] plans to explore the frontier of unknown space via a Dijkstra graph-search. While these ideas typically resort to a separate planner, they motivate our use of perception-driven costs in a single controller that rewards the line of sight to the goal and prevents motion into unknown regions, with a focus on highly cluttered environments obtained from depth images.

**Algorithms for Model Predictive Control.** In the following, the proposed Model Predictive Path Integral (MPPI) algorithm is motivated, which contrasts with other sampling or derivative-based algorithms [25].

*a) Derivative-based MPC.:* When objectives and constraints admit smooth approximations, derivative-based MPC offers strong local convergence and tight constraint handling. Progressive smoothing and continuation strategies have been proposed to navigate nonconvex obstacle costs [19], conceptually paralleling equal to annealing in sampling-based MPC [26]. A recent approach proposed an algorithm with an external active set solver for cluttered point-cloud obstacles [27], which achieves a feasible average but has an ample worst-case computation time. Still, in settings where, in addition to obstacles represented by raw depth maps also the cost functions are highly nonsmooth, constructing reliable differentiable surrogates remains challenging. Sampling meth-

ods showed superior performance in these settings [28].

*b) Sampling-based MPC.:* Sampling-based MPC methods optimize control sequences by Monte Carlo rollouts rather than local gradients, making them attractive for nonconvex, nonsmooth objectives and dynamics. The most straightforward but largely inefficient random shooting method purely randomizes actions [29]. The Cross Entropy Method (CEM) [30] iteratively refines a distribution toward high-performing controls by elites-only update and has been widely adopted as a trajectory optimizer and within model-based RL pipelines [31]. MPPI control [32]–[34] uses all samples via weighting to iteratively refine an trajectory [35]. MPPI has recently been pushed to demanding hardware settings, including agile UAVs [11] and whole-body locomotion [36].

A key advantage of sampling-based MPC is its robustness to nonconvex costs and discontinuities that arise in the presence of obstacles. In such regimes, gradient estimators can be biased or have high variance [28]. Nevertheless, sampling can suffer from local minima and sample inefficiency. Recent work improves exploration via annealing and diffusion-inspired smoothing of the control distribution [26]. By incorporating novel concepts from [26], we demonstrate how our proposed algorithm outperforms the current state-of-the-art MPPI control algorithm for UAVs [11] in challenging real-world experiments.

## III. Preliminaries

In the following, the notation of this paper is introduced in Sect. III-A, followed by a brief description of the MPC framework in Sect. III-B.

### A. Notation

We introduce two reference frames: $W$, the fixed world frame, whose $z$-axis is gravity-aligned, and $B$, the quadrotor body frame, whose $x$-axis aligns with the onboard camera's principal axis. In this paper, vectors and matrices are written in bold, with matrices indicated by capital letters. Each vector carries a subscript specifying the frame in which it is expressed and its endpoint. For instance, $\boldsymbol{p}_{WB}$ denotes the position of the body frame $B$ relative to the world frame $W$, and $\boldsymbol{R}_{WB}$ denotes the rotation from frame $B$ to $W$. We use $\boldsymbol{y}_{1:N} \in \mathbb{R}^{Nn_y}$ for the vectorization of vectors $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N$, with $\boldsymbol{y}_i \in \mathbb{R}^{n_y}$. The quaternion algebra is $\mathbb{H}$ and $q \in \mathbb{H}_1 := \{q \in \mathbb{H} \mid \|q\| = 1\}$.

### B. Model Predictive Control

Given an environment, often formulated as a Markov decision process (MDP), with the states $\boldsymbol{x} \in \mathbb{R}^{n_x}$, controls $\boldsymbol{u} \in \mathbb{R}^{n_u}$, a stage cost $l : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R} \cup \{\infty\}$, a discount factor $\gamma$, and a stochastic model $P : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathrm{Dist}(\mathbb{R}^{n_x})$, MPC provides a means to yield nearly optimal (sufficiently suboptimal) controls by solving an approximated implicit version of the MDP online, local at the current state $\hat{\boldsymbol{x}} \in \mathbb{R}^{n_x}$, cf. [25] for details. MPC typically uses a simplified deterministic model $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$, possibly simplified stage cost $\ell : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$ and an approximation $\bar{V} : \mathbb{R}^{n_x} \to \mathbb{R}$ of the optimal value function $V : \mathbb{R}^{n_x} \to \mathbb{R}$. Moreover, an

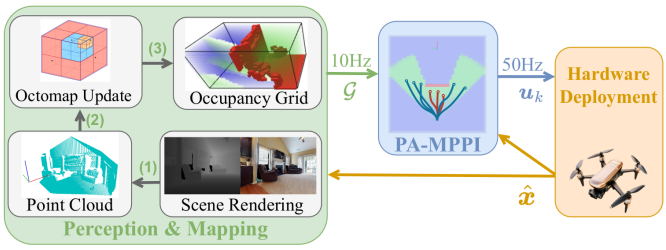



Fig. 2: Illustration of the control stack. The custom perception and mapping module has three parts: (1) a point cloud is generated from the rendered depth image and transformed to the world frame, (2) the Octomap updates the aggregated observations with the new point cloud, and (3) the leaf nodes of the Octomap are used to generate the occupancy grid.

open-loop trajectory $\boldsymbol{u}_{0:H-1}$ with open-loop states $\boldsymbol{x}_i \in \mathbb{R}^{n_x}$, actions $\boldsymbol{u}_i \in \mathbb{R}^{n_u}$ and horizon $H$ is optimized, instead of a policy. The resulting single shooting MPC optimization problem is therefore

$$\begin{aligned} &\min_{\boldsymbol{u}} \ \bar{V}(\boldsymbol{x}_H; \boldsymbol{p}) + \sum_{k=0}^{H-1} \ell(\boldsymbol{x}_k, \boldsymbol{u}_k; \boldsymbol{p}) \\ &\text{with} \quad \boldsymbol{x}_0 = s, \ \boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k), \ 0 \le k < H, \end{aligned} \tag{1}$$

where constraints are approximated in the value and cost function. We denote the dependency of the cost function on external parameters, such as obstacle parameters or the goal by $\boldsymbol{p} \in \mathbb{R}^{n_p}$. Unlike gradient-based algorithms for solving the MPC problem (1), which typically rely on first or second-order derivatives, MPPI uses Monte Carlo sampling-based optimization. Despite the initial derivation of MPPI control for the stochastic system [32]–[34], it is often used for the deterministic counterpart (1), where noise is added as part of the optimization algorithm [37].

