A Planning-and-Control Framework for Aerial Manipulation of Articulated Objects

A Planning-and-Control Framework for Aerial Manipulation of Articulated Objects

Maximilian Brunner, Giuseppe Rizzi, Matthias Studiger, Roland Siegwart, Marco Tognon

***Abstract*— While the variety of applications for Aerial Manipulators (AMs) has increased over the last years, they are mostly limited to push-and-slide tasks. More complex manipulations of dynamic environments are poorly addressed and still require handcrafted designs of hardware, control, and trajectory planning. In this paper we focus on the active manipulation of articulated objects with AMs. We present a novel planning and control approach that allows the AM to execute complex interaction maneuvers with as little as possible priors given by the operator. Our framework combines sampling-based predictive control to generate pose trajectories with an impedance controller for compliant behaviours, applied to a fully-actuated flying platform. The framework leverages a physics engine to simulate the dynamics of the platform and the environment in order to find optimal motions to execute manipulation tasks. Experiments on two selected examples of pulling open a door and of turning a valve show the feasibility of the proposed approach.**

I. INTRODUCTION

The advancement of aerial robotics in recent years has come with increasing focus on aerial interaction tasks. The underactuated nature of early micro aerial vehicles (MAVs) initially allowed for simple tasks only. Examples are aerial pick-and-place [1], [2] or exertion of light pushing forces against flat and rigid structures for inspection or writing tasks [3], [4]. To provide more flexibility during contact, they can also be equipped with a compliant or articulated robotic arm [5], [6].

The introduction of fully-actuated MAVs has opened a new variety of possible interaction tasks and higher interaction forces. The full actuation of an aerial manipulator (AM) allows it to exert forces and torques (i.e., *wrenches*) in multiple directions. This allows the platform to counteract a larger range of reaction wrenches that appear during an interaction, as well as to decouple linear and angular dynamics. Contact-based inspection and push-and-slide operations with fully-actuated MAVs with an end-effector rigidly connected to the platform have been demonstrated in [7], [8], while [9], [10] employed an articulated robotic arm.

While the community rapidly developed new aerial designs as well as control approaches for aerial physical interaction, the interaction tasks studied so far are limited to *static manipulation* involving only pure touching between the AMs and their environment [11]. On the other hand, *active manipulation*, where the physical interaction leads to a change of the environment, like for the manipulation of articulated objects, still remains a mostly unexplored and open problem. This problem requires to address new challenges: (i) The dynamic behavior of the manipulated object affects the dynamics of the AM, which needs to be accounted for by the controller; (ii) the execution of the task can require or result in multiple switching contacts between the AM and the environment, leading to a recurring change of dynamics; (iii) to generate and execute safe interaction motions, the system must consider the inherent instability as well as geometric and actuation constraints of AMs. In this work, we address these challenges providing a framework for AMs enabling active manipulation of objects in a generalized, optimal, and robust way.

This work was supported by armasuisse S+T, NCCR Robotics, and NCCR Digital Fabrication.

The authors are with the Autonomous Systems Lab at ETH Zurich, Switzerland firstname.lastname@mavt.ethz.ch

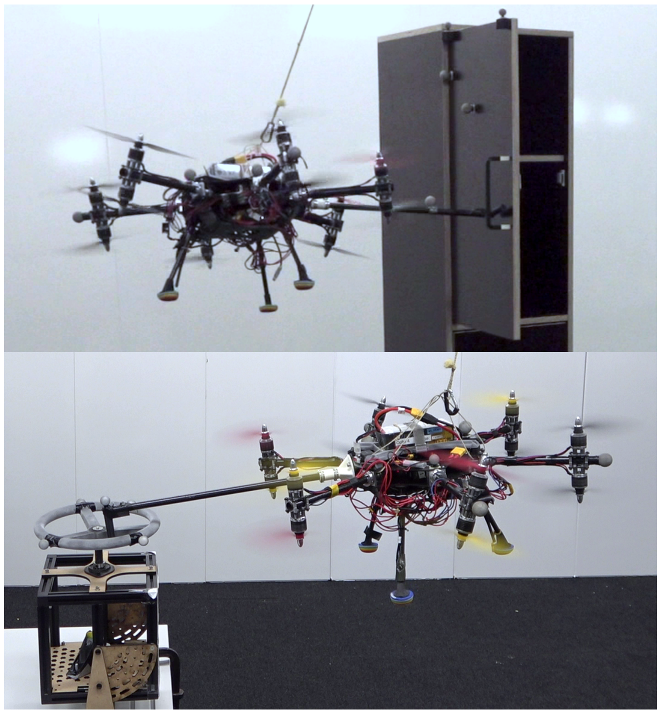
Fig. 1: An Aerial Manipulator equipped with a rigid end-effector autonomously opening a door and turning a valve.

A. Related work

In order to perform aerial manipulation tasks, e.g., opening doors, drawers, or turning valves, the community firstly relied on underactuated platforms equipped with a compliant or articulated robotic arm. Opening a door with an under-actuated platform was firstly achieved in [12], where the authors developed a control strategy for a quadrotor that is able to perch against a door to subsequently open it by generating lift with its propellers. More recently, the work in [13] uses a model predictive controller (MPC) to push open a hinged door with an articulated end-effector (EE). The coupled system AM-door (i.e., its kinematic constraints and dynamics) is modeled using the Lagrange formalism. The MPC then acts as a trajectory generator for the pose of the platform and the robotic arm joints, where only the reference attitude for the AM and the reference door angle is given. This reference pose is tracked by a Disturbance-Observer Based (DOB) controller. While the approach properly models and optimizes the physical interaction, it is limited to the modeled case of pushing a door at an arbitrary contact point. It is also based on the strong assumption of a rigid connection between the end-effector and the door, whereas experiments show that this connection cannot be maintained. The authors already propose to use an impedance controller to tackle this issue in future work.

Another approach to open a door by pushing has been presented in [14]. In this work an aerial manipulator is integrated with a specific sensor and hardware setup to control and detect the physical interaction required to open a door. Positional uncertainties during the interaction are compensated by a compliant non-rigid gripping mechanism. The motion trajectory generation is tightly coupled with an onboard camera and a force sensor to detect the door, and to estimate the robot and contact state.

A solution to opening a drawer has been presented in [15]. In this work, a 2-degrees of freedom (DoF) manipulator and control framework are designed to open and close a drawer whose position and orientation is automatically detected by an onboard camera. The controller equations are directly derived from modeling the coupled system of the manipulator and the drawer.

