A Model Predictive Path Integral Method for Fast, Proactive, and Uncertainty-Aware UAV Planning in Cluttered Environments

Jacob Higgins, Nicholas Mohammad, Nicola Bezzo

***Abstract*— Current motion planning approaches for autonomous mobile robots often assume that the low level controller of the system is able to track the planned motion with very high accuracy. In practice, however, tracking error can be affected by many factors, and could lead to potential collisions when the robot must traverse a cluttered environment. To address this problem, this paper proposes a novel receding-horizon motion planning approach based on Model Predictive Path Integral (MPPI) control theory – a flexible sampling-based control technique that requires minimal assumptions on vehicle dynamics and cost functions. This flexibility is leveraged to propose a motion planning framework that also considers a data-informed risk function. Using the MPPI algorithm as a motion planner also reduces the number of samples required by the algorithm, relaxing the hardware requirements for implementation. The proposed approach is validated through trajectory generation for a quadrotor unmanned aerial vehicle (UAV), where fast motion increases trajectory tracking error and can lead to collisions with nearby obstacles. Simulations and hardware experiments demonstrate that the MPPI motion planner proactively adapts to the obstacles that the UAV must negotiate, slowing down when near obstacles and moving quickly when away from obstacles, resulting in a complete reduction of collisions while still producing lively motion.**

I. Introduction

The interest in autonomous mobile robots (AMR) is fast growing in the private. military, and commercial sectors for its promise to revolutionize key components of many industries, such as logistics, structural inspection and transportation. For these applications, robust autonomy is the key that enables such operations to rapidly scale while keeping human intervention to a minimum, allowing such endeavors to remain affordable. The robots that are deployed in these real-world situations, however, are subject to many sources of uncertainty that introduce risks while in motion, e.g. dynamic obstacle position and disturbances. To compensate for this uncertainty, many approaches are receding-horizon in nature, so that newly-obtained information may affect the resulting trajectory. In particular, receding-horizon motion planning has recently grown into a large field of study within the robotics community.

Typical receding-horizon approaches decouple motion planning and trajectory tracking, and are often treated as separate problems [1]. Such decoupling could lead to potentially dangerous situations if the motion planner is not aware of the limits and capabilities of the lower-level planner. Consider the scenario depicted in Fig. 1 in which an aerial robot must pass through a narrow opening to the other side. A receding-horizon motion planner may command a fast-moving trajectory to reduce travel time, but such a trajectory

Jacob Higgins, Nicholas Mohammad and Nicola Bezzo are with the Department of Electrical and Computer Engineering, University of Virginia, Charlottesville, VA 22903, USA {jdh4je, nm9ur, nbezzo}@virginia.edu

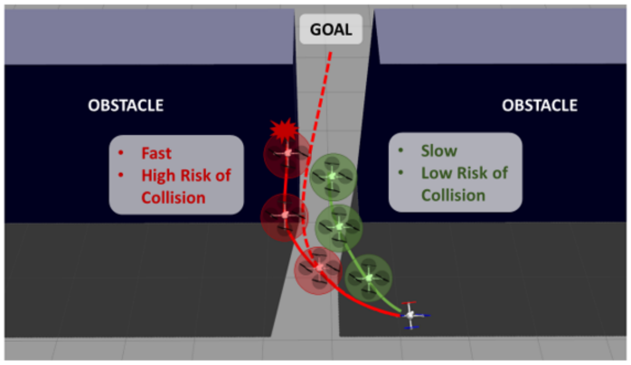


Fig. 1. Motivating example in which a slower speed results in safer motion through a small gap.

may not be perfectly tracked by the low level controller. This could induce a large tracking error, meaning that tracking a collision-free trajectory may not result in collision-free motion. In order to mitigate the risk of collision under this kind of uncertainty, the robot must command slower motion through the gap, reducing the tracking error.

One possible solution is a data-informed approach, where the tracking error and subsequent risk of collision are inferred from past performance of the robot. This data-informed risk assessment allows the risk measure to accurately reflect the performance of the low level controller, but must capture a potentially complex relationship between the commanded trajectory and the risk of tracking that trajectory. Gradient-based and quadratic-programming-based approaches are restrictive in that risk-based costs must have certain numerical qualities for real-time use. Alternatively, sampling-based approaches consider costs with minimal assumptions, allowing a greater flexibility and generality when defining risk. For this reason, the heart of the proposed approach in this paper is a receding-horizon Model Predictive Path Integral (MPPI) motion planner, adapted from the sampling-based MPPI control used in information theoretic control theory [2]. Typical MPPI works by rapidly sampling the low level control space of the system around a "best guess" of the optimal open-loop control policy, and a weighted sum is performed to iteratively update this best guess, converging to the optimal control policy after many iterations. One consequence of sampling within the low level control is the need for a large number of samples at a high rate (typically on the order of 50-100 Hz). For real-time use, this requires the robot to have a GPU on board to speed up the sampling time.

Our approach adopts MPPI control to a trajectory planning setting; instead of sampling within the control space, we sample within a trajectory parameter space. In particular, the proposed MPPI trajectory planner determines waypoints that define a spline-based trajectory. This reduces the sampling space dimension, allowing our MPPI path planner to run on a CPU in real-time, hence for a more relaxed set of system requirements. Overall, our approach allows for a computationally more efficient method to sample trajectories within the MPPI framework, without sacrificing the ability to generate fast and safe trajectories.

This paper presents two main contributions. First, the MPPI control approach is cast as a parameterized high level planner, reducing the dimension of the optimization space and mitigating the hardware requirements needed to find reasonable solutions to the motion planning problem. Second, a data-informed risk measurement is included inside the cost function of the MPPI motion planner, using the actual trajectory tracking performance of the system to determine safe and lively trajectories. The result is motion that minimizes risk of collision due to trajectory tracking error while avoiding obstacles and moving towards a goal. Together, the overall approach provides a practical method for run-time risk-aware trajectory generation towards safe navigation. While this approach is flexible enough to be applied in a general motion planning setting, this paper focuses on motion planning for an unmanned aerial vehicle (UAV) since these types of systems can be greatly affected by tracking error, and such tracking error can induce risk of collision, especially when navigating cluttered environments.

