Robust Active Visual Perching With Quadrotors on Inclined Surfaces

Jeffrey Mao, *Graduate Student Member, IEEE*, Stephen Nogar, Christopher M. Kroninger, and Giuseppe Loianno, *Member, IEEE*

***Abstract*—Autonomous micro aerial vehicles are deployed for a variety of tasks including surveillance and monitoring. Perching and staring allow the vehicle to monitor targets without flying, saving battery power and increasing the overall mission time without the need to frequently replace batteries. This article addresses the active visual perching (AVP) control problem to autonomously perch on inclined surfaces up to 90°. Our approach generates dynamically feasible trajectories to navigate and perch on a desired target location while taking into account actuator and field-of-view constraints. By replanning in midflight, we take advantage of more accurate target localization increasing the perching maneuver's robustness to target localization or control errors. We leverage the Karush–Kuhn–Tucker (KKT) conditions to identify the compatibility between planning objectives and the visual sensing constraint during the planned maneuver. Furthermore, we experimentally identify the corresponding boundary conditions that maximize the spatio-temporal target visibility during the perching maneuver. The proposed approach works on-board in real time with significant computational constraints relying exclusively on cameras and an inertial measurement unit. Experimental results validate the proposed approach and show a higher success rate as well as increased target interception precision and accuracy compared to a one-shot planning approach, while still retaining aggressive capabilities with flight envelopes that include large displacements from the hover position on inclined surfaces up to 90°, angular speeds up to 750°/s, and accelerations up to 10 m/s^2.**

***Index Terms*—Aerial robotics, perception-aware planning, vision for robotics.**

I. Introduction

MICRO aerial vehicles (MAVs) such as quadrotors have great speed and maneuverability while being able to hover in place. This makes them ideal for exploration and surveillance. However, MAVs are limited by low flight time in the 20–30 min range. By perching on a surface, a quadrotor can extend its mission time and save power while still monitoring one or multiple targets. This motivates the need for autonomous perching solutions in Fig. 1. Inclined flat planar surfaces like walls and rooftops are plentiful especially in urban environments. By focusing on this avenue, we aim to greatly reduce the energy consumption for multiple types of missions. Several challenges complicate the perching maneuver execution. The quadrotor is an underactuated system where both orientation and acceleration of the vehicle are dependent on each other. In addition, during perching maneuvers the vehicle requires large excursions from the hover position. In this article, we propose active visual perching (AVP) designed for robust perching with quadrotors on inclined surfaces. Our previous work [1] addressed the state estimation, control, and planning problems for aggressive perching generating dynamically and physically feasible trajectories on planar surfaces up to 90° solely using on-board sensing and computation perching algorithm with a one-shot planning approach. Conversely, in this work, AVP involves planning new perching trajectories based on the quadrotor's subsequent target localization during the perching maneuver in real time while concurrently maximizing its spatio-temporal visibility. This reduces the noise detection effects or partial visibility of the target by consecutively replanning trajectories in an active fashion as well as reducing the estimation and control errors induced by the aggressive motions. Furthermore, this compensates for the use of low-resolution images and a low-cost inertial measurement unit (IMU) to localize the vehicle and the target. The sum of our replanned trajectories naturally favors trajectories with reduced spatial distance compared to a one-shot planning approach, which further facilitates the target's interception.

This article presents multiple contributions. First, we present the perception, control, and planning approaches for active agile

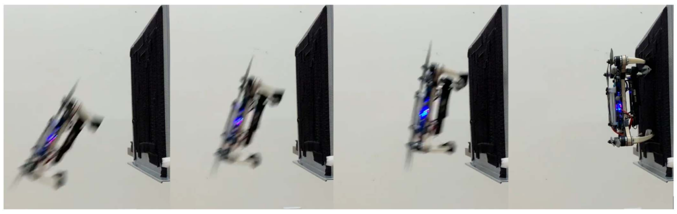


Fig. 1. Aggressive visual perching sequence maneuver for a 90° inclined surface [1].

Manuscript received 31 August 2022; accepted 21 December 2022. This work was supported in part by the DEVCOM ARL under Grant DCIST CRA W911NF-17-2-0181, in part by NSF CAREER Award under Grant 2145277, in part by DARPA YFA under Grant D22AP00156-00, in part by Qualcomm Research, in part by Nokia, and in part by NYU Wireless. This article was recommended for publication by Associate Editor R. Tron and Editor P. Robuffo Giordano upon evaluation of the reviewers' comments. *(Corresponding author: Jeffrey Mao.)*

Jeffrey Mao and Giuseppe Loianno are with the New York University, Tandon School of Engineering, Brooklyn, NY 11201 USA (e-mail: jm7752@nyu.edu; loiannog@nyu.edu).