Finally, the task of turning a horizontal valve has been addressed with an AM able to grasp the valve from the top and and to turn it by executing a yaw-motion. [16] applied this strategy using a quadrotor with a dedicated grasping mechanism. More recently, [17] showed an aerial manipulator designed as an aerial robotic chain, connecting two underactuated MAVs. Once contact to the valve is established through an electromagnetic gripper, the chain of MAVs is commanded a reference angular velocity in order to turn the valve.

All presented earlier works have in common the fact that the control algorithm is specifically designed for the respective task, either by modeling the dynamics analytically or by generating references strongly relying on heuristics. As a result, the controller is mostly only suitable for tasks with low complexity, often requiring further engineering effort to adapt it to slightly different scenarios.

MPC-based methods can reduce the amount of heuristics needed, but they rely on differentiable models which mostly have to be derived analytically. This derivation quickly becomes intractable for certain tasks. In particular, modeling contact forces and kinematic constraints for all possible scenarios during interaction can be infeasible for complex geometries of both object and AM. Recently, sampling-based predictive control approaches, specifically Model Predictive Path Integral Control (MPPI) [18], have been developed which do not rely on analytical models. Instead, trajectories are optimized by repeatedly sampling varying input trajectories (also referred to as *rollouts*) and simulating the resulting system dynamics with a physics engine. By keeping only the most successful rollouts, one obtains optimal trajectories. This approach has been applied to robotic manipulation tasks [19] (e.g., to open drawers), to drone racing with simultaneous vision-based feature detection [20], and to aggressive autonomous driving [18]. However, due to the complex interaction dynamics of manipulation tasks and the unstable nature of AMs, the direct transfer of MPPI to aerial manipulation is not straightforward.

B. Contributions

We address the challenge of enabling autonomous aerial manipulation of articulated objects with a fully-actuated MAV equipped with a robotic end-effector. To this end, we propose a new manipulation controller that combines stochastic optimal control methods, which have already proved successful in mobile manipulation, with a classical impedance controller. We present a control architecture that ensures a high control rate independent of the complexity of the manipulation task at hand, despite potentially long computation times of the optimization.

The sampling-based approach does not require an analytical model of the interaction dynamics and can handle multiple and recurring contacts between the aerial manipulator and the environment. Additionally, it allows for solving different manipulation tasks with only minimum engineering effort.

The approach is applicable — but not limited to — fully-actuated MAVs, characterized by their ability to exert forces and torques in 6 DoF. We present the versatility of the approach on two real-world experiments for the cases of opening a door and turning a valve. We have chosen these tasks as common benchmark manipulation tasks which require both position and attitude changes. Furthermore, instead of using customized manipulators for the individual tasks, we employ a single generic end-effector. In contrast to earlier works, this only allows non-rigid point-contacts with its environment, making the interaction more challenging. Finally, we show the reactivity of the method by applying external disturbances that move the end-effector away from the object. Still, the robot is able to re-gain contact and to accomplish the task.

II. Modeling

We consider the challenging task of manipulating articulated objects with an AM. To address this problem, we consider the following assumptions:

- The AM consists of a fully-actuated MAV with a rigidly attached end-effector which enables it to interact with its environment.
- Thanks to the full actuation of the AM, it can be modeled as a rigid body floating in free space, controlled by arbitrary bounded forces and torques (combined as *wrenches*). This is a common simplification in the related state of the art [7].

- The dynamics of both the AM and the object, as well as any present kinematic constraints, are known. We leave the estimation of the object state and its dynamics to future works.

In this section, we first introduce a generic model of the aerial manipulator, as well as the object that is to be manipulated.

A. Aerial manipulator

We use the following frames: The inertial world frame is described by $\mathcal{F}_W = \{O_W, \boldsymbol{x}_W, \boldsymbol{y}_W, \boldsymbol{z}_W\}$ and the frame fixed to the AM body is denoted by $\mathcal{F}_B = \{O_B, \boldsymbol{x}_B, \boldsymbol{y}_B, \boldsymbol{z}_B\}$.

We model the AM as a rigid body system which can exert bounded forces and torques in arbitrary directions. The mass of the platform is given by m and its inertia by $\boldsymbol{I} = \text{diag}\begin{bmatrix} I_{xx} & I_{yy} & I_{zz}\end{bmatrix}$. We combine the mass and inertia into a single matrix called total inertia, $\boldsymbol{J} = \text{diag}\begin{bmatrix} m & m & m & I_{xx} & I_{yy} & I_{zz}\end{bmatrix}$. We express the AM pose by $\boldsymbol{q} \in \text{SE}(3)$, with its position given by $\boldsymbol{r} \in \mathbb{R}^3$ in $\mathcal{F}_W$, its attitude by $\boldsymbol{R}_B \in \text{SO}(3)$, and its twist (i.e., the stacked linear and angular velocity) as $\boldsymbol{t} = \begin{bmatrix}\boldsymbol{v}^\top & \boldsymbol{\omega}^\top\end{bmatrix}^\top \in \mathbb{R}^6$ expressed in $\mathcal{F}_B$. The AM is controlled by wrench inputs $\boldsymbol{w}_c \in \mathbb{R}^6$ which are translated through an *allocation method* into actuator commands for both rotor velocities and tilt angles [21]. The AM dynamics can be modeled following the common Newton-Euler approach[1]:

$$\dot{\boldsymbol{r}} = \boldsymbol{R}_B \boldsymbol{v} \tag{1}$$

$$\dot{\boldsymbol{R}}_B = \boldsymbol{R}_B [\boldsymbol{\omega}]_\times \tag{2}$$

$$\boldsymbol{J}\dot{\boldsymbol{t}} + \boldsymbol{C}(\boldsymbol{t})\boldsymbol{t} + m\boldsymbol{g} = \boldsymbol{w}_c + \boldsymbol{w}_{ext}, \tag{3}$$

with $\boldsymbol{g} \in \mathbb{R}^6$ as the gravity vector in $\mathcal{F}_B$ and $\boldsymbol{w}_{ext} \in \mathbb{R}^6$ representing any external wrenches. The cross terms matrix $\boldsymbol{C}(\boldsymbol{t}) \in \mathbb{R}^{6\times 6}$ captures kinematic effects:

$$\boldsymbol{C}(\boldsymbol{t}) = \begin{bmatrix} [\boldsymbol{\omega}]_\times & \boldsymbol{0}_{3\times 3} \\ \boldsymbol{0}_{3\times 3} & -[\boldsymbol{I}\boldsymbol{\omega}]_\times \end{bmatrix}. \tag{4}$$

Defining the states of the AM as $\boldsymbol{x} = \begin{bmatrix}\boldsymbol{q}^\top, \boldsymbol{t}^\top\end{bmatrix}^\top$ we can then write its dynamics as $\dot{\boldsymbol{x}} = f_R(\boldsymbol{x}, \boldsymbol{w}_c, \boldsymbol{w}_{ext})$.