## IV. Methodology

A graphical overview of the integrated control stack is shown in Fig. 2. As illustrated, PA-MPPI receives an occupancy grid from the perception module and optimizes a trajectory as control sequences, which are directly executed by the quadrotor. In the hardware-in-the-loop setting, the quadrotor state is used to render depth images, which are then processed by the perception pipeline to update the occupancy grid in real time. The proposed PA-MPPI algorithm utilizes a dynamic model, as described in Sect. IV-A. A detailed description of the MPPI formulation is given in Sect. IV-B. The perception and mapping part, and the cost definition are detailed in Sect. IV-C and IV-D, respectively.

### A. Quadrotor Dynamics

We denote the quadrotor's position, orientation, and linear velocity in the world frame by $\boldsymbol{p}_{WB} \in \mathbb{R}^3$, $\boldsymbol{q}_{WB} \in \mathbb{H}_1$, and $\boldsymbol{v}_{WB} \in \mathbb{R}^3$, respectively, and the quadrotor's angular velocity in the body frame by $\boldsymbol{\omega}_B \in \mathbb{R}^3$. The collective thrust and the corresponding thrust vector in the body frame are defined as $c = c_1 + \ldots + c_{N_{\text{rot}}}$ and $\boldsymbol{c}_B = \begin{bmatrix} 0 & 0 & c \end{bmatrix}^\top$, where $c_i$ is the thrust generated by the $i$-th of $N_{\text{rot}}$ motors. The quadrotor mass is $m$ and $\boldsymbol{g}_W$ is the gravity vector in the world frame. Finally, the diagonal moment of inertia matrix is $\boldsymbol{J} \in \mathbb{R}^{3\times3}$, and the

body torque is $\tau_B \in \mathbb{R}^3$. The quadrotor dynamics can then be expressed as

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{\boldsymbol{p}}_{WB} \\ \dot{\boldsymbol{q}}_{WB} \\ \dot{\boldsymbol{v}}_{WB} \\ \dot{\boldsymbol{\omega}}_B \end{bmatrix} = \begin{bmatrix} \boldsymbol{v}_{WB} \\ \frac{1}{2}\Lambda(\omega_B) \cdot \boldsymbol{q}_{WB} \\ \boldsymbol{q}_{WB} \odot \boldsymbol{c}_B/m + \boldsymbol{g}_W \\ \boldsymbol{J}^{-1}(\boldsymbol{\tau}_B - \boldsymbol{\omega}_B \times \boldsymbol{J} \cdot \boldsymbol{\omega}_B) \end{bmatrix}. \tag{2}$$

The lowest-level flight controller tracks the zero-order hold PA-MPPI control $\boldsymbol{u}_t = \begin{bmatrix} c_t & \boldsymbol{\omega}_{B,t} \end{bmatrix}^\top$. To ensure a feasible total thrust and body rate at each timestep, we follow the single motor thrust clipping in [11], using the motor thrust limits to acquire clipped control input $u_t^{\text{clip}}$, which is then used by PA-MPPI to simulate the dynamics (2) via forward Euler integration.

### B. MPPI formulation

In the MPPI framework, at each timestep $k$, $N$ parallel trajectories of $H$ steps are sampled by adding multivariate Gaussian noise to the nominal control input $\boldsymbol{u}_{k:k+H}^{\text{nom}}$. The perturbed control sequences $\boldsymbol{u}_{k:k+H}^j$, $j = 1, ..., N$, are then rolled out from the quadrotor state $\boldsymbol{x}_k$ using the model (2). Each sampled trajectory $\boldsymbol{x}_{k:k+H}^j$ is then evaluated by summing the per-step costs $\mathcal{L}^j = \sum_{i=k}^{k+H-1} \ell(\boldsymbol{x}_i^j, \boldsymbol{u}_i^j) + \bar{V}(\boldsymbol{x}_H^j)$. The optimized control sequence is calculated using the exponentially weighted average based on the summed cost: $\boldsymbol{u}_{k:k+H-1} = \sum_{j=1}^N w^j \boldsymbol{u}_{k:k+H-1}^j$, $w^j = \frac{\exp\left(-\frac{\mathcal{L}^j - \mathcal{L}^{\min}}{\lambda}\right)}{\sum_{j=1}^N \exp\left(-\frac{\mathcal{L}^j - \mathcal{L}^{\min}}{\lambda}\right)}$, where $\mathcal{L}^{\min}$ is the lowest summed cost out of the $N$ rollouts, and $\lambda$ is the temperature parameter. A low $\lambda$ assigns higher weight to the best-performing rollout, while higher values assign more uniform weights to all rollouts [35]. The first action of the averaged sequence is then executed, and the optimization process repeats in the next time step.

Recent works [11], [26], [38] have shown success in decoupling the prediction time step size $\Delta t_{\text{pred}}$, and the control step size $\Delta t_{\text{ctrl}}$. With $\Delta t_{\text{pred}} > \Delta t_{\text{ctrl}}$, the policy rollouts can predict over a longer real-time horizon for the same number of forward simulation steps, allowing optimization of actions further into the future. Since the optimization loop executes the first action at control frequency and the actions in the sequence are spaced by $\Delta t_{\text{pred}}$, to reuse the remaining actions as $\boldsymbol{u}^{\text{nom}}$ for the next iteration, the control sequence is linearly interpolated and shifted by $\Delta t_{\text{ctrl}}$, then down-sampled at $\Delta t_{\text{pred}}$.