The rest of the paper is organized as follows: in Sec. II we provide an overview of the current state of the art in receding-horizion motion planning, risk-aware motion planning and sampling-based motion planning. In Sec. III we formally define the problem of fast and safe trajectory generation under the risk of collision due to trajectory tracking error. Sec. IV describes the components of the proposed approach which is validated with simulations in Sec. V and physical experiments in Sec. VI. Finally, we draw conclusions and discuss future work in Sec. VII.

II. Related Work

Motion planning remains an active field of research within the robotics community. For agile robotic systems such as quadrotors, most research effort is focused on leveraging this agility by creating fast, aggressive maneuvers [3]. Cutting-edge agile motion planning often uses gradient-based optimization [4], mixed-intenger quadratic programming [1], sample- and search-based methods [5], or some mixture of these techniques [6]. One assumption implicit within these approaches is the ability of a low level controller to accurately track the aggressive trajectories. Many factors, however, can contribute to a gap between the generated trajectory and the actual trajectory. For example, despite the existence of sophisticated nonlinear controller techniques [7], linear PID low level controllers offer sub-optimal performance but remain ubiquitous in many real-world applications for low level tracking due to speed and ease of implementation [8]. Aerodynamic effects [9], [10] and errors within the visual odometry pipeline [11], [12] are additional factors that can degrade trajectory tracking, which in turn introduces a risk of collision.

One possible approach is to use data-driven models to compensate for this uncertainty, such as end-to-end neural network policies [13], self-adjusting system models [14], or disturbance estimation and rejection [15]. More analytical approaches seek to provide rigorous safety guarantees, such as Tube-MPC [16], [17], chance-constrained optimization techniques [18], or risk-constrained motion planning approaches [19]. In these cases, simplifying assumptions are usually placed on the risk involved, or the policy is too computationally intensive for real-time control.

Sampling-based techniques, such as RRT-based techniques [20] and motion primitive sampling [21], [22], offer approximate solutions to otherwise intractable problems, especially when solutions are required in real time. One particular sampling-based technique named STOMP [23] computes trajectories for robotic arm manipulators by stochastic-approximated gradient descent. Such a technique allows motion planning with non-differentiable costs, but is not receding-horizon and requires knowledge of the entire configuration space to perform well, making it not suitable for AMR applications. This work eventually evolved into model predictive path integral (MPPI) control [2], [24], [25], which directly samples within the low level control space of the robot and is theoretically motivated with information-theoretic control techniques. In order to achieve aggressive maneuvers, MPPI control requires approximately 10,000 samples to be taken at a rate of 40 Hz, necessitating the use of a GPU to parallelize and speed up the computation.

Our approach leverages the ability of the MPPI control technique to optimize non-differentiable or complex costs to incorporate data-informed risk within the cost function. Specifically, we examine the risk of collision due to tracking error of a low level controller when aggressive, high-speed trajectories are planned by the MPPI path planner. In order to generalize this procedure and mitigate the number of samples required per MPPI iteration, these trajectories are parameterized over carefully-chosen polynomials, allowing the MPPI procedure to be performed in real-time on a CPU. To the best of our knowledge, our work is the first to utilize MPPI techniques to generate trajectories in a receding horizon fashion while optimizing for a risk cost that does not need to be differentiable.

III. Problem Formulation

In this work, we are interested in creating a receding-horizon trajectory generation policy that addresses the risk of collision due to tracking error between the commanded trajectory and the actual trajectory of a UAV, especially when navigating potentially cluttered environments. Additionally, this trajectory generation policy should be able to handle data-informed functions of risk, and consider potentially complex relationships by placing minimal assumptions on the properties such functions may have (e.g. smoothness, differentiability). We separate this problem into two parts: (i) creation of a receding-horizon trajectory generator that can optimize for a general cost function at run time, and (ii) the inclusion of a risk factor that, when minimized, commands safe and lively trajectories in the presence of low level tracking error.

Problem 1: ***Receding Horizon Trajectory Generation***: We seek a policy $\mathcal{P}_\tau(\boldsymbol{x}(t_0))$ that takes in the current state of the robot $\boldsymbol{x}(t_0) \in \mathbb{R}^{n_x}$ and at run time returns a time-based trajectory $\tau(t)$ defined over a future horizon $t \in [t_0, t_0 + t_H]$ for a low level controller to track. This trajectory should move the robot closer to a goal state $\boldsymbol{x}_g \in \mathbb{R}^{n_x}$, as well as avoid the state set $\mathcal{X}_O \in \mathbb{R}^{n_x}$ occupied by obstacles:

$$\begin{gathered} |\boldsymbol{x}(t_0 + t_H) - \boldsymbol{x}_g| \leq |\boldsymbol{x}(t_0) - \boldsymbol{x}_g| \\ \boldsymbol{x}(t) \notin \mathcal{X}_O,\, \forall t' \in [t_0, t_0 + t_H] \end{gathered} \tag{1}$$

In addition to these requirements, this policy should optimize over a cost function $S(\tau)$ that may be nonlinear, non-smooth and even non-differentiable:

$$\mathcal{P}_\tau = \arg\min_\tau \left[S(\tau)\right] \tag{2}$$

Note that the commanded trajectory $\tau(t)$ may not be the same as the actual trajectory τ_{act} of the system over time, due to tracking error. To compensate for this, the proposed approach also considers a risk measure $\rho(\cdot) \in \mathbb{R}$ that relates a given trajectory $\tau(t)$ to the risk of collision due to this error $|\tau(t) - \tau_{\text{act}}(t)|$. Although we do not constrain ρ to have any particular form, basic assumptions must be placed in order to cast the problem of risk minimization correctly: (i) $\rho(\cdot)$ is positive semi-definite, (ii) $\rho(\cdot) = 0$ for situations that have no risk, and (iii) $\rho(\cdot) > 0$ for situations that have risk of collision. With these assumptions, this risk can be included inside the cost function of the trajectory generation policy.

Problem 2: ***Risk-aware Navigation*****:** Given a risk measure $\rho(\cdot)$, create a policy $\mathcal{P}_\tau$ for finding a trajectory τ that also minimizes risk:

$$\mathcal{P}_\tau = \arg\min_\tau \left[S(\tau) + \int_\tau \rho(\cdot)dt\right] \tag{3}$$

Next, we discuss our proposed approach for safe, risk-aware navigation of a UAV by combining risk measures with a novel MPPI-based motion planning technique.