B. Articulated object

We describe the state of the object by $\boldsymbol{o} \in \mathbb{R}^{n_o}$, with n_o as the number of DoF of the object. In this paper we consider two different objects, namely a door and a valve. As both have a single revolute joint, we have in both cases $n_o = 1$, and we use an angle to describe their state, with $o_d \in [0, 90°]$ for the door and $o_v \in \mathbb{R}$ for the valve.

III. Model Predictive Path Integral Control

In order to execute a manipulation task, different approaches could be devised: (i) Heuristic handcrafted trajectory planning according to the task, as it has been applied in [14] to pull open a door. Depending on the task complexity, this approach can provide a feasible trajectory for performing the task. In case of more complex scenarios with possibly multiple contact points or changes of dynamics, it can quickly turn out infeasible and not robust. (ii) Model Predictive Control (MPC): This method can optimize the direct control inputs or the reference trajectory to achieve a desired object state. The advantage of using MPC is that — if the model and the objective function are well designed — it can result in optimal trajectories that do not require heuristics about the optimal interaction procedure. However, it relies on analytical models of the platform, the object, and the interaction behavior. [13] applied this method to push open a door, where the connection between the AM and the door were modeled by a kinematic constraint. Transitions between free flight and interaction, resulting in varying dynamics, have to be specifically distinguished and handled in the controller.

As opposed to analytical MPC, we follow a *sampling-based* model predictive approach. Instead of relying on differentiable system dynamics, this method samples input trajectories according to a probability distribution and applies them to a dynamic model to generate the corresponding state trajectories. We then employ methods from *Stochastic Control* to find an input distribution that optimizes an objective function.

Mathematical formulation of MPPI

Model Predictive Path Integral Control (MPPI) [19], [22] is a control method that aims to minimize a cost function represented by the cumulative expected cost over an infinite future time horizon. We define $h : \mathcal{X} \times \mathcal{X}^* \times \mathcal{U} \to \mathbb{R}_{\geq 0}$ as the objective function to minimize, with $\boldsymbol{x} \in \mathcal{X}$ representing the system states, $\boldsymbol{x}^* \in \mathcal{X}^*$ the state reference, $\boldsymbol{u} \in \mathcal{U}$ the inputs to the system, and $f : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ a continuous function that describes the state dynamics. We want to find an optimal control policy that minimizes the expected infinite-time horizon cost starting at time t_0, given that the inputs are distributed according to a feedback policy distribution $\boldsymbol{u}_t \sim \pi_{\boldsymbol{\theta}}$:

$$\begin{aligned} \min_{\boldsymbol{\theta}} \quad & \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[\int_{t_0}^{\infty} h(\boldsymbol{x}_t^*, \boldsymbol{x}_t, \boldsymbol{u}_t)\, dt\right] \\ \text{s.t.} \quad & \dot{\boldsymbol{x}}_t = f(\boldsymbol{x}_t, \boldsymbol{u}_t) \\ & \boldsymbol{u}_t \sim \pi_{\boldsymbol{\theta}}(\boldsymbol{x}_t) \\ & \boldsymbol{x}_t \in \mathcal{X},\ \boldsymbol{u}_t \in \mathcal{U}, \end{aligned} \tag{5}$$

where $\mathbb{E}_{\pi_{\boldsymbol{\theta}}}(\cdot)$ is the expectation value of the infinite time horizon cost and $\boldsymbol{\theta}$ is a vector that parameterizes the distribution $\pi_{\boldsymbol{\theta}}$.

The problem (5) can be simplified and rewritten as a discrete finite time horizon problem with N prediction steps and discretization time δt, resulting in a prediction horizon of $T = N \cdot \delta t$. Let a sequence of control inputs be $U_t = \{\boldsymbol{u}_t, \boldsymbol{u}_{t+\delta t}, \ldots, \boldsymbol{u}_{t+(N-1)\delta t}\}$, where the input sequence is sampled from a distribution $U_t \sim \left\{\pi_{\boldsymbol{\theta}_t}, \pi_{\boldsymbol{\theta}_{t+\delta t}}, \ldots, \pi_{\boldsymbol{\theta}_{t+(N-1)\delta t}}\right\} = \pi_{\boldsymbol{\theta}}$. We define the vector $\boldsymbol{\theta}$ as the collection of all distribution parameters for each timestep, i.e., $\boldsymbol{\theta} = \left\{\boldsymbol{\theta}_t, \boldsymbol{\theta}_{t+\delta t}, \ldots, \boldsymbol{\theta}_{t+(N-1)\delta t}\right\}$. Furthermore, let $X_t = \{\boldsymbol{x}_t, \boldsymbol{x}_{t+\delta t}, \ldots, \boldsymbol{x}_{t+T}\}$ be the sequence of

[1]We use $[\cdot]_\times : \mathbb{R}^3 \to \mathfrak{so}(3)$ to denote the skew-symmetric operator and the *vee*-map $(\cdot)^\vee : \mathfrak{so}(3) \to \mathbb{R}^3$ for its inverse.

states obtained from applying the input sequence U_t to the system. The expected cost in (5) can then be approximated by

$$\mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[\int_{t_0}^{\infty} h(\boldsymbol{x}_t^*, \boldsymbol{x}_t, \boldsymbol{u}_t)\ dt\right] \approx \underbrace{\mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[c_{term}(\boldsymbol{x}_{t+T}^*, \boldsymbol{x}_{t+T}) + \sum_{t=t_0}^{t_0+T-\delta t} c(\boldsymbol{x}_t^*, \boldsymbol{x}_t, \boldsymbol{u}_t))\right]}_{J(X_t,U_t)}. \tag{6}$$

The time-discretized cost function is represented by $c(\boldsymbol{x}_t^*, \boldsymbol{x}_t, \boldsymbol{u}_t)$ as the stage cost and the terminal cost $c_{term}(\boldsymbol{x}_{t+T}^*, \boldsymbol{x}_{t+T})$.