While the terminal value $\bar{V}$ function should in principle resemble the optimal value function and account for a recursive feasible safe set, cf. [25], we only consider a simple terminal hovering safe set with zero velocity and resort to a long enough planning horizon to diminish its influence on the open loop cost.

### C. Perception & Environment Mapping

We use a depth sensor on the quadrotor to continuously build a 3D map of the environment as the policy navigates toward the goal. From each depth image, we extract a point cloud and transform it into the world frame using the corresponding camera pose. The point clouds are then inserted into an Octomap [40], which efficiently aggregates all past observations. During this process, ray-tracing marks the space between the camera and each occupied voxel as free. Since Octomap stores occupied and free voxels in a tree structure, we convert its leaf nodes into a 3D occupancy grid representation to enable $O(1)$ occupancy lookup. The occupancy grid uses the same voxel resolution as the Octomap. This conversion also defines unknown regions that are neither occupied nor free within the grid boundary. As a result, each voxel in the occupancy grid has one of three states: occupied, free, or unknown, with integer values $\{1, 0, -1\}$ respectively, as visualized in Fig. 3. Formally, the 3D occupancy grid is defined as $\mathcal{G} \in \{-1, 0, 1\}^{X \times Y \times Z}$. We denote by $\mathcal{G}(\mathbf{p}_{WB})$ the operation that looks up the voxel corresponding to position $\mathbf{p}_{WB}$ and returns its value. The mapping pipeline is able to process depth images at 30 Hz and update the occupancy grid at 10 Hz for a $4 \times 4 \times 2$m grid with 0.1 m voxel resolution.

### D. Optimization Cost Definition

The stage cost of (1) for the navigation task has the following terms: $\ell = \ell_{\text{goal}} + \ell_{\text{act}} + \ell_{\text{collision}} + \ell_{\text{perception}}$, with

$$\begin{aligned}
\ell_{\text{goal}} &= \left( - c_{\text{pos}} + c_\psi \cdot |\Delta\psi| \right) \cdot \exp\left( - \|\boldsymbol{d}_{\text{goal}}\|^2 \right), \\
\ell_{\text{act}} &= \|\boldsymbol{u}\|_R^2 + \|\Delta \boldsymbol{u}\|_{R_\Delta}^2, \\
\ell_{\text{collision}} &= c_{\text{collision}} \cdot \mathbf{1}_{\{\mathcal{G}(\boldsymbol{p}_{WB}) \neq 0\}}, \\
\ell_{\text{perception}} &= c_{\text{PoI}} \cdot (1 - \langle \hat{\mathbf{x}}_{WB}, \hat{\boldsymbol{d}}_{\text{goal}} \rangle)^2 \cdot \mathbf{1}_{\{\|\boldsymbol{d}_{\text{goal}}\| > c_{\text{thresh}}\}} \\
&\quad + c_{\text{occupied}} \cdot \mathbf{1}_{\{\mathcal{G}(\boldsymbol{r}(t^*)) = 1\}} \\
&\quad + c_{\text{unknown}} \cdot \mathbf{1}_{\{\mathcal{G}(\boldsymbol{r}(t^*)) = -1\}} \\
&\quad + c_{\text{free}} \cdot \mathbf{1}_{\{\mathcal{G}(\boldsymbol{r}(t^*)) = 0\}}
\end{aligned}$$

and the quantities:

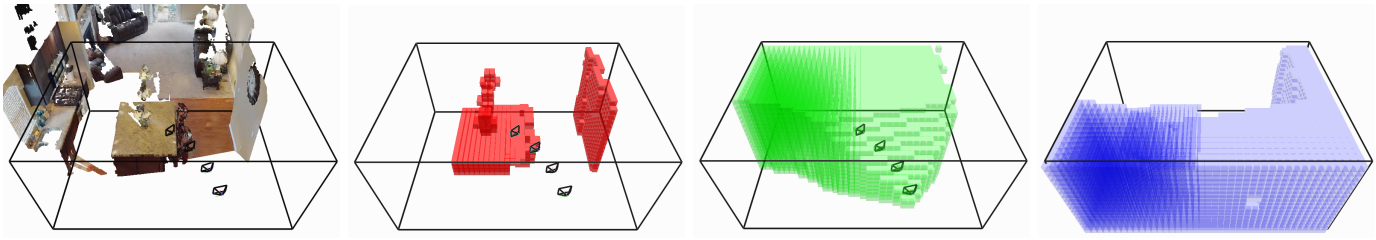
$$\begin{aligned}
\boldsymbol{d}_{\text{goal}} &= \boldsymbol{p}_{WB} - \boldsymbol{p}_{\text{goal}}, \quad \hat{\boldsymbol{d}}_{\text{goal}} = \frac{\boldsymbol{d}_{\text{goal}}}{\|\boldsymbol{d}_{\text{goal}}\|}, \\
\hat{\mathbf{x}}_{WB} &= \boldsymbol{R}_{WB} \cdot \boldsymbol{e}_1, \\
\psi_{WB} &= \text{atan2}(\boldsymbol{R}_{WB\{1,0\}}, \boldsymbol{R}_{WB\{0,0\}}), \\
\Delta\psi &= \text{atan2}(\sin(\psi_{WB} - \psi_{\text{goal}}), \cos(\psi_{WB} - \psi_{\text{goal}})).
\end{aligned}$$

The terminal value function is $\bar{V}(x) = c_{\text{safe}} \cdot \mathbf{1}_{\{\|\boldsymbol{v}_{WB}\| > \underline{v}\} \vee \|\boldsymbol{\omega}_B\| > \underline{\omega}}$, with bounds $\underline{v}$ and $\underline{\omega}$, and a large penalty $c_{\text{safe}}$ to account for safe set (hovering).

The goal cost $\ell_{\text{goal}}$ is split into two components: a position cost with weight $c_{\text{pos}}$, which penalizes distance from the desired goal position, and a yaw cost with weight $c_\psi$, which encourages alignment of the quadrotor's heading with the goal pose. Only yaw alignment is explicitly considered, as each episode terminates with the quadrotor at the goal pose with zero velocity and acceleration. The yaw penalty remains small when the quadrotor is far from the goal, allowing perception loss to have a greater influence over yaw during navigation.