IV. Approach

We propose an MPPI-based motion planning policy that finds a trajectory $\tau(t)$ for the robot to track, which is chosen by the policy for its ability to guide the robot toward the goal state $\boldsymbol{x}_g$, while also minimizing the risk measure $\rho(\cdot)$. We consider trajectories defined by parameters R so that the task of the MPPI planner is to find a set of parameters R^* that optimize a control objective function over a future time horizon. The motion planner is sampled in a receding horizon fashion, allowing the robot to react to a potentially changing environment. Fig. 2 shows our approach within a typical autonomy stack, in which $\tau(t, R)$ is then fed to a low level controller that produces controls $\boldsymbol{u}$ for reliable tracking.

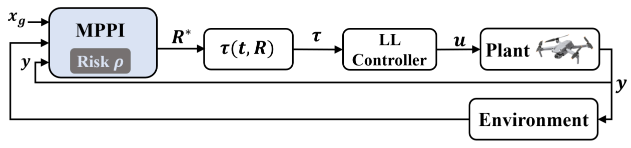


Fig. 2. Diagram showing the proposed motion planner (blue shaded cell) within the context of a general autonomy stack.

To facilitate discussion of the proposed motion planner, the scenario assumed by this paper involves a UAV traveling through a potentially cluttered obstacle course. As is typical of the control pipelines for many AMR, there is some mismatch between the commanded trajectory $\tau(t)$ and the actual trajectory τ_{act}, leading to some tracking error in the state space of the UAV. When the UAV is traveling in a cluttered environment, this mismatch could lead to collision with an obstacle. Thus, the MPPI motion planner should be aware of this risk and command trajectories that move towards the goal while remaining safe.

To this end, Sec. IV-A discusses the parameters R that fully define a trajectory τ. Sec. IV-B then discusses how $\rho(\cdot)$ can be concretely defined. Sec. IV-C discusses how an MPPI can utilize a general risk measure $\rho(\cdot)$ for fast, online and risk-aware motion planning. Finally, Sec. IV-D discusses safety considerations under failure of the receding-horizon MPPI-based motion planner.

A. Trajectory Parametrization

The underlying MPPI motion planning algorithm relies on sampling different perturbations $\mathcal{E}$ to the parameters R, resulting in different trajectories defined by $\tau(t, R + \mathcal{E})$. These trajectories are continuously recomputed over some future horizon and applied in a receding-horizon fashion. In order to minimize the number of random samples needed for best results, the dimension of the sampling space $\mathcal{E}$ should also be relatively small. This paper assumes τ takes the form of a minimum jerk trajectory, since it is a popular choice for UAV trajectory generation [3], [26]. These minimum jerk trajectories are defined by P waypoints $R = \{\boldsymbol{r}_1, \boldsymbol{r}_2, ..., \boldsymbol{r}_P\}$, where $\boldsymbol{r} \in \mathbb{R}^3$ is the position of each waypoint, forming P different trajectory segments. Every min-jerk trajectory segment is defined by a fifth-order polynominal in time along each cartesian direction $i \in [x, y, z]$:

$$\tau^i(t) = \sum_{j=0}^{5} (c_j^i / j!) t^j \tag{4}$$

Although the proposed approach may use any number of waypoints, the rest of the paper will restrict a single trajectory to be defined by $P = 2$ waypoints. This is done to both simplify discussion and reduce the number of decision variables as much as possible, mitigating the computational burden of this approach. This means a trajectory will have two distinct segments, where each is defined by a set of coefficients $\{c_{jk}^i\}$, with $j \in [0, 5]$ denoting the associated power of t and $k \in [1, 2]$ denoting the trajectory segment. Fig. 3(a) shows an example trajectory defined by two waypoints, with each segment having an equal time span of T seconds so that the total trajectory has a time span of $t_h = 2T$ seconds, equal to the horizon-length of the receding horizon planner.

The first trajectory segment (blue colored in Fig. 3(a)) is defined by the initial position p_0^i, velocity v_0^i and acceleration a_0^i of the UAV, as well as the position of the first waypoint r_1^i:

$$\begin{aligned} p_0^i &= c_{01}^i \\ v_0^i &= c_{11}^i \\ a_0^i &= c_{21}^i \\ r_1^i &= \sum_{j=0}^{5} (c_{j1}^i / j!) T^j \end{aligned} \tag{5}$$

Likewise, the second trajectory segment (shown in orange in Fig. 3(a)) is defined by the first waypoint position r_1^i, the second waypoint position r_2^i, as well as the desired velocity

v_e^i and acceleration a_e^i at the end of the total trajectory, as the UAV reaches the second waypoint.

$$\begin{aligned} r_1^i &= c_{02}^i \\ r_2^i &= \sum_{j=0}^{5} (c_{j2}^i / j!) T^j \\ v_e^i &= \sum_{j=1}^{5} (c_{j2}^i / (j-1)!) T^{j-1} \\ a_e^i &= \sum_{j=2}^{5} (c_{j2}^i / (j-2)!) T^{j-2} \end{aligned} \tag{6}$$

In order to avoid defining the velocity and acceleration conditions at the first waypoint, Pontryagin's minimum principle [3] may be used to allow these conditions to remain free. It can be shown that not constraining $v(t)$ and $a(t)$ at the first waypoint directly implies continuity to the associated costate variables $\lambda_v(t)$ and $\lambda_a(t)$, respectively. For a given direction i and trajectory segment k, these costates are:

$$\lambda_v(t) = c_{4k}^i + c_{5k}^i t \tag{7}$$

$$\lambda_a(t) = c_{3k}^i + c_{4k}^i t + (c_{5k}^i/2) t^2 \tag{8}$$

Continuity of these costate values at the first waypoint amounts to the following constraints:

$$\begin{aligned} c_{41}^i + c_{51}^i T &= c_{42}^i \\ c_{31}^i + c_{41}^i T + (c_{51}^i/2) T^2 &= c_{32}^i \end{aligned} \tag{9}$$

Lastly, continuity of the velocity and acceleration at the first waypoint must also be enforced:

$$\begin{aligned} \sum_{j=1}^{5} (c_{j1}^i/(j-1)!) T^{j-1} &= c_{12}^i \\ \sum_{j=2}^{5} (c_{j1}^i/(j-2)!) T^{j-2} &= c_{22}^i \end{aligned} \tag{10}$$

Together, (5), (6), (9) and (10) form 12 linear equations of 12 unknown constants, which may be solved efficiently by any number of linear algebra solvers available [27].