We minimize (6) by following a Bayesian approach similar to [23]. We map the expected cost through an exponential function to a *pseudo success likelihood* $\mathcal{J}(X_t, U_t) = \exp(-\lambda J(X_t, U_t))$, where λ determines the scaling between the cost function and the success likelihood. We then find the optimal distribution parameters which maximize the expected log-likelihood of the resulting trajectory through stochastic gradient descent (SGD):

$$\boldsymbol{\theta}^{i+1} = \boldsymbol{\theta}^i + \rho \nabla_{\boldsymbol{\theta}} \log \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[\mathcal{J}(X_t, U_t)\right], \tag{7}$$

where ρ is the step size. We model the input distribution $\pi_{\boldsymbol{\theta}}$ as a multivariate Gaussian distribution $\pi_{\boldsymbol{\theta}_t} = \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma})$ with $\boldsymbol{\mu}_t = \boldsymbol{\theta}_t$ and constant diagonal variance $\boldsymbol{\Sigma} \in \mathbb{R}^{n_u \times n_u}$. This choice ensures that consecutive trajectories do not exhibit a too high variability. The input mean is initialized at zero and then warm started from the previous optimization iteration.

IV. Planning and Control Framework

We aim to use the proposed MPPI controller to execute aerial manipulation tasks in an optimal way. In this regard we face the following challenge: An AM is a highly dynamic system with — depending on its configuration — a large number of control inputs. Its inherent instability commonly requires a consistently high control rate of ≥ 100 Hz. On the other hand, the complexity of solving the MPPI problem highly depends on the size of its state and input space, as well as on the speed at which the system dynamics can be computed. As a result, optimization times can be too slow or have a too high variance, prohibiting the optimization of direct actuator commands.

In order to mitigate the influence of varying optimization times and to maintain a high control rate, we propose a cascaded control architecture: (i) The MPPI controller acts as a trajectory planner to generate state references for the AM. It updates the policy through (7) continuously and always uses the most recent state estimate as initial state. It finds optimal trajectories by sampling reference accelerations and outputs the most recent optimal acceleration rollout at a fixed rate. (ii) Each reference acceleration rollout is integrated twice to obtain a full state reference trajectory. (iii) The optimized reference trajectories are tracked by an *impedance controller* which runs at a constant high rate. This not only ensures a sufficiently high rate to control the platform, but it also renders its dynamics as a 6-DoF impedance, ensures compliance with the environment, and simplifies the dynamics used in the MPPI. Figure 2 illustrates the entire control block diagram.

A. Impedance control

The impedance controller ensures that the AM closed-loop dynamics can be rendered as the dynamics of a second-order system with a desired impedance. This is driven by the external wrench $\boldsymbol{w}_{ext} \in \mathbb{R}^6$ and a reference state $\boldsymbol{x}_r$ that comprises the reference pose $\boldsymbol{q}_r \in \mathrm{SE}(3)$, twist $\boldsymbol{t}_r \in \mathbb{R}^6$, and acceleration $\boldsymbol{a}_r \in \mathbb{R}^6$, respectively. The desired closed-loop error dynamics are:

$$\boldsymbol{J}_v \dot{\boldsymbol{t}}_e + \boldsymbol{D}_v \boldsymbol{t}_e + \boldsymbol{K}_v \boldsymbol{q}_e = \boldsymbol{w}_{ext}, \tag{8}$$

which is characterized by the virtual total inertia $\boldsymbol{J}_v \in \mathbb{R}^{6\times 6}$, the virtual damping $\boldsymbol{D}_v \in \mathbb{R}^{6\times 6}$, and the virtual stiffness $\boldsymbol{K}_v \in \mathbb{R}^{6\times 6}$. The pose, twist, and acceleration errors are computed according to the geometric tracking controller [24]:

$$\begin{aligned} \boldsymbol{q}_e &= \begin{bmatrix} \boldsymbol{r}_e \\ \boldsymbol{\theta}_e \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_B^\top(\boldsymbol{r} - \boldsymbol{r}_r) \\ \frac{1}{2}\left(\boldsymbol{R}_{B,r}^\top \boldsymbol{R}_B - \boldsymbol{R}_B^\top \boldsymbol{R}_{B,r}\right)^\vee \end{bmatrix} \\ \boldsymbol{t}_e &= \begin{bmatrix} \boldsymbol{v}_e \\ \boldsymbol{\omega}_e \end{bmatrix} = \begin{bmatrix} \boldsymbol{v} - \boldsymbol{v}_r \\ \boldsymbol{\omega} - \boldsymbol{R}_B^\top \boldsymbol{R}_{B,r} \boldsymbol{\omega}_r \end{bmatrix} \\ \dot{\boldsymbol{t}}_e &= \dot{\boldsymbol{t}} - \boldsymbol{a}_r. \end{aligned} \tag{9}$$

Assuming that we have an estimate of the external wrench $\hat{\boldsymbol{w}}_{ext}$ (e.g., through a disturbance observer), we can then compute the impedance control law as

$$\begin{aligned} \boldsymbol{w}_c = \left(\boldsymbol{J}\boldsymbol{J}_v{}^{-1} - \mathbf{I}_6\right) \hat{\boldsymbol{w}}_{ext} - \boldsymbol{J}\boldsymbol{J}_v{}^{-1}(\boldsymbol{D}_v \boldsymbol{t}_e + \boldsymbol{K}_v \boldsymbol{q}_e) \\ + \boldsymbol{C}(\boldsymbol{t})\boldsymbol{t} + \boldsymbol{J}\boldsymbol{a}_r + m\boldsymbol{g}, \end{aligned} \tag{10}$$

which, when applied to (3), results in the desired dynamics (8). We write the dynamics of the impedance-controlled AM in a more compact form as a function of the reference states and external wrenches:

$$\dot{\boldsymbol{x}} = f_{imp}(\boldsymbol{x}, \boldsymbol{x}_r, \boldsymbol{w}_{ext}). \tag{11}$$

Remark: During interaction with articulated objects, the external wrench is a function of the AM and object states, i.e., $\boldsymbol{w}_{ext} = f_o(\boldsymbol{x}, \boldsymbol{o})$.