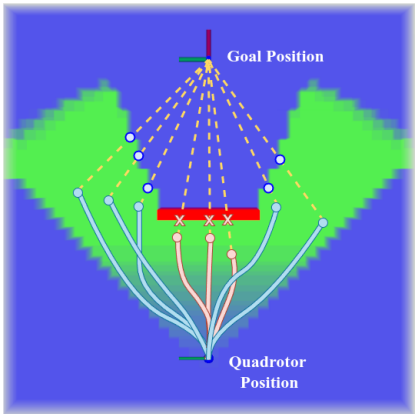
The action cost $\ell_{\text{act}}$ penalizes the magnitude and change in control inputs, similar to the implementation in [11]. To penalize each component of the collective thrust and body rate, we use the positive semidefinite diagonal matrices $R \in \mathbb{R}^{4\times4}$ and $R_\Delta \in \mathbb{R}^{4\times4}$ in corresponding dimensions.

The collision cost $\ell_{\text{collision}}$ is a binary value weighted by a large constant $c_{\text{collision}}$. The indicator function $\mathbf{1}_{\{\mathcal{G}(\boldsymbol{p}_{WB}) \neq 0\}}$ returns 1 if the quadrotor's current position lies outside the set of free voxels defined in Section IV-C. This large penalty

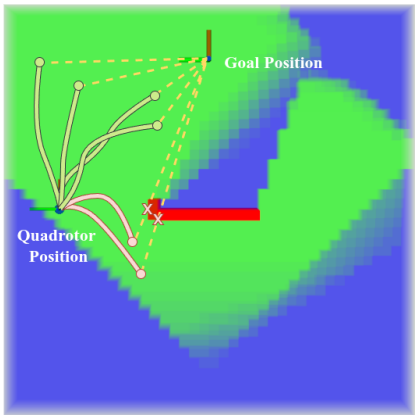


(a) Reconstructed Scene (b) Occupied Voxels (c) Free Voxels (d) Unknown Voxels

Fig. 3: Mapping example of a scene from the Habitat Matterport dataset [39] using four consecutive RGB–D images with the corresponding camera poses. (a) Scene reconstruction obtained by projecting RGB values onto the depth point cloud, with camera poses and the mapping bounding box overlaid. (b)(c) Occupied and free voxels extracted from the Octomap. (d) Unknown voxels, which naturally appear outside the camera's FoVs or in areas occluded by objects.




(a) Given only one depth observation, there is no position in known free space that has a direct line-of-sight to the goal. Trajectories receive a reward for exploring unknown regions (blue trajectories) or a penalty for facing obstacles (red trajectories).




(b) After moving to a new position while mapping the environment, the trajectories that contain positions with direct line-of-sight to the goal (green trajectories) are strongly preferred in the optimization process by receiving a large reward.

Fig. 4: A top-down visualization of the ray-tracing in perception cost calculation, showing the occupied voxels (red), free voxels (green), and unknown voxels (blue), and sampled trajectories on which ray-tracing is performed.

enforces the quadrotor not only to avoid collisions with known obstacles but also to refrain from entering unknown regions, which is critical for ensuring safety when navigating in unknown environments.

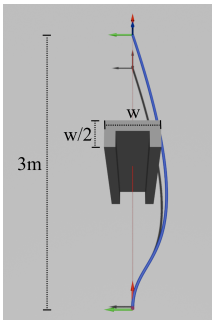
The perception cost, $\ell_{\text{perception}}$, consists of two components. The first, weighted by $c_{\text{PoI}}$, encourages alignment of the quadrotor's x-axis (coinciding with the depth camera's principal axis) with the direction of the goal position (point of interest), thereby maximizing the goal's visibility within the image frame [2], [13]. As the quadrotor approaches the goal (distance below $c_{\text{thresh}}$), this term becomes inactive and yaw control authority is handed to the goal cost. The last three terms in $\ell_{\text{perception}}$ represent three mutually exclusive cases of the quadrotor position with respect to the mapped region of the environment. We define a ray that starts from the quadrotor position $\boldsymbol{p}_{WB}$ and ends at the goal position $\boldsymbol{p}_{\text{goal}}$ as: $\boldsymbol{r}(t) = \boldsymbol{p}_{WB} + t \cdot \boldsymbol{d}_{\text{goal}}$, $0 \leq t \leq 1$. Since $\boldsymbol{p}_{WB}$ is constrained to the known free space due to the collision cost, there exists a $t^*$ such that the ray either exits the free space (i.e., no direct line-of-sight to the goal; see Fig. 4a) or reaches the goal (direct line-of-sight available; see Fig. 4b). If $\boldsymbol{r}(t^*)$ lies in an occupied voxel, a cost $c_{\text{occupied}}$ is assigned. If $\boldsymbol{r}(t^*)$ falls in an unknown voxel, then it suggests an exploration frontier towards the goal is present, and a negative cost $c_{\text{unknown}}$ is given. If $\boldsymbol{r}(t^*)$ remains in a free voxel, then the ray successfully reaches the goal without leaving free space, indicating a direct path to the goal, and a large negative cost $c_{\text{free}}$ is applied. As illustrated in Fig. 4, this ray-tracing term favors sampled trajectories that either explore unknown regions when the goal is blocked by obstacles (Fig. 4a) or move directly toward the goal when possible, cf. Fig. 4b. This design enables the PA-MPPI controller to exploit map information, allowing it to plan around obstacles and efficiently explore unknown space. For the implementation of ray tracing, we adopt the 3D Digital Differential Analyzer (DDA) algorithm [41], which does not require a signed distance field representation of the environment. Due to high computational cost, ray tracing is performed on every $10^{\text{th}}$ time step of the open loop trajectory.