B. Risk Measure

Many types of risk measures have been used in different robotic applications, from defining $\rho(\cdot)$ as the probability of a fault for legged motion [28], to conditional value at risk [19], a more sophisticated measure of risk that accounts for the severity of unlikely but possible events.

This paper proposes a data-informed risk measure that models geometric mismatch between the trajectory $\tau(t,R)$ tracked by the low level controller and the actual motion of the UAV, τ_{act}. Specifically, a relationship is established between this mismatch and the maximum speed commanded by $\tau(t,R)$. This relationship captures how higher robot speeds often worsen tracking error of a desired trajectory by the low level controller. This degradation in performance can lead to unsafe situations, especially when the robot is travelling in a cluttered environment. Thus, the risk measure $\rho(\tau(t,R))$ relates the speed of the commanded trajectory to the risk of collision with nearby obstacles.

In order to define this risk measure, first define $d(t_1,t_2)$ as the euclidean distance between a point $\tau_1(t_1)$ on one trajectory and point $\tau_2(t_2)$ on another. The Hausdorff distance $d_H(\cdot)$ between two trajectories τ_1 and τ_2 is defined as:

$$d_H(\tau_1,\tau_2) = \max\left\{ \max_{t_1\in[0,PT]}\left[\min_{t_2\in[0,PT]} d(t_1,t_2)\right], \max_{t_2\in[0,PT]}\left[\min_{t_1\in[0,PT]} d(t_2,t_1)\right]\right\} \tag{11}$$

Fig. 3(b) shows an example of how $d_H(\cdot)$ is found between the commanded trajectory $\tau(t,R)$ and the actual trajectory τ_{act} that results from trying to track $\tau(t,R)$. If $d_H(\tau(t,R),\tau_{\text{act}}) \approx 0$, then both trajectories have considerable overlap in the xyz space over the entire trajectory, while $d_H(\tau(t,R),\tau_{\text{act}}) \gg 0$ signifies at least some portion where there is significant deviation. Through simulation or experiment, data can be collected that measures $d_H(\tau(t,R),\tau_{\text{act}})$ for various commanded trajectories. This data can then be used to train an estimate $\hat{d}_H(\tau(t,R))$ that is only dependent on the commanded trajectory. For the specific UAV application considered in this paper we have observed - as intuitively expected - that the deviation is a function of the maximum speed v_{max} commanded by $\tau(t,R)$. In this way it is possible to predict the tracking error using only information from the commanded trajectory.

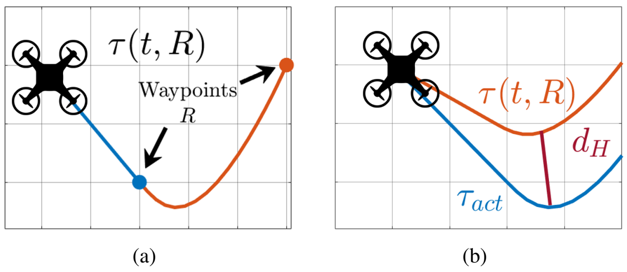


Fig. 3. Example of min-jerk trajectory shown in (a), and example of trajectory tracking error shown in (b).

Denote d_{obs} as the distance between $\tau(t,R)$ and the nearest obstacle. For the UAV, it is considered risky when $d_{\text{obs}} < \hat{d}_H$, since the deviation of the actual trajectory may extend toward the obstacle, potentially colliding with it. Likewise, if $d_{\text{obs}} \geq \hat{d}_H$, then there is no risk of collision, since the robot is expected to deviate from $\tau(t,R)$ by a distance smaller than the nearest obstacle.

To this end, the risk measure is defined as:

$$\rho(\tau(t,R)) = \max\left[0, \frac{\hat{d}_H}{d_{\text{obs}}} - 1\right] \tag{12}$$

In this way, $\rho(\cdot) > 0$ when there exists the potential for τ_{act} to intersect with the boundary of an obstacle, and $\rho(\cdot) = 0$ when $d_{\text{obs}} \geq \hat{d}_H$.

C. MPPI For Motion Planning

MPPI control is a sampling-based control method to find the solution to a stochastic optimal control problem (OCP). In the proposed approach, the MPPI solver is used to find a series of Cartesian waypoint positions $R = \{\boldsymbol{r}_1, \boldsymbol{r}_2\}$ that define a min-jerk trajectory $\tau(t,R)$.

The MPPI algorithm must be defined with respect to some stochasic equations of motion for the state $\boldsymbol{x}$:

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \tau(t_k, R + \mathcal{E})) \tag{13}$$

Here, $\boldsymbol{f}(\cdot)$ represents a discrete-time equation of motion in which the robot $\boldsymbol{x}$ evolves under the influence of a trajectory $\tau(t, R+\mathcal{E})$, where $\mathcal{E} = \{\boldsymbol{\epsilon}_1, \boldsymbol{\epsilon}_2\}$ are random perturbations on the Cartesian xy position of the p^{th} waypoint, with $\boldsymbol{\epsilon}_p \sim \mathcal{N}(0, \Sigma)$.

The stochastic optimal control problem may be defined as the minimization of an expectation value, denoted by $\mathbb{E}(\cdot)$:

$$R^* = \arg\min_R \mathbb{E}_{\mathbb{Q}}\left[S(R+\mathcal{E})\right] \tag{14}$$

The term $S(\cdot)$ defines the cost of the total trajectory, with waypoints R being perturbed by stochastic variables $\mathcal{E}$ that create the probability distribution $\mathbb{Q}$. Sec. IV-C.1 describes the components of this cost function, and Sec. IV-C.2 describes the MPPI algorithm for finding the solution to (14).