B. State augmentation with reference states

With $\boldsymbol{a}_r = \begin{bmatrix} \dot{\boldsymbol{v}}_r^\top & \dot{\boldsymbol{\omega}}_r^\top \end{bmatrix}^\top$ as reference acceleration we can derive the dynamics of the reference states which are simply given by the double integration:

$$\boldsymbol{t}_r(t) = \begin{bmatrix} \boldsymbol{v}_r \\ \boldsymbol{\omega}_r \end{bmatrix} = \int_0^t \boldsymbol{a}_r(\tau) d\tau \tag{12a}$$

$$\boldsymbol{r}_r = \int_0^t \boldsymbol{v}_r(\tau) d\tau \tag{12b}$$

$$\boldsymbol{R}_{B,r} = \int_0^t \boldsymbol{R}_{B,r}(\tau) \left[\boldsymbol{\omega}_r(\tau)\right]_\times d\tau, \tag{12c}$$

which can be written as the reference state dynamics

$$\dot{\boldsymbol{x}}^* = f_r(\boldsymbol{x}^*, \boldsymbol{a}_r). \tag{13}$$

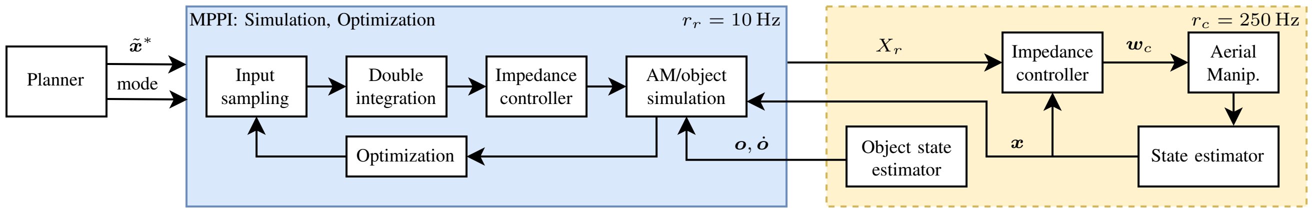

Fig. 2: Control block diagram including the MPPI optimizer and the impedance-controlled system. At each timestep t_r the MPPI releases the optimal reference trajectory $X_{r,t_r} = [\boldsymbol{x}_r(t_r), \ldots, \boldsymbol{x}_r(t_r + T)]$. The impedance control loop, represented by the block on the right, runs at a constant rate of r_c.

We introduce an augmented state vector that comprises the AM states $\boldsymbol{x}$, the reference state $\boldsymbol{x}_r$, and the object state $\boldsymbol{o}$:

$$\tilde{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{x}^\top & \boldsymbol{x}_r^\top & \boldsymbol{o}^\top \end{bmatrix}^\top, \tag{14}$$

and write the augmented system dynamics as

$$\dot{\tilde{\boldsymbol{x}}} = \tilde{f}(\tilde{\boldsymbol{x}}, \boldsymbol{a}_r). \tag{15}$$

Note that the dynamics are now purely driven by the reference acceleration $\boldsymbol{a}_r$. As we generally aim to control either the AM pose or the object states, we also introduce the augmented state reference

$$\tilde{\boldsymbol{x}}^* = \begin{bmatrix} \boldsymbol{q}^{*\top} & \boldsymbol{o}^{*\top} \end{bmatrix}^\top. \tag{16}$$

Finally, we can use the augmented system dynamics (15) to find optimal trajectories that bring the system to the desired states represented by (16).

C. Objective function

We employ multiple objective costs to construct the overall cost function $c : \mathcal{X} \times \mathcal{X}^* \times \mathcal{U} \to \mathbb{R}$ as presented in (6).

1) Pose cost: We use a pose cost function in order to track a reference pose $\boldsymbol{q}^*$ at all times. The reference pose is used to navigate the platform to the point of interaction and to maintain a certain orientation during the interaction. Note that $\boldsymbol{q}^*$ differs from $\boldsymbol{q}_r$ as it is given by the user or another higher level planner, whereas $\boldsymbol{q}_r$ is generated by the MPPI.

$$c_q = (\boldsymbol{q} - \boldsymbol{q}^*)^\top \boldsymbol{Q}_q (\boldsymbol{q} - \boldsymbol{q}^*), \tag{17}$$

with $\boldsymbol{Q}_q = \text{blkdiag}\left(Q_{pos}\mathbf{I}_3, Q_{att}\mathbf{I}_3\right)$ to distinguish between position and attitude weights. We employ high weights during approach and retreat motions to improve free-flight position accuracy, while lower weights (nearly zero) during interaction.

2) Object cost: The object cost is simply driven by the difference of the object state and its reference:

$$c_o = (\boldsymbol{o} - \boldsymbol{o}^*)^\top \boldsymbol{Q}_o (\boldsymbol{o} - \boldsymbol{o}^*). \tag{18}$$

3) Distance to object: In order to provide some heuristic information about the location of interaction we use a cost component that penalizes the distance of the end-effector from the object $d = \|\boldsymbol{r}_{EE} - \boldsymbol{r}_o\|$ when a certain threshold d_{min} is reached:

$$c_d = \begin{cases} 0 & \text{if } d < d_{min}, \\ Q_d(d - d_{min}) & \text{else.} \end{cases} \tag{19}$$

4) Force reduction: Furthermore, we penalize large interaction forces between the AM and the object in order to avoid too aggressive or infeasible interaction:

$$c_f = Q_f \boldsymbol{f}_i^\top \boldsymbol{f}_i, \tag{20}$$

where $\boldsymbol{f}_i$ represents any interaction force. Note that $\boldsymbol{f}_i$ is a function of the augmented system state (14) and can be extracted from the rollout dynamics.

5) Orthogonality: Specifically for the task of opening a door, we introduce a cost component which aims to maintain an orthogonal angle between the AM end-effector and the door during interaction. This helps to guide the platform towards opening the door by a pulling motion and to reduce torques on the body center and is favorable due to the relatively limited yaw torque of our experimental platform.

$$c_{or} = Q_{or}(1 - \cos(\gamma)), \tag{21}$$

where $\gamma \in [0, \frac{\pi}{2}]$ is the angle between the door surface normal and the rod.

D. MPPI as a control-aware trajectory planner

The MPPI optimizer solves the SGD in (7) by continuously sampling input trajectories U_t, given the most recent optimal distribution $\pi_{\boldsymbol{\theta}}$. The corresponding state rollouts X_t are then obtained by applying the inputs $\boldsymbol{u} = \boldsymbol{a}_r$ to the augmented state dynamics (15) which are implemented in the physics engine RaiSim [25]. Using the resulting state rollout and the reference given by (14), we apply the cost functions to compute the expected cost and update the optimal distribution parameters. At each arrival of a new state estimate the initial time t_0 as well as initial state $\boldsymbol{x}_0$ are reset to the most recent odometry. Since the MPPI simulates the closed-loop system (both robot dynamics and impedance controller), it acts as a *control-aware trajectory planner*.