Stephen Nogar and Christopher M. Kroninger are with the DEVCOM Army Research Laboratory, Adelphi, MD 20783 USA (e-mail: stephen.m.nogar.civ@mail.mil; christopher.m.kroninger.civ@army.mil).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TRO.2023.3238911.

Digital Object Identifier 10.1109/TRO.2023.3238911

autonomous quadrotor perching on planar surfaces up to 90° inclinations. The proposed approach is robust to sensor and control noises as well as to target localization errors by replanning trajectories in midflight. The robot exploits consecutive and more accurate target localization during the perching approach maneuver, thus providing robustness compared to a one-shot planning approach presented in our previous work [1]. Second, our strategy enforces the target visibility within the camera field of view (FoV) to ensure that the landing pad remains in sight. We exploit Karush–Kuhn–Tucker (KKT) conditions to verify the compatibility between the FoV constraint, the perching maneuver objectives, and additional actuator and sensing constraints to ensure the planned trajectory is both dynamically and physically feasible. Third, we experimentally identify the boundary conditions that can further help the FoV constraint to maximize the spatio-temporal target visibility during the maneuver. Finally, our solution is lightweight and run solely on-board limited computational unit without any external sensors or computation at 30 Hz until the target is intercepted. Multiple experimental results demonstrate that the proposed approach is able to achieve more consistent and accurate target interception for planar targets up to 90° inclinations compared to the one-shot planning approach presented in [1].

II. Related Works

Perching on a vertical surface is a challenging problem because quadrotors are nonholonomic and underactuated systems where orientation depends on linear acceleration. As a result, the perching maneuver on steep inclined surfaces such as 90° requires high angular momentum that induces target loss and camera motion blur, making the localization, control, and planning problem extremely challenging. This has led to two main research avenues. The first one focuses on planning and control problems to generate feasible aggressive maneuvers. Multiple works use external sensors to localize the quadrotor and target. The second addresses the mechanical design of proposing novel mechanisms or attachments to facilitate perching on specific structures or surfaces without aggressive maneuvers.

Related to planning, Richter et al. [2] detailed how to employ polynomial splines to formulate a trajectory on the flat outputs or position in the inertial frame and yaw. Further optimizations using different polynomial splines and additional costs have been proposed for solving time optimal trajectories in [3], [4], and [5]; however, these works purely focus on the planning problem for aerial navigation without studying the appropriate planning constraints to resolve the perching problem. More complex planners [6], [7] propose approaches on SE(3) that generate feasible trajectories on both rotation and translation. However, these works require substantially more computation than planning on the flat outputs requiring nonconvex optimization and nonlinear constraints, making them unsuitable for on-board computation on robots with limited computational units. Furthermore, works such as [8], [9], and [10] plan trajectories on the full robot state, but are even more computationally expensive than [6] and [7]. Similarly, for planners considering the full robot state, Paneque et al. [11] tackled the perching problem specifically for power lines as opposed to planar surfaces and considers the full robot state as well as FoV and perching constraints. These approaches do not represent the trajectory based on basis functions instead with discrete points constrained by system dynamics, which are normally solved with a specialized solver nonlinear such as ACADOS [12]. Consequentially, this approach is more computationally expensive taking 400–1500 ms to solve compared to other methods based on flat outputs which generally take less than 100 ms. Our current work is similar to [1], [13], [14] parameterizing the trajectory on the flat outputs and formulating the perching constraint as an acceleration constraint, making a convex optimization with linear constraints. However, we also apply an active vision feedback planning approach to refine our trajectory and achieve more consistent interception with our 90° inclined surfaces.

Some specialized controllers perform similar aggressive maneuvers such as multiple flips in [15] and [16]. These controllers tend to be fairly narrow and are designed to solely execute one maneuver such as multiple flips. They tend to not generalize very well if a different maneuver is required such as perching at a different incline and require motion capture systems. Conversely, Kaufmann et al. [17] and Habas et al. [18] performed aggressive motions such as quadrotor flips using solely on-board sensors, but this is based on a deep learning approach that is still very computationally expensive to run on-board small-scale robots and does not present guarantees. Other works focus on a much simpler target interception while relying purely on on-board camera. Visual servoing approaches [19], [20] have shown controllers capable of landing on targets through purely visual feedback, but these methods are highly dependent on objects' shapes and require the object to continuously be in the FoV to preserve the control stability. Therefore, their maneuvers are not aggressive and do not present large excursions from the hover position, as in our proposed case.