1) Cost Function Components

The total cost $S(\cdot)$ is defined over $P = 2$ waypoints as:

$$S(R+\mathcal{E}) = \phi(\boldsymbol{x}(PT)) + \int_0^{PT} \mathcal{C}(\boldsymbol{x}(t))dt \tag{15}$$

The term $\phi(\cdot)$ is a terminal cost function defined as the error between the final state $\boldsymbol{x}(PT)$ and the goal state:

$$\phi(x(PT)) = w_g|\boldsymbol{x}(PT) - \boldsymbol{x}_g| \tag{16}$$

The constant $w_g > 0$ is a scaling factor that is tuned to adjust the relative weight of the different objectives within (15). The running cost $\mathcal{C}(\boldsymbol{x}(t))$ is defined by three terms:

$$\mathcal{C}(\boldsymbol{x}) = \mathcal{C}_{ct}(\boldsymbol{x}) + \mathcal{C}_{obs}(\boldsymbol{x}) + \mathcal{C}_{\rho}(\boldsymbol{x}) \tag{17}$$

The first term $\mathcal{C}_{ct}$ penalizes any violation of state and actuation constraints of the UAV. Because the UAV model is differentially flat, both the state and controls may be expressed as a function of $\boldsymbol{x}$ and its derivatives, meaning constraint violations can easily be checked and heavily penalized. The second term $\mathcal{C}_{obs}$ is an obstacle cost that heavily penalizes collisions with known obstacles in the environment. Since MPPI is a gradient-free method, the exact form of $\mathcal{C}_{obs}$ can be quite sparse with gradient information, such as an indicator function used with an occupancy map. For the obstacle course assumed by this paper, the obstacle cost is defined using an indicator function $I_{o_j}(x)$ that returns 1 when state $\boldsymbol{x}$ lies within obstacle o_j, else it returns zero.

$$\mathcal{C}_{obs} = w_{obs}\sum_{j=0}^{n_o}\int_0^{PT} I_{o_j}(\boldsymbol{x}(t))dt \tag{18}$$

The second term $\mathcal{C}_{\rho}$ is the portion of the cost related to the risk of a trajectory. Since we constrain $\rho(\cdot)$ to be positive semi-definite, the risk cost can be defined as proportional to this risk measure:

$$\mathcal{C}_{\rho} = w_{\rho}\rho(\cdot) \tag{19}$$

As was the case with the obstacle cost, using an MPPI-based approach to solving (14) allows the exact form of $\rho(\cdot)$ to be quite flexible in its definition, since it does not require information about the gradient of $\rho(\cdot)$. For example, $\rho(\cdot)$ may be a function that approximates some notion of risk that may be hard to write by hand, e.g., a learned policy trained on simulated or experimental data.

Inclusion of the risk measure defined by (12) in the cost function has two different effects. First, trajectories are generally planned to be spatially away from obstacles in order to increase d_{obs}. Second, if the robot must move through small gaps in order to reach its goal, then trajectories that command slower motion are preferred in order to decrease d_H. These behaviors are more concretely explored in both simulation (Sec. V) and experiment (Sec. VI) later in the paper.

2) Solving the Stochastic OCP

Authors in [25] show how it is possible to obtain a theoretical *exact* solution to (14). Unfortunately it is impossible to compute directly, but may be approximated using an iterative sampling method. This iterative algorithm relates the $(k+1)^{\text{th}}$ iteration to the k^{th} iteration by:

$$R_{k+1} = R_k + \sum_{i=1}^{N} w(\mathcal{E}_i)\mathcal{E}_i \tag{20}$$

$$w(\mathcal{E}_i) = \frac{1}{\eta}\exp\left[-S(R_k + \mathcal{E}_i)\right] \tag{21}$$

Here, η is a normalization factor to ensure $\sum_i^N w(\mathcal{E}_i) = 1$. Fig. 4 illustrates how this algorithm finds a trajectory around an obstacle with $P = 2$ waypoints. At the k^{th} iteration, the current waypoint locations R_k are subjected to a series of random perturbations $\{\mathcal{E}_i\}$. The blue/green trajectories are the results of these perturbations, with the color corresponding to the weight of the trajectory as determined by (21). The $(k+1)^{\text{th}}$ iteration is found through a weighted sum of these perturbations, so that $\tau(t, R_{k+1})$ has a lower cost than the previous iteration. This procedure is repeated n_{iter} times, and the resulting waypoints $R_{n_{\text{iter}}}$ are applied. In general, n_{iter} is chosen to be as large as possible so that the iterative procedure can reasonably converge to the optimal solution.

D. Ensuring Safety Under Failure

Lastly, it should be noted that due to the receding-horizon nature of the approach, a given trajectory may not be entirely traversed before another trajectory is replanned by the MPPI. Our approach leverages this by designing each trajectory segment shown in Fig. 3(a) with different purposes. The first trajectory segment (blue) commands basic acceleration, deceleration and coasting behaviors of the UAV, and the UAV is expected to track most if not all of this trajectory segment. However, the UAV is *not expected to track the second trajectory segment (orange) under normal operation.* Instead, the MPPI should replan a new trajectory before the UAV reaches the first waypoint, thus ignoring the second trajectory segment altogether. If, however, the MPPI planner should fail, the UAV will continue to follow the second trajectory segment. In order to ensure the safety of the UAV, the second segment is designed so that the UAV safely stops at the second waypoint. This is easily done by setting the boundary velocity and acceleration of the second waypoint to zero, or $v_e^i = a_e^i = 0$ in (6).

V. Simulated Experiments

Simulations were performed to validate the ability of the proposed approach to reduce risk and negotiate obstacles

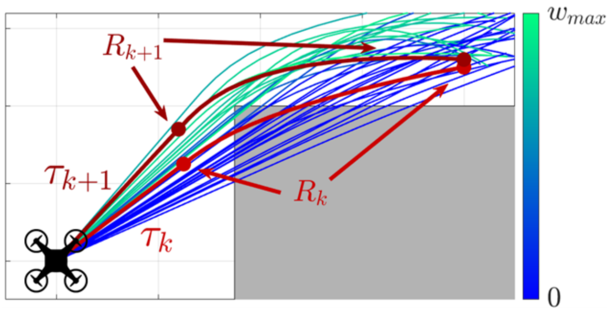


Fig. 4. An example iteration step of the MPPI algorithm.

in a cluttered environment while navigating towards a goal. RotorS [29] was used as a high-fidelity simulator for UAV motion that also includes a low level trajectory nonlinear controller [7]. Given a commanded trajectory $\tau(t)$, the low level controller attempts to track this trajectory, but due to measurement noise and physical limitations the actual trajectory τ_{act} is somewhat different, leading to a non-zero tracking error $d_H(\tau(t), \tau_{\text{act}}(t))$. In order to estimate $d_H(\cdot)$, data were collected by commanding different trajectories $\tau(t)$ and recording the resulting trajectory τ_{act}. Each trajectory was defined by $P = 2$ waypoints placed in the xyz space, with $T = 2.5$ seconds between each waypoints. Noise was injected into the simulated odometry sensor as a way to exacerbate the tracking error of the low level controller.