Depending on the complexity of the geometry and collisions between objects, the time Δt_{opt} that is required to obtain a new reference trajectory $X_{r,t}$ can vary significantly and is a priori unknown. The MPPI needs to account for the fact that the impedance controller will only receive a new trajectory after this uncertain amount of time. To this end, we define a constant rate r_r at which trajectories are released to the impedance controller. This rate is chosen low enough to allow the MPPI enough time budget to finish an iteration even in complex scenarios. In the MPPI we ensure that the reference trajectory does not change before the next

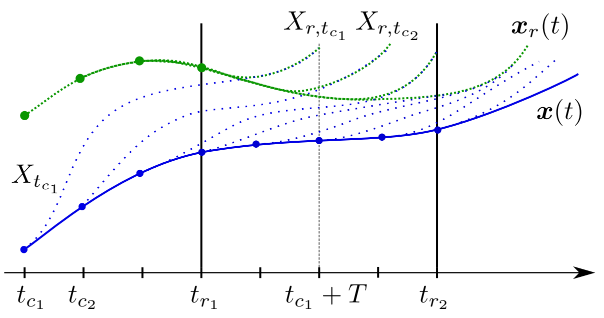

Fig. 3: Illustration of actual pose (blue lines) and pose reference rollouts (green lines), represented by $X_{t_{c_i}}$ and $X_{r,t_{c_i}}$, respectively. At each state estimate update at time t_{c_i}, new rollouts are generated. Dotted lines represent the simulated state and reference state rollouts. The continuous blue line represents the finally executed state trajectory $\boldsymbol{x}(t)$. Reference state rollouts are only allowed to vary after the next release, scheduled at each t_r. The concatenation of reference states results in the final reference trajectory, given by $\boldsymbol{x}_r(t)$.

scheduled release of a new trajectory at time t_{r_2} by setting the covariance to zero, i.e., $\boldsymbol{\Sigma}(t) = \mathbf{0}\ \forall\ t \in [t_{r_1}, t_{r_2}]$. That way, we can ensure that the reference trajectory $X_{r,t}$ is equal in both the simulated rollout and the real system at all times. Figure 3 illustrates this mechanism to ensure smooth trajectories despite unknown optimization times. The final trajectory results from a concatenation of optimal trajectories $X_{r,t_{r_i}}$.

E. Cost scheduling/state machine

In order to enable a fully autonomous execution of the task, we use a simple state machine to transition between different phases.

1) During the *approach* phase only the pose cost is active and the pose reference $\boldsymbol{q}^*$ is set to a point close to the point of interaction.
2) During the *interaction* phase the pose cost is reduced and all other costs relevant for the task execution are enabled.
3) During the *retreat* phase the weights are the same as during the approach, with the pose reference set to a point in sufficient distance from the object.

The state machine is triggered manually, it could however be automatically triggered by the value of the cost function. This state machine is represented in Fig. 2 by the *planner* control block. The state of the *mode* (either *free flight* or *interaction*) determines the weights applied in the optimizer.

V. Experiments

In this section we evaluate the proposed framework based on two selected interaction tasks: 1) Opening a door by pulling on its handle, and 2) turning a valve around its vertical rotation axis. Both tasks are performed using a custom built AM with a rigid end-effector (see Fig. 1).

The AM is based on a symmetric hexacopter design, with 6 arms carrying double rotor groups. Each arm can be tilted individually through six separate servos that are mounted in the center of the platform [7]. The end-effector is mounted at the end of a rod in a horizontal distance of 60 cm from the body center and has the shape of a hook with a length of 5 cm, pointing vertically down. Both the rod and the hook are made from carbon fiber, only allowing small compliance through bending.

TABLE I: Impedance parameters.

Param	Value
m_v	$m = 4.4\,\mathrm{kg}$
$\boldsymbol{I}_v$	$\boldsymbol{I} = \mathrm{diag}\,[0.07, 0.08, 0.15]\,\mathrm{kg\,m^2}$
$\boldsymbol{K}_v$	$\mathrm{blkdiag}\,(15\mathbf{I}_3, 60\mathbf{I}_3)$
$\boldsymbol{D}_v$	$\mathrm{blkdiag}\,(5\mathbf{I}_3, 15\mathbf{I}_3)$

TABLE II: MPPI weights used in experiments.

Param	Free flight	Door	Valve
Q_d	0.0	100	100
d_{min} [m]	n/a	0.03	0.08
Q_f	0.0	1.0	1.0
Q_o	0.0	150	1.0
Q_{or}	n/a	100	n/a
Q_{pos}	20	0.0	0.0
Q_{att}	10	0.0	10

The MPPI solver runs on an onboard Intel NUC computer, generating trajectories ideally at a rate of $r_r = 10\,\mathrm{Hz}$. State estimation fuses pose measurements through a VICON motion capture system and onboard IMU measurements. The impedance controller runs at the same rate as state estimation at $r_c = 250\,\mathrm{Hz}$. It computes wrench commands which are sent to an onboard Pixhawk flight controller, where the allocation is performed to obtain the actuator commands. The object states of either the door or the valve angle are obtained through VICON measurements. The MPPI sampling time is set to $\delta t = 0.015\,\mathrm{s}$ and the rollout length is set to $N = 66$, resulting in a horizon length of $T = 0.99\,\mathrm{s}$. State rollouts are simulated in RaiSim, running 8 threads simultaneously.

For all experiments we use the same impedance controller settings with parameters listed in Table I. Note that by setting $\boldsymbol{J}_v = \boldsymbol{J}$ we do not apply any external wrench estimates in the controller. The optimization weight parameters are listed in Table II.

A. Door opening

For the task of opening a door we consider a hinged shelf door with a handle and only small friction in its hinge joints. Note that both the door handle and the hook on the AM are aligned vertically which requires the platform to tilt in order to allow a proper grasp. We specify the object reference as constant at $o_d^* = 90°$ (where $o_d = 0°$ is fully closed and 90° is fully open). The approach phase commands a single reference pose that brings the end-effector close to the handle, after which the MPPI mode is switched to *interaction*. After a fixed duration of 20 s the mode is switched back to free flight and a single retreat pose is commanded.

Figure 4 shows three stages during the task execution. The MPPI generates a trajectory which leads to a slight roll of the AM such that it can insert the hook behind the door handle. After having inserted the hook, the position

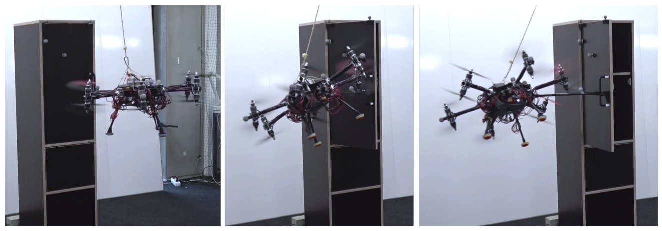

Fig. 4: Three steps during the opening of a door with a vertically aligned handle.