## V. Experiments

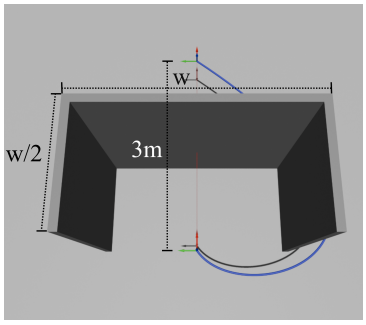
To evaluate the performance of PA-MPPI without considering the effect of other modules in the control loop, such as depth sensor noise or imperfect state estimation, we conduct hardware-in-the-loop (HIL) [42] experiments. We use a motion capture system to acquire the ground truth state of the quadrotor, and the Flightmare simulator [43] to render depth images at the corresponding poses in real time. The PA-MPPI controller is implemented in JAX and integrated into the Agilicious control framework [42], running on a laptop with an i7-13800H CPU with 64GB RAM and a NVIDIA A1000 laptop GPU with 6GB VRAM. The quadrotor used has a mass of 0.21 kg and arm length $l$ = 19.4 cm, with propeller radius of 3.81 cm and a thrust to weight ratio of 6.8.

TABLE I: MPPI parameter.

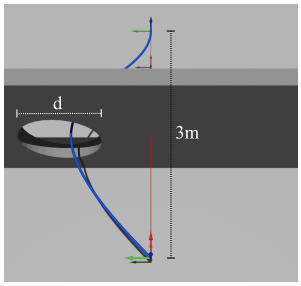
| MPPI param. | | Cost definition constants | | | |
|---|---|---|---|---|---|
| $N$ | 10,000 | $c_{\text{pos}}$ | 2.5 | $c_{\text{free}}$ | -5.0 |
| $H$ | 15 | $c_{\psi}$ | 1.0 | $c_{\text{unknown}}$ | -1.0 |
| $\lambda$ | 0.05 | $c_{\text{collision}}$ | 15.0 | $c_{\text{occupied}}$ | 2.0 |
| $\Delta t_{\text{pred}}$ | 0.1 | $c_{\text{PoI}}$ | 5.0 | $R$ | diag(0.01, 0.1, 0.1, 0.2) |
| $\Delta t_{\text{ctrl}}$ | 0.02 | $c_{\text{thresh}}$ | 0.5 | $R_{\Delta}$ | diag(0.02, 0.02, 0.02, 0.05) |



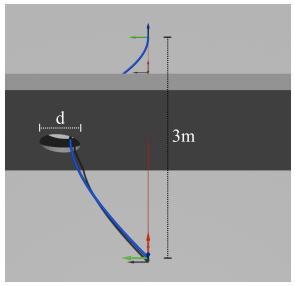

(a) C-wall $w = 0.25\,\text{m}$



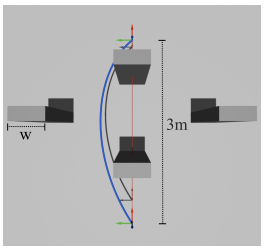

(b) C-wall $w = 3.0\,\text{m}$



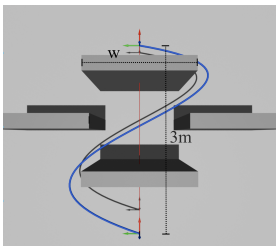

(c) Hole $d = 1.0\,\text{m}$




(d) Hole $d = 0.5\,\text{m}$




(e) 4-Wall $d = 0.5\,\text{m}$

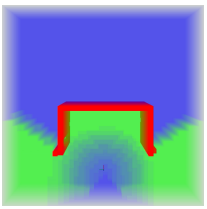



(f) 4-Wall $d = 1.5\,\text{m}$

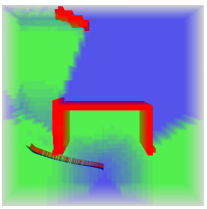
Fig. 5: Synthetic scenes for navigation experiments. The goal pose for each task is always 3 m ahead of the initial pose, with three types of obstacles in between: a C-shaped wall (a)(b), a wall with a hole (c)(d), and four walls (e)(f). The easiest and hardest settings for each scene are shown here, with the reference trajectory shown in red (for baseline only) and example successful trajectories in blue.

Two sets of experiments are conducted. The first consists of navigating synthetic scenes of varying difficulty (Fig. 5) to quantitatively evaluate the performance of PA-MPPI. The second consists of indoor navigation scenes from the Habitat Matterport dataset [39], using goal poses proposed by a navigation foundation model [4] to demonstrate an example usage of PA-MPPI as the action policy for a Vision-Language-Action (VLA) model in unknown environments. A list of PA-MPPI parameters are provided in Table I. As in our experiments, the weight of the terminal safe set $c_{\text{safe}}$ barely had an influence, we set it to zero.

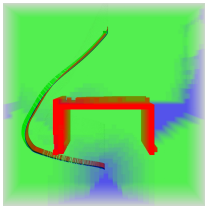
**Synthetic Scene Experiment.** To quantitatively evaluate the performance of PA-MPPI, we design three scenes to navigate through. The first is a C-shaped wall of varying sizes that is a good example of challenging obstacles. The second is a hole in a wall that the quadrotor must go through, similar to manholes that quadrotors must navigate through during ship inspections [44]. The third scene consists of four walls that the quadrotor must navigate past, which tests PA-MPPI's path-finding capabilities. The difficulty of each scene is parametrized by the size of the obstacles $w$ (each difficulty tested 5 times) or the diameter of the holes $r$ (5 random locations, each tested 2 times), as shown in Fig. 5.