Fig. 5(a) shows a recorded example in which τ_{act} differs from $\tau(t)$. Also included in this plot is $d_H(\cdot)$ between the two trajectories. Fig. 5(b) shows a plot of $d_H(\cdot)$ as a function of maximum speed $v_{\max}$ along $\tau(t)$. It can be seen that as $v_{\max}$ increases, the set difference $d_H(\cdot)$ also increases. A regression line $\hat{d}_H(v_{max})$ was fit to the data to capture the 95th percentile, giving a conservative estimate of the actual set difference, $\hat{d}_H(v_{max}) \approx d_H(\cdot)$. This estimator of the set difference $\hat{d}_H$ was used in (12) to find an estimate of the risk $\rho(\cdot)$.

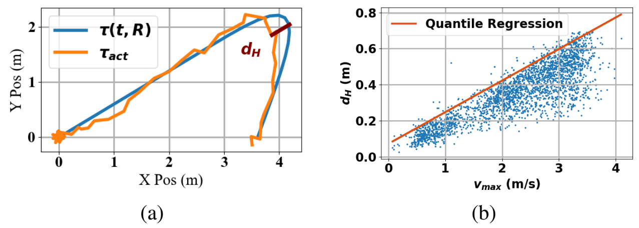


Fig. 5. RotorS UAV trajectory data used for risk cost $\mathcal{C}_\rho$.

The MPPI algorithm that solves (14) was written in C++, using perturbation covariance matrix $\Sigma = diag(0.15, 0.15, 0.0)$ m, $N = 50$ samples per iteration and $n_{\text{iter}} = 200$ total iterations per MPPI sample. The MPPI was run on a Lenovo ThinkPad X1 with Intel i7 6-core processor, and took an average of 0.19 ± 0.03s to run. The algorithm took the current state of the UAV $\boldsymbol{x}_0$ as well as a set of local obstacles $\{\boldsymbol{o}_j\}$ and returned an optimal set of waypoints R^* that defined a trajectory $\tau(t, R^*)$ to be tracked. This trajectory was re-planned in a receding-horizon fashion, with the MPPI being re-sampled at a rate of 1 Hz to find an updated set of waypoints.

To test the capabilities of the MPPI planner, the UAV was tasked with navigating an obstacle course, shown in Fig. 6(a,b), with both small and large gaps to negotiate. To facilitate clockwise motion around the track, goal points were chosen a priori and commanded sequentially as the goal state in the MPPI cost function (16). These goal states also acted as a warm-start for the MPPI algorithm, choosing the initial waypoints R_0 to be along these goal points.

Fig. 6 also shows the result of 20 laps around the obstacle course. Fig. 6(a) visualizes the resulting motion of the UAV as it tracked trajectories commanded by the MPPI. To highlight the effect of the risk objective on the overall motion of the UAV, Fig. 6(b) shows the resulting trajectories with $w_\rho = 0$ in the risk cost objective (19). This effectively removed the consideration of risk within the MPPI when planning motion, instead finding a trajectory that avoided obstacles while moving as quickly as possible. Qualitatively, the difference between these trajectories is only apparent when the UAV approaches small gaps **A** and **B**. With our full risk-aware approach, the UAV slowed down enough to ensure safe passage through these tight spots, whereas the policy with no risk consideration sped through these gaps, resulting in collisions due to tracking error. These collisions are shown visually by the red dots in Fig. 6(b). Alternatively, the risk-aware approach commanded the same high speed as the risk-agnostic policy through corridor **C**, since there were no close obstacles and it was safe to move quickly through this region.

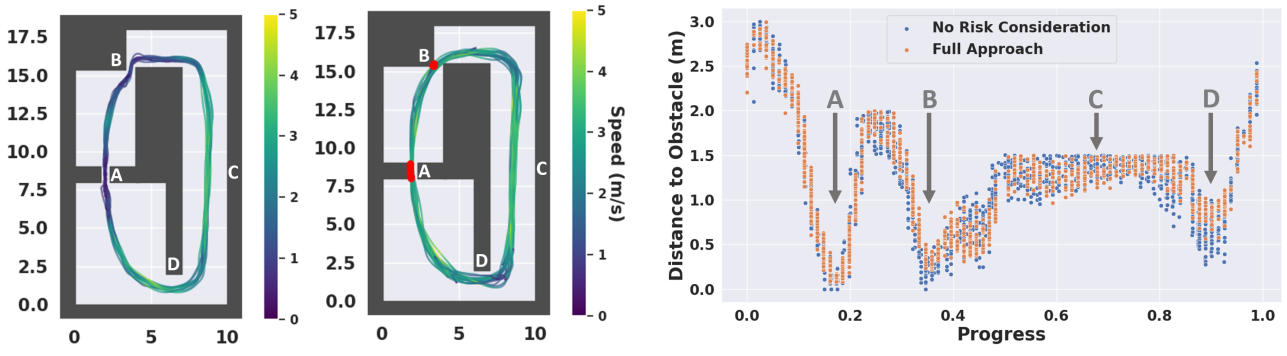
Fig. 6(c) additionally shows the distance between the UAV and the nearest obstacle as it traveled around the track. This distance is plotted against progress along the track, where progress $= 0$ when the UAV was at the start of the course, and progress $= 1$ at the end of the course. This plot shows how the full approach worked to increase the distance between the UAV and obstacles, whereas the MPPI without risk consideration commanded motion closer to obstacles. Over the 20 laps, the full approach with risk consideration had 0 collisions, while motion with no consideration for risk resulted in 4 collisions.

VI. Physical Experiments

The proposed approach was verified experimentally on a Bitcraze Crazyflie quadrotor in two case studies: (i) a rectangular loop and (ii) a 4-way city block. A Vicon motion capture system provided odometry information to an offboard laptop, which then used the MPPI path planner with the same parameters as in simulation to send trajectories to the Crazyflie's nonlinear controller [30]. Similar to the simulated experiments, trajectory tracking data were collected by commanding the Crazyflie to track various trajectories and recording $d_H(\tau(t, R), \tau_{\text{act}})$, which was used to train $\hat{d}_H(\tau(t, R))$. The effect of including the risk measure inside the MPPI cost function is shown by comparing the full approach to the same MPPI with $w_\rho = 0$ in (19).