Our past [1] and current work ensures dynamic and physical feasibility in the planned perching trajectory and executes them without external sensors, the proposed approach performs repeated target localization (using an Apriltag [21]) and planning to actively refine the robot's motion during the maneuver to increase the perching accuracy at longer distances and robustness to both control and localization errors. Relevant to this work are other papers that consider the midflight replanning problem [22], [23], [24], [25], [26]. However, the approaches proposed in [22], [23], and [24] are specifically designed for navigation and obstacle avoidance. Penin et al. [24] added target tracking to the problem however that work specializes in nonaggressive trajectories tracking targets moving at most 1 m/s. Falanga et al. [25] only considered perching on targets with 0° inclines. None of the above planners address the challenges of planning maneuvers for aggressive perching. Finally, while Ji et al. [26] do employed a replanning framework targeted for aggressive maneuvers such as perching, they used external sensors to localize the target and did not consider the perception constraints required for maintaining the target in their drone's FoV. In comparison, we propose a more generalizable FoV constraint and a different form of anticipation to reduce errors induced by the replanning computation phase. We also employ a global bound checking algorithm to make

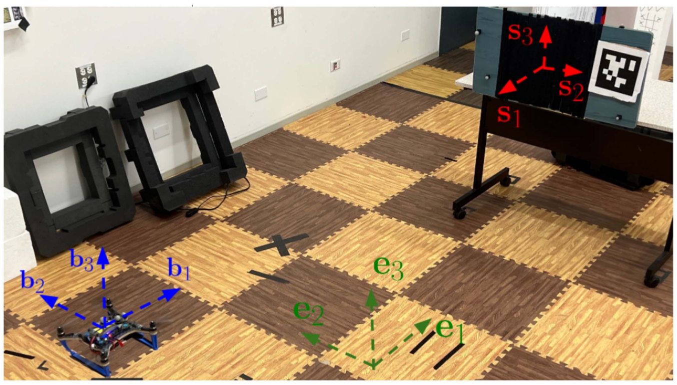


Fig. 2. Visualization of the quadrotor, inertial, and target frame.

the approach computationally efficient for multiple nonlinear constraints such as maximum thrust. Iteratively adding time and performing global bound checks provides in our case a more efficient and accurate bound solver as opposed to the process of linearizing various nonlinear actuator constraints and enforcing them at discrete time intervals across the trajectory.

There are also a variety of perching mechanism designs that aim to simplify the perching problem on a wide range of surfaces such as claws [27], [28], [29] for cylindrical objects, dry adhesives [30], [31], [32], suction gripper mechanisms [33], arms attached to the quadrotor [34], [35], electroadhesive material [36], or other passive mechanisms [37]. Employing customized mechanisms can ease the perching problem on specific surfaces sometimes at the price of a mass increase of the vehicle that can range from 10% in [34] to 40% in [37]. Our proposed planning strategy is general and can complement any customized mechanism by imposing additional boundary constraints to robustify the perching maneuver. In this work, we employ a VELCRO mechanism due to its simplicity and low cost. Finally, fixed-wing solutions for vertical perching have been proposed [38], [39], [40]. Fixed-wing aircraft gain greater flight endurance, but are bulkier and have reduced maneuverability compared to quadrotors. Since quadrotors have reduced flight time compared to fixed-wing solutions, we are further motivated in our research to exploit their maneuverability to land on ideal perching locations and extend their corresponding mission time.

III. Preliminaries

The inertial frame $\mathcal{I}$ is defined by the three axes $[\mathbf{e}_1 \ \ \mathbf{e}_2 \ \ \mathbf{e}_3]$, as shown in Fig. 2. The quadrotor body frame $\mathcal{B}$ is defined by $[\mathbf{b}_1 \ \ \mathbf{b}_2 \ \ \mathbf{b}_3]$. The target frame $\mathcal{S}$ is represented by the landing pad axes $[\mathbf{s}_1 \ \ \mathbf{s}_2 \ \ \mathbf{s}_3]$ and identified in Fig. 2 with an Apriltag [21]. Similarly, the position of the quadrotor's center of mass in the inertial frame is $\boldsymbol{x} = [x \ \ y \ \ z]^\top$, and the target's center as $\boldsymbol{s} = [s_x \ \ s_y \ \ s_z]^\top$. The perching problem requires the vehicles to plan and execute a feasible trajectory in a predefined time frame $t \in [t_0, t_f]$, such that both $\mathcal{B} \equiv \mathcal{S}$ and $\boldsymbol{x} = \boldsymbol{s}$ are reached at $t = t_f$. This is achieved in a loop structure demonstrated in Fig. 3. First, the vehicle visually locates the target and estimates the relative transformation from the $\mathcal{B}$ to $\mathcal{S}$ frames (i.e., relative position $\mathbf{p}_{\mathcal{S}}^{\mathcal{B}} \in \mathbb{R}^3$ and orientation $\mathbf{R}_{\mathcal{S}}^{\mathcal{B}} \in SO(3)$). Second, the relative configuration information is incorporated at the planning and control levels to generate and execute trajectories that are feasible. The proposed setup and approach is shown in Fig. 3. The quadrotor system dynamic model in the inertial frame $\mathcal{I}$ is