(a) $t = 0\,\text{s}$



(b) $t = 1.6\,\text{s}$



(c) $t = 5.0\,\text{s}$

Fig. 6: PA-MPPI's trajectory in the C-wall scene ($w = 2.0\,\text{m}$)

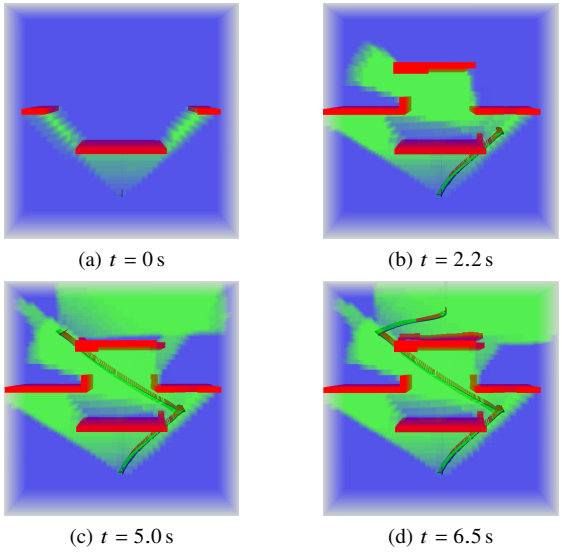
To evaluate the effectiveness of incorporating mapping information in the perception cost, we chose a trajectory tracking MPPI controller as the baseline. As reported in [11], this baseline is able to deviate from the reference trajectory to avoid obstacles, making it a suitable comparison in the navigation task. The tracking MPPI controller shares the same cost definition except that the perception cost $\ell_{\text{perception}}$ is not used. Moreover, the goal cost $\ell_{\text{goal}}$ is replaced with the trajectory tracking cost in [11]. The reference trajectory is created by uniformly sampling waypoints along the straight line connecting the initial and goal position, and optimizing for a minimum-jerk trajectory with 4 s horizon.

The experiment results are summarized in Table II. Success termination is defined as reaching the goal pose without getting stuck on or colliding with an obstacle. PA-MPPI drastically improves the success rate by avoiding getting stuck on all obstacles. Given an incomplete initial map of the environment, it moves to positions that help map the environment between its position and the goal, which is visualised in Fig. 6 and Fig. 7, respectively. In these cases, where the successful path requires a large deviation from the straight line reference, tracking MPPI fails by getting stuck on the obstacles, failing to find a path towards the goal. However, PA-MPPI is not restricted by the reference trajectory and instead is guided by the perception cost to navigate around the obstacles. Both controllers suffer collisions, especially when the velocity is high. Investigating the collisions shows that the PA-MPPI controller enters the occupied region with an average distance of 4 cm. Repeating the experiment in simulation does not result in such collisions, suggesting a gap between the simplified quadrotor model for trajectory rollouts and the real drone dynamics.

**PA-MPPI combined with Navigation Foundation Model.** We validate the real-world applicability of the PA-MPPI controller by using it as the action policy for a navigation foundation model that proposes trajectory waypoints. In this case, we use two scenes from the Habitat Matterport dataset

TABLE II: Synthetic scene navigation experiment results.

| | Termination Condition | C-Shaped Wall Obstacle size $w$ | | | | Hole Traversal Hole diameter $d$ | | 4-Wall Traversal Wall size $w$ | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | 0.5 m | 1.0 m | 2.0 m | 3.0 m | 0.5 m | 1.0 m | 0.5 m | 1.0 m | 1.5 m |
| Tracking MPPI | Success | **100%** | 20% | 0% | 0% | 20% | 50% | 20% | **80%** | 0% |
| | Stuck | 0% | 60% | 100% | 100% | 40% | 20% | 0% | 0% | 80% |
| | Collision | 0% | 20% | 0% | 0% | 40% | 30% | 80% | 20% | 20% |
| PA-MPPI | Success | **100%** | **100%** | **100%** | **60%** | **80%** | **100%** | **100%** | 40% | **80%** |
| | Stuck | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 0% |
| | Collision | 0% | 0% | 0% | 40% | 20% | 0% | 0% | 60% | 20% |

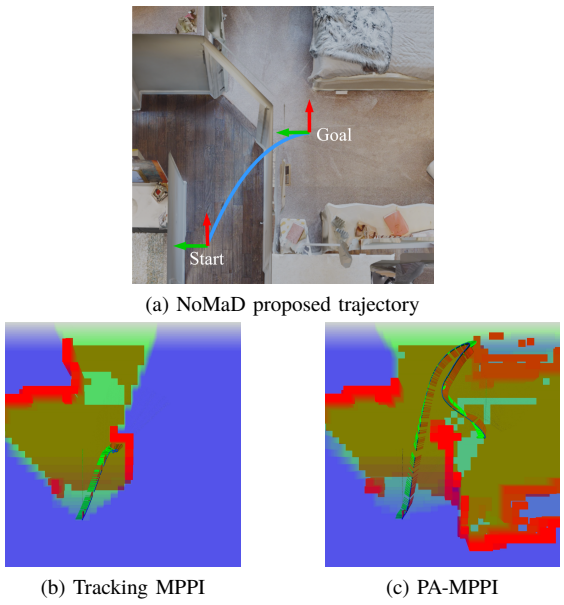


(a) $t = 0\,\text{s}$ (b) $t = 2.2\,\text{s}$ (c) $t = 5.0\,\text{s}$ (d) $t = 6.5\,\text{s}$

Fig. 7: PA-MPPI's trajectory in the 4-wall scene ($w = 1.5\,\text{m}$)

[39] and NoMaD [4] for waypoint generation. The reference trajectory for the tracking MPPI controller is a minimum-jerk trajectory that passes through all the waypoints in 4 s.