A. Rectangular Loop Case Study

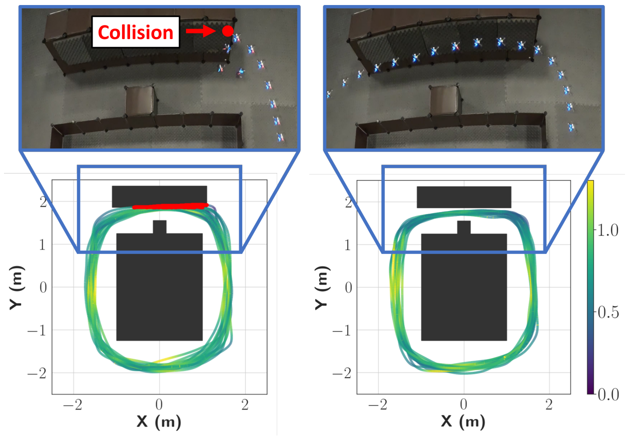
In this case study, the Crazyflie was tasked to complete loops around a central rectangular obstacle while negotiating its way through a narrow 30 cm gap between the northern wall and a protruding square obstacle. The top portion of Fig. 7 shows a snapshot of a sample pass through the narrow gap, both without risk consideration and with our full approach. For these single trajectory examples, the physical



(a) (b) (c)

Fig. 6. Simulation of quadrotor navigating an obstacle course using the MPPI motion planner.

obstacle height was raised to demonstrate a collision without risk consideration. For the rest of data collected, the physical obstacle height was lowered to allow multiple laps around the environment without disruption. The results of the multiple laps are shown in the bottom plots of Fig. 7. For this study, 20 laps were recorded for both the no risk and full approach cases, giving 40 total laps tested. Fig. 7(a) shows how the UAV collided with the north wall 6 times (highlighted in red) when risk was not taken into account due to overshooting the planned MPPI trajectories at high speeds. However, with the full approach (Fig. 7(b)), no collisions occurred because the UAV slowed down at the bends of the loop in order to mitigate the overshooting behavior. Additionally, the UAV achieved comparable speeds both without risk consideration and with our full approach on the east, south and west side of the central square. This demonstrates how our full approach proactively adapted to move quickly through regions where there were no close obstacles, and the risk of collision due to tracking error was minimal.



(a) No risk (b) Full approach

Fig. 7. Crazyflie positions and velocities for the rectangular loop environment, along with snapshots of the physical experiment in our lab.

B. 4-Way City Block Case Study

In the second case study, the UAV was tasked to complete a complex path that involved alternating between straight lines and turning in different directions through a 4-way city block-like circuit. First, the UAV passed through a narrow 20 cm horizontal channel in the center of the configuration (Fig. 8(a)). It then executed a u-turn around the narrow rectangular obstacle in the second quadrant (Fig. 8(b)) and went through a 20 cm vertical passage (Fig. 8(c)) before looping back to the start (Fig. 8(d)). Fig. 9 shows the trajectory of the Crazyflie over 10 laps in both the no risk and full approach cases, giving 20 laps total. As can be seen from the results, in the no risk case (Fig. 9(a)), the UAV collided with obstacles on 8 occasions within the narrow corridors, while in the full approach (Fig. 9(b)) it never collided. The speed profiles also demonstrate that, when obstacles are far enough away, the full approach allowed the UAV to reach the same speeds as with no risk consideration. Furthermore, thanks to our risk-aware framework, the UAV slowed down when traversing the cluttered sections of the environment, allowing for safer navigation that avoided collisions.

VII. Conclusions and Future Work

In this work we have presented a receding-horizon path planning approach that can proactively adapt the trajectory of a robot at run-time in order to reduce overall risk while navigating through a cluttered environment. The proposed approach utilizes an MPPI control algorithm in order to accommodate a general, data-informed risk measure. Importantly, the trajectory planned by the MPPI is parameterized by a few number of variables, greatly reducing the computational requirements to run the MPPI algorithm and allowing the approach to run on more general hardware. The full approach was validated on a UAV robotic system navigating around obstacles towards a goal, with risk defined by the tracking error between commanded trajectory and the actual trajectory. Both simulation and experiment demonstrated how the inclusion of this risk measure inside the cost function allows the robot to move more safely through the environment, compared to motion without risk consideration.

Future work includes deploying this approach in additional robotic contexts, such as drones flying in outdoor environments with more complex sources of risk, or ground vehicles where risk may be associated with human-robot interaction. Additionally, the development of more sophisticated risk measures may facilitate more aggressive motion through cluttered environments, allowing the overall trajectory to be less conservative while ensuring risk is minimized.

VIII. Acknowledgements

This work is based on research sponsored by Amazon Research Award and by DARPA under Contract No. FA8750-18-C-0090.

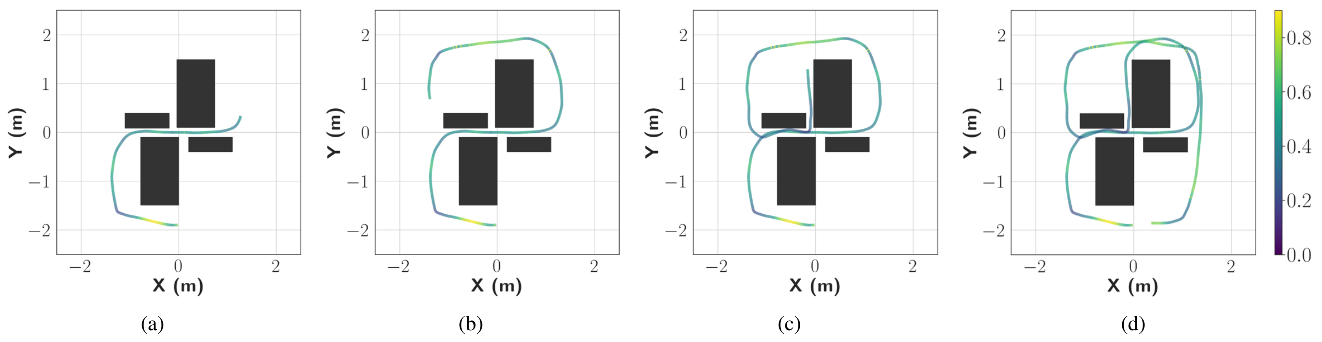


Fig. 8. A demonstration of a single lap that the UAV performs around the 4-way city block environment, along with its velocity profile.