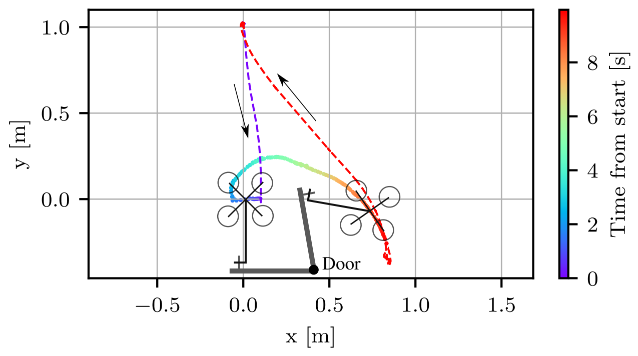

Fig. 5: Top-down view of the motion during the opening of the door. The approach and retreat trajectories are marked in dashed purple and red, respectively. The colorbar shows the time from the start of the manipulation task.

trajectory resembles a circular motion in a horizontal plane which pulls the door open, maintaining contact between the door handle and the hook. Figure 5 shows the entire trajectory in a top-down view, including the approach and the retreat phase. Figure 6 presents the evolution of the individual cost components and the tracking of the door reference angle. Once the interaction mode is enabled, the object cost c_o causes the opening of the door. During this motion the orthogonality cost c_{or} ensures that the AM and the door maintain an orthogonal angle. Although the distance and force cost, c_d and c_f, are too small to be seen in the figure, they are crucial for a proper execution: c_d ensures that the hook remains close to the door handle, guiding the MPPI towards a feasible solution. c_f keeps interaction forces in a feasible range, avoiding too aggressive maneuvers. Also note the pose cost before and after the interaction, causing the approach and retreat of the AM to and away from the door.

B. Valve turning

We use a custom built mock-up valve with a diameter of 24 cm and unlimited rotation range. It is set up with a vertical rotation axis, as shown in Fig. 1. Similar to the previous experiment, after an approach phase we activate the interaction mode and set the valve angle reference to either a constant value or a dynamically updated value for continuous rotation. Accordingly, we perform two experiments.

1) We command alternating angle references of $6\,\text{rad} \approx 344°$ and $0\,\text{rad}$, respectively. Figure 7 shows the successful reference angle tracking in six consecutive valve rotations. The robot turns the valve by inserting the hook in the valve plane and then performing a circular position trajectory, thereby rotating the valve by pushing and pulling horizontally on its spokes.
2) We command a continuous valve rotation and disturb the end-effector by manually lifting it off the valve with a stick. Figure 8 shows that the AM always successfully resumes the task after each interruption.

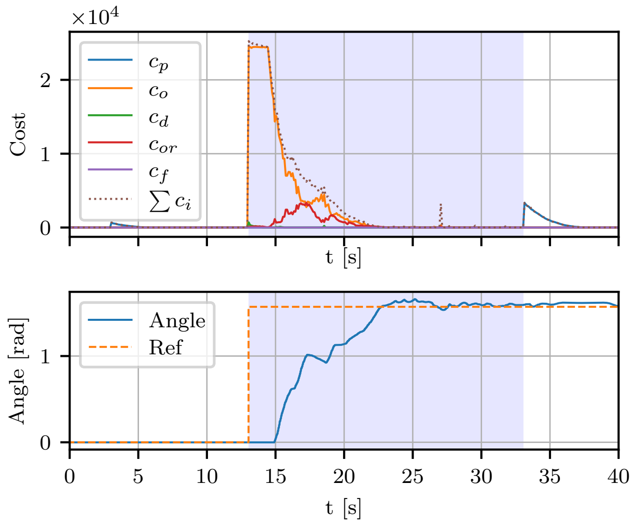

Fig. 6: Door cost components (top) and door angle tracking (bottom). The area highlighted in blue represents the period during which the interaction mode is enabled.

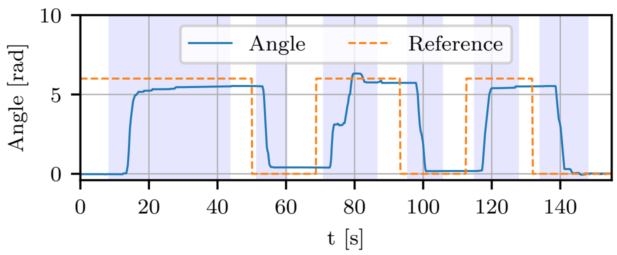

Fig. 7: Valve rotations with alternating reference angles of $6\,\text{rad} \approx 344°$ and $0\,\text{rad}$. The references and interaction modes are set manually.

Figure 9 illustrates the optimization times Δt_{opt} required by the MPPI to finish its iterations and the time differences between the releases of subsequent trajectories. Δt_{opt} significantly increases during contact, but always stays well below the trajectory length T, avoiding outages of references for the impedance controller.

VI. Conclusion

In this work we provide a framework that enables the robust execution of complex manipulation of articulated objects with AMs. This is achieved by combining recent advances in sampling-based optimal control with a classical impedance controller. The control architecture autonomously generates optimal trajectories for an AM to manipulate an articulated object and to bring it to a desired state. The dynamics are embedded in a physics engine which

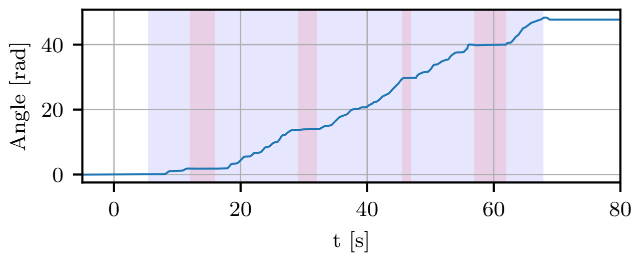

Fig. 8: Continuous valve rotation with disturbances. The areas highlighted in red represent the disturbance periods.