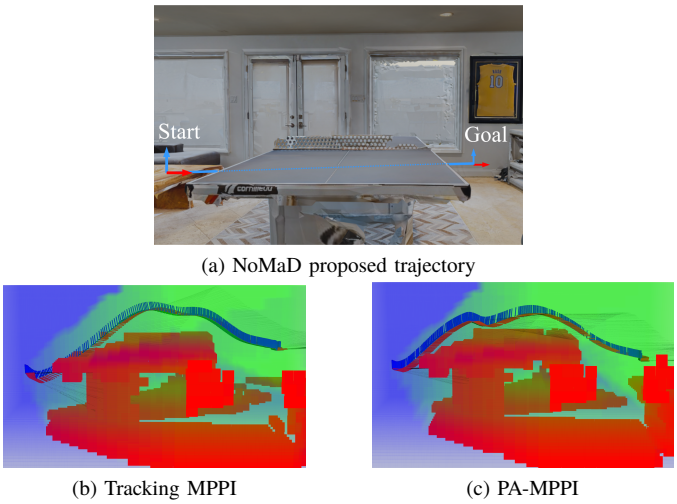
In the first scene, visualized in Fig. 8, the NoMaD proposed trajectory attempts to enter the room but misses the door. The trajectory tracking controller follows the proposed trajectory and gets stuck on the wall. On the other hand, PA-MPPI was able to explore and map more regions in this scene, including navigating through the door and ultimately reaching the goal position. In the second scene, visualized in Fig. 9, the NoMaD trajectory starts close to a ping-pong table and goes directly through it. Although both controllers attempt to avoid collisions with the obstacle, the tracking MPPI, guided by the reference trajectory, collides with the ping-pong table at the beginning of its trajectory. On the contrary, PA-MPPI successfully avoids the obstacle. Both scenarios demonstrate the robustness of PA-MPPI against imperfect trajectory proposals in unknown environments, making it a suitable action policy for navigation foundation models.




(a) NoMaD proposed trajectory

(b) Tracking MPPI (c) PA-MPPI

Fig. 8: Habitat Scene 1, top-down view of the floorplan, with NoMaD trajectory in blue.




(a) NoMaD proposed trajectory

(b) Tracking MPPI (c) PA-MPPI

Fig. 9: Habitat Scene 2, with NoMaD trajectory in blue. The tracking MPPI trajectory collides with the ping-pong table at the beginning of the trajectory, whereas PA-MPPI safely navigates around it.

## VI. Discussion & Future Work

Despite outperforming the tracking MPPI controller, the current implementation of PA-MPPI can still suffer from collisions, as observed in Section V, due to its simplified dynamics model and the low update rate of 50 Hz. Additionally, due to the limited FoV of the depth sensor, a good initial observation is crucial for successful navigation around large obstacles. For example, in the experiment shown in Fig. 6, the initial observation at $t = 0\,\text{s}$ is initialized by turning the quadrotor $\pm 90°$ to observe free space outside the convex hull of the C-shaped wall. However, this issue can be solved by replacing the depth camera with LIDAR, which only requires modification to the mapping module. Finally, we only considered local navigation tasks within a fixed 3D boundary. We leave the implementation of long-horizon navigation scenarios, which

may involve navigating to intermediate waypoints proposed by foundation models, for future work.

## VII. Conclusion

This work introduces a perception-aware MPPI controller that incorporates a novel perception-aware cost to enhance quadrotor navigation in partially known cluttered environments. By leveraging the current map, the perception term guides trajectories toward promising frontiers as they progress toward the goal. Real-world experiments demonstrate its effectiveness compared to state-of-the-art MPPI controllers that do not account for the potential benefit of exploring unknown areas. Moreover, we demonstrate its potential as an action policy for foundation models for navigating challenging environments. Future improvements will include a more realistic quadrotor dynamics model with higher control frequency to increase robustness, as well as validation on long-term navigation tasks.

## References


[1] L. F. Recalde, B. S. Guevara, C. P. Carvajal, V. H. Andaluz, J. Varela-Aldás, and D. C. Gandolfo, "System identification and nonlinear model predictive control with collision avoidance applied in hexacopters UAVs," *Sensors*, vol. 22, no. 13, 2022.

[2] J. Xing, G. Cioffi, J. Hidalgo-Carrió, and D. Scaramuzza, "Autonomous power line inspection with drones via perception-aware MPC," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 1086–1093.

[3] C. Papachristos, F. Mascarich, S. Khattak, T. Dang, and K. Alexis, "Localization uncertainty-aware autonomous exploration and mapping with aerial robots using receding horizon path-planning," *Autonomous Robots*, vol. 43, no. 8, pp. 2131–2161, 2019.

[4] A. Sridhar, D. Shah, C. Glossop, and S. Levine, "NoMaD: Goal Masked Diffusion Policies for Navigation and Exploration," *arXiv preprint*, 2023.

[5] A.-C. Cheng, Y. Ji, Z. Yang, X. Zou, J. Kautz, E. Biyik, H. Yin, S. Liu, and X. Wang, "Navila: Legged robot vision-language-action model for navigation," in *RSS*, 2025.

[6] T. Cieslewski, E. Kaufmann, and D. Scaramuzza, "Rapid exploration with multi-rotors: A frontier selection method for high speed flight," in *IROS*, 2017.

[7] M. Naazare, D. Ramos, J. Wildt, and D. Schulz, "Application of graph-based path planning for uavs to avoid restricted areas," in *2019 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2019, pp. 139–144.

[8] W. Liu, Y. Ren, and F. Zhang, "Integrated planning and control for quadrotor navigation in presence of suddenly appearing objects and disturbances," *IEEE Robotics and Automation Letters*, 2024.

[9] M. W. Achtelik, S. Weiss, M. Chli, and R. Siegwart, "Path planning for motion dependent state estimation on micro aerial vehicles," in *2013 IEEE International Conference on Robotics and Automation*, 2013.

[10] M. Hehn and R. D'Andrea, "Real-time trajectory generation for quadrocopters," *IEEE Transactions on Robotics*, 2015.

[11] M. Minařík, R. Pěnička, V. Vonásek, and M. Saska, "Model predictive path integral control for agile unmanned aerial vehicles," in *IROS*, 2024.

[12] I. S. Mohamed, G. Allibert, and P. Martinet, "Model predictive path integral control framework for partially observable navigation: A quadrotor case study," 04 2020.

[13] D. Falanga, P. Foehn, P. Lu, and D. Scaramuzza, "PAMPC: Perception-aware model predictive control for quadrotors," in *IROS*, 2018.