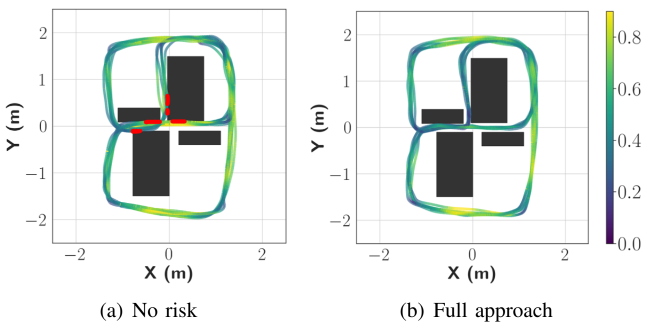


Fig. 9. Crazyflie positions and velocities for the 4-way city block environment (a) without considering risk and (b) with the full approach.

References

[1] J. Tordesillas, B. T. Lopez, and J. P. How, "Faster: Fast and safe trajectory planner for flights in unknown environments," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 1934–1940.

[2] G. Williams, B. Goldfain, P. Drews, K. Saigol, J. M. Rehg, and E. A. Theodorou, "Robust sampling based model predictive control with sparse objective information." in *Robotics: Science and Systems*, 2018.

[3] M. W. Mueller, M. Hehn, and R. D'Andrea, "A computationally efficient motion primitive for quadrocopter trajectory generation," *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1294–1310, 2015.

[4] B. Pozzan, B. Elaamery, and A. Cenedese, "Non-linear model predictive control for autonomous landing of a uav on a moving platform," in *2022 IEEE Conference on Control Technology and Applications (CCTA)*, 2022, pp. 1240–1245.

[5] R. Penicka and D. Scaramuzza, "Minimum-time quadrotor waypoint flight in cluttered environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5719–5726, 2022.

[6] L. Yan, Y. Zhao, H. Xie, J. Dai, Y. Han, and S. Su, "An inert and efficient fusion replanning method for uav fast autonomous flight," in *2022 IEEE International Conference on Unmanned Systems (ICUS)*, 2022, pp. 140–145.

[7] T. Lee, M. Leok, and N. H. McClamroch, "Control of complex maneuvers for a quadrotor uav using geometric methods on se (3)," *arXiv preprint arXiv:1003.2005*, 2010.

[8] H. Shraim, A. Awada, and R. Youness, "A survey on quadrotors: Configurations, modeling and identification, control, collision avoidance, fault diagnosis and tolerant control," *IEEE Aerospace and Electronic Systems Magazine*, vol. 33, no. 7, pp. 14–33, 2018.

[9] J. Jia, K. Guo, X. Yu, W. Zhao, and L. Guo, "Accurate high-maneuvering trajectory tracking for quadrotors: A drag utilization method," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 6966–6973, 2022.

[10] M. Bisheban and T. Lee, "Geometric adaptive control with neural networks for a quadrotor in wind fields," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 4, pp. 1533–1548, 2021.

[11] Z. Zhang and D. Scaramuzza, "Perception-aware receding horizon navigation for mavs," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 2534–2541.

[12] X. Wu, S. Chen, K. Sreenath, and M. W. Mueller, "Perception-aware receding horizon trajectory planning for multicopters with visual-inertial odometry," *IEEE Access*, vol. 10, pp. 87 911–87 922, 2022.

[13] 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.

[14] G. Torrente, E. Kaufmann, P. Föhn, and D. Scaramuzza, "Data-driven mpc for quadrotors," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3769–3776, 2021.

[15] W. Xie, W. Zhang, and C. Silvestre, "Saturated backstepping-based tracking control of a quadrotor with uncertain vehicle parameters and external disturbances," *IEEE Control Systems Letters*, vol. 6, pp. 1634–1639, 2022.

[16] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model predictive control: theory, computation, and design*. Nob Hill Publishing Madison, WI, 2017, vol. 2.

[17] A. Wischnewski, T. Herrmann, F. Werner, and B. Lohmann, "A tube-mpc approach to autonomous multi-vehicle racing on high-speed ovals," *IEEE Transactions on Intelligent Vehicles*, pp. 1–1, 2022.

[18] K. Ren, H. Ahn, and M. Kamgarpour, "Chance-constrained trajectory planning with multimodal environmental uncertainty," *IEEE Control Systems Letters*, vol. 7, pp. 13–18, 2023.

[19] A. Hakobyan, G. C. Kim, and I. Yang, "Risk-aware motion planning and control using cvar-constrained optimization," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3924–3931, 2019.

[20] B. H. Meng, I. S. Godage, and I. Kanj, "Rrt*-based path planning for continuum arms," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 6830–6837, 2022.

[21] M. Dharmadhikari, T. Dang, L. Solanka, J. Loje, H. Nguyen, N. Khedekar, and K. Alexis, "Motion primitives-based path planning for fast and agile exploration using aerial robots," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 179–185.

[22] G. Würsching and M. Althoff, "Sampling-based optimal trajectory generation for autonomous vehicles using reachable sets," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, 2021, pp. 828–835.

[23] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 4569–4574.

[24] 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.

[25] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou, "Information theoretic mpc for model-based reinforcement learning," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 1714–1721.

[26] S. Liu, N. Atanasov, K. Mohta, and V. Kumar, "Search-based motion planning for quadrotors using linear quadratic minimum time control," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 2872–2879.

[27] G. Guennebaud, B. Jacob *et al.*, "Eigen v3," http://eigen.tuxfamily.org, 2010.

[28] D. D. Fan, A.-a. Agha-mohammadi, and E. A. Theodorou, "Learning risk-aware costmaps for traversability in challenging environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 1, pp. 279–286, 2022.

[29] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, *Robot Operating System (ROS): The Complete Reference (Volume 1)*. Cham: Springer International Publishing, 2016, ch. RotorS—A Modular Gazebo MAV Simulator Framework, pp. 595–625. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-26054-9_23

[30] J. A. Preiss, W. Honig, G. S. Sukhatme, and N. Ayanian, "Crazyswarm: A large nano-quadcopter swarm," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 3299–3304.