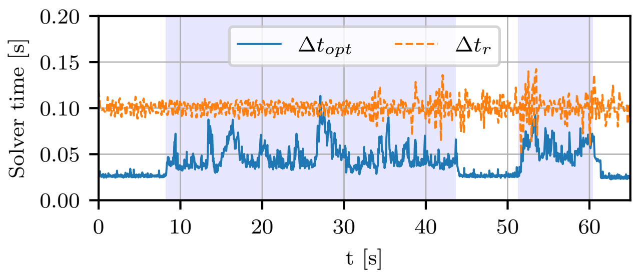

Fig. 9: Optimization times Δt_{opt} and differences between trajectory release times Δt_r. Although optimization times sometimes exceed the targeted maximum time of $0.1\,\mathrm{s}$, the trajectory length of $0.99\,\mathrm{s}$ provides enough margin to provide the impedance controller with new references at all times.

simulates contacts and interaction forces that appear during manipulation tasks. In combination with a reference-tracking impedance controller, our method abolishes the need for analytical models and heuristic-based solutions. Experiments show the ability to generate and track optimal trajectories to accomplish different interaction tasks. They also show that the framework is able to handle intermittent contacts and disturbances effectively by re-planning trajectories accordingly.

Future work will focus on increasing the complexity of the manipulation task as well as on increasing the robustness for real-world scenarios. As for most model-based controllers, this approach relies on an accurate description of the AM and the object. Along this line, we aim to employ onboard vision and tactile sensors to update the state and dynamics estimates of the object within the simulator in real time.

References

[1] D. Mellinger, Q. Lindsey, M. Shomin, and V. Kumar, "Design, modeling, estimation and control for aerial grasping and manipulation," *IEEE International Conference on Intelligent Robots and Systems*, pp. 2668–2673, 2011.

[2] P. E. Pounds, D. R. Bersak, and A. M. Dollar, "Grasping from the air: Hovering capture and load stability," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 2491–2498, 2011.

[3] G. Darivianakis, K. Alexis, M. Burri, and R. Siegwart, "Hybrid predictive control for aerial robotic physical interaction towards inspection operations," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 53–58, 2014.

[4] D. Tzoumanikas, F. Graule, Q. Yan, D. Shah, M. Popovic, and S. Leutenegger, "Aerial Manipulation Using Hybrid Force and Position NMPC Applied to Aerial Writing," 2020.

[5] A. Jimenez-Cano, J. Braga, G. Heredia, and A. Ollero, "Aerial manipulator for structure inspection by contact from the underside," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 2015-Decem. IEEE, sep 2015, pp. 1879–1884.

[6] M. Fumagalli, S. Stramigioli, and R. Carloni, "Mechatronic design of a robotic manipulator for Unmanned Aerial Vehicles," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 2016-Novem. IEEE, oct 2016, pp. 4843–4848.

[7] K. Bodie, M. Brunner, M. Pantic, S. Walser, P. Pfndler, U. Angst, R. Siegwart, and J. Nieto, "An Omnidirectional Aerial Manipulation Platform for Contact-Based Inspection," in *Robotics: Science and Systems XV*. Robotics: Science and Systems Foundation, jun 2019.

[8] R. Rashad, J. B. C. Engelen, and S. Stramigioli, "Energy tank-based wrench/impedance control of a fully-actuated hexarotor: A geometric port-hamiltonian approach," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 6418–6424.

[9] M. Tognon, H. A. Chavez, E. Gasparin, Q. Sable, D. Bicego, A. Mallet, M. Lany, G. Santi, B. Revaz, J. Cortes, and A. Franchi, "A Truly-Redundant Aerial Manipulator System with Application to Push-and-Slide Inspection in Industrial Plants," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1846–1851, 2019.

[10] G. Nava, Q. Sablé, M. Tognon, D. Pucci, and A. Franchi, "Direct Force Feedback Control and Online Multi-Task Optimization for Aerial Manipulators," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 331–338, 2020.

[11] A. Ollero, M. Tognon, A. Suarez, D. Lee, and A. Franchi, "Past, present, and future of aerial robotic manipulators," *IEEE Transactions on Robotics*, vol. 38, no. 1, pp. 626–645, 2022.

[12] H. Tsukagoshi, M. Watanabe, T. Hamada, D. Ashlih, and R. Iizuka, "Aerial manipulator with perching and door-opening capability," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2015-June, no. June. IEEE, may 2015, pp. 4663–4668.

[13] D. Lee, H. Seo, D. Kim, and H. J. Kim, "Aerial Manipulation using Model Predictive Control for Opening a Hinged Door," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, may 2020, pp. 1237–1242.

[14] D. R. Mcarthur, A. B. Chowdhury, and D. J. Cappelleri, "Autonomous Door Opening with the Interacting-BoomCopter Unmanned Aerial Vehicle," *Journal of Mechanisms and Robotics*, vol. 12, no. 2, 2020.

[15] S. Kim, Hoseong Seo, and H. J. Kim, "Operating an unknown drawer using an aerial manipulator," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2015-June, no. June. IEEE, may 2015, pp. 5503–5508.

[16] C. Korpela, M. Orsag, and P. Oh, "Towards valve turning using a dual-arm aerial manipulator," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 3411–3416.

[17] H. Nguyen and K. Alexis, "Forceful Aerial Manipulation Based on an Aerial Robotic Chain: Hybrid Modeling and Control," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3711–3719, apr 2021.

[18] 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 *Proceedings - IEEE International Conference on Robotics and Automation*. IEEE, may 2017, pp. 1714–1721.

[19] I. Abraham, A. Handa, N. Ratliff, K. Lowrey, T. D. Murphey, and D. Fox, "Model-Based Generalization Under Parameter Uncertainty Using Path Integral Control," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2864–2871, apr 2020.

[20] 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.

[21] M. Allenspach, K. Bodie, M. Brunner, L. Rinsoz, Z. Taylor, M. Kamel, R. Siegwart, and J. Nieto, "Design and optimal control of a tiltrotor micro-aerial vehicle for efficient omnidirectional flight," *The International Journal of Robotics Research*, vol. 39, no. 10-11, pp. 1305–1325, sep 2020.

[22] 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, dec 2018.

[23] S. Levine, "Reinforcement learning and control as probabilistic inference: Tutorial and review," 2018. [Online]. Available: https://arxiv.org/abs/1805.00909

[24] T. Lee, M. Leok, and N. H. McClamroch, "Geometric tracking control of a quadrotor UAV on SE(3)," in *49th IEEE Conference on Decision and Control (CDC)*. IEEE, dec 2010, pp. 5420–5425.

[25] J. Hwangbo, J. Lee, and M. Hutter, "Per-contact iteration method for solving contact dynamics," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 895–902, 2018. [Online]. Available: www.raisim.com