[14] M. Sarvaiya, G. Li, and G. Loianno, "Hpa-mpc: Hybrid perception-aware nonlinear model predictive control for quadrotors with suspended loads," 11 2024.

[15] B. Zhou, J. Pan, F. Gao, and S. Shen, "Raptor: Robust and perception-aware trajectory replanning for quadrotor fast flight," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1992–2009, 2021.

[16] Q. Yu, C. Qin, L. Luo, H. H.-T. Liu, and S. Hu, "Cpa-planner: Motion planner with complete perception awareness for sensing-limited quadrotors," *IEEE Robotics and Automation Letters*, 2022.

[17] T. Marcucci, M. Petersen, D. von Wrangel, and R. Tedrake, "Motion planning around obstacles with convex optimization," *Science Robotics*, 2023.

[18] Y. Gao, F. Messerer, N. v. Duijkeren, B. Houska, and M. Diehl, "Real-time-feasible collision-free motion planning for ellipsoidal objects," in *2024 IEEE 63rd Conference on Decision and Control (CDC)*, 2024.

[19] R. Reiter, K. Baumgärtner, R. Quirynen, and M. Diehl, "Progressive smoothing for motion planning in real-time nmpc," in *2024 European Control Conference (ECC)*, 2024, pp. 1816–1823.

[20] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Learning high-speed flight in the wild," *Science Robotics*, vol. 6, no. 59, p. eabg5810, 2021.

[21] A. Bhattacharya, N. Rao, D. Parikh, P. Kunapuli, Y. Wu, Y. Tao, N. Matni, and V. Kumar, "Vision transformers for end-to-end vision-based quadrotor obstacle avoidance," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025.

[22] K. Lee, J. Gibson, and E. A. Theodorou, "Aggressive perception-aware navigation using deep optical flow dynamics and pixelmpc," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1207–1214, 2020.

[23] Z. Zhang and D. Scaramuzza, "Fisher information field: an efficient and differentiable map for perception-aware planning," *arXiv preprint*, 2020.

[24] J. Lim, N. Lawrance, F. Achermann, T. Stastny, R. Girod, and R. Siegwart, "Fisher information based active planning for aerial photogrammetry," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 1249–1255.

[25] R. Reiter, J. Hoffmann, D. Reinhardt, F. Messerer, K. Baumgärtner, S. Sawant, J. Boedecker, M. Diehl, and S. Gros, "Synthesis of model predictive control and reinforcement learning: Survey and classification," 2025.

[26] H. Xue, C. Pan, Z. Yi, G. Qu, and G. Shi, "Full-order sampling-based mpc for torque-level locomotion control via diffusion-style annealing," in *ICRA*, 2025.

[27] Y. Gao, F. Messerer, N. van Duijkeren, R. Dabir, and M. Diehl, "Semi-infinite programming for collision-avoidance in optimal and model predictive control," 2025.

[28] H. J. Suh, M. Simchowitz, K. Zhang, and R. Tedrake, "Do differentiable simulators give better policy gradients?" in *Proceedings of the 39th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, Eds., 2022.

[29] J. L. Piovesan and H. G. Tanner, "Randomized model predictive control for robot navigation," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 94–99.

[30] R. Y. Rubinstein, "Optimization of computer simulation models with rare events," *European Journal of Operational Research*, 1997.

[31] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," *Advances in neural information processing systems*, 2018.

[32] H. J. Kappen, "Linear Theory for Control of Nonlinear Stochastic Systems," *Physical Review Letters*, Nov. 2005.

[33] E. A. Theodorou and E. Todorov, "Relative entropy and free energy dualities: Connections to Path Integral and KL control," in *IEEE 51st IEEE Conference on Decision and Control (CDC)*, Dec. 2012.

[34] E. A. Theodorou, "Nonlinear Stochastic Control and Information Theoretic Dualities: Connections, Interdependencies and Thermodynamic Interpretations," *Entropy*, no. 5, May 2015.

[35] 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.

[36] J. Alvarez-Padilla, J. Z. Zhang, S. Kwok, J. M. Dolan, and Z. Manchester, "Real-time whole-body control of legged robots with model-predictive path integral control," in *IEEE International Conference on Robotics and Automation*, 2025.

[37] H. Homburger, F. Messerer, M. Diehl, and J. Reuter, "Optimality and suboptimality of mppi control in stochastic and deterministic settings," *IEEE Control Systems Letters*, vol. 9, pp. 763–768, 2025.

[38] T. Howell, N. Gileadi, S. Tunyasuvunakool, K. Zakka, T. Erez, and Y. Tassa, "Predictive Sampling: Real-time Behaviour Synthesis with MuJoCo," dec 2022.

[39] S. K. Ramakrishnan, A. Gokaslan, E. Wijmans, O. Maksymets, A. Clegg, J. M. Turner, E. Undersander, W. Galuba, A. Westbury, A. X. Chang, M. Savva, Y. Zhao, and D. Batra, "Habitat-matterport 3d dataset (HM3d): 1000 large-scale 3d environments for embodied AI," in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.

[40] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013.
[41] J. Amanatides and A. Woo, "A fast voxel traversal algorithm for ray tracing," *Proceedings of EuroGraphics*, vol. 87, 08 1987.
[42] P. Foehn, E. Kaufmann, A. Romero, R. Pěnička, S. Sun, L. Bauersfeld, T. Laengle, G. Cioffi, Y. Song, A. Loquercio, and D. Scaramuzza, "Agilicious: Open-source and open-hardware agile quadrotor for vision-based flight," *Science Robotics*, vol. 7, 06 2022.
[43] Y. Song, S. Naji, E. Kaufmann, A. Loquercio, and D. Scaramuzza, "Flightmare: A flexible quadrotor simulator," in *CoRL*, 2020.
[44] M. Dharmadhikari, P. De Petris, H. Nguyen, M. Kulkarni, N. Khedekar, and K. Alexis, "Manhole detection and traversal for exploration of ballast water tanks using micro aerial vehicles," in *2023 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2023, pp. 103–109.