$$\dot{\boldsymbol{x}} = \boldsymbol{v}, \dot{\boldsymbol{v}} = \boldsymbol{a}, m\boldsymbol{a} = \mathbf{R}\tau\mathbf{e}_3 - mg\mathbf{e}_3$$
$$\dot{\mathbf{R}} = \mathbf{R}\hat{\mathbf{\Omega}}, \mathbf{J}\dot{\mathbf{\Omega}} + \mathbf{\Omega} \times \mathbf{J}\mathbf{\Omega} = \mathbf{M} \quad (1)$$

where $\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{a} \in \mathbb{R}^3$ are the position, velocity, and acceleration of the quadrotor's center of mass in Cartesian coordinates with respect to the inertial frame $\mathcal{I}$, and $\mathbf{R}$ represents the orientation of the quadrotor with respect to $\mathcal{I}$. $\mathbf{\Omega} \in \mathbb{R}^3$ is the angular velocity of the quadrotor with respect to $\mathcal{B}$, $m \in \mathbb{R}$ denotes the mass of the quadrotor, $\mathbf{J} \in \mathbb{R}^{3\times3}$ represents its inertial matrix with respect to $\mathcal{B}$, $g = 9.81$ m/s^2 is the standard gravitational acceleration, $\mathbf{M} \in \mathbb{R}^3$ is the total moment with respect to $\mathcal{B}$, $\tau \in \mathbb{R}$ represents the total thrust to the quadrotor, and the $\hat{\cdot}$ represents the mapping such that $\hat{\mathbf{a}}\mathbf{b} = \mathbf{a} \times \mathbf{b} \, \forall \mathbf{a}, \mathbf{b} \in \mathbb{R}^3$. To achieve aggressive maneuvers, we apply a nonlinear geometric controller that was leveraged from our previous work [41] to achieve agile flight in indoor environments, where $\mathbf{K}_R, \mathbf{K}_\Omega, \mathbf{K}_x, \mathbf{K}_v \in \mathbb{R}^{3\times3}$ represent the feedback gains for the errors in orientation, angular velocity, position, and velocity, respectively, as positive definite diagonal matrices. Thrust τ and moment $\mathbf{M}$ are the control inputs selected as

$$\begin{aligned}\tau &= (-\mathbf{K}_x\mathbf{e}_x - \mathbf{K}_v\mathbf{e}_v + mg\mathbf{e}_3 + m\ddot{\boldsymbol{x}}) \cdot \mathbf{R}\mathbf{e}_3 = \mathbf{f} \cdot \mathbf{R}\mathbf{e}_3\\ \mathbf{M} &= -\mathbf{K}_R\mathbf{e}_R - \mathbf{K}_\Omega\mathbf{e}_\Omega + \mathbf{\Omega} \times \mathbf{J}\mathbf{\Omega}\\ &\quad - \mathbf{J}\left(\hat{\mathbf{\Omega}}\mathbf{R}^\top\mathbf{R}_C\mathbf{\Omega}_C - \mathbf{R}^\top\mathbf{R}_C\dot{\mathbf{\Omega}}_C\right) \quad (2)\end{aligned}$$

$\mathbf{e}_R, \mathbf{e}_\Omega, \mathbf{e}_x, \mathbf{e}_v \in \mathbb{R}^3$ are the orientation, angular velocity, position, and velocity error vectors, which are detailed in works [42] and [43], and the $*_C$ are the command or desired values obtained from the planning algorithm, as shown in Section V-A. To plan a vehicle's motion in the $\mathcal{I}$ frame with respect to the target, the robot needs to localize itself with respect to the inertial frame $\mathcal{I}$ as well as with respect to the target.

Compared to our previous work [1], the robot continuously exploits the increased accuracy on target localization approaching the target to refine the estimate of the relative configurations from the $\mathcal{B}$ to $\mathcal{S}$ frames. Therefore, as shown in Fig. 3, the system loops back with an updated target and quadrotor localization to replan trajectories in an active fashion with an updated target and quadrotor until the quadrotor has intercepted the target as described in the following.

IV. Active Visual Perching

The goal of active visual perching (AVP) is to be able to plan new trajectories midflight to increase robustness to sensor and control disturbances and noises during aggressive flight by exploiting increased target localization accuracy and precision once the vehicle approaches the target for perching. Compared to our previous work [1], the active visual planning strategy imposes two additional steps in the aforementioned planner. First, since the vehicle is planning while moving and the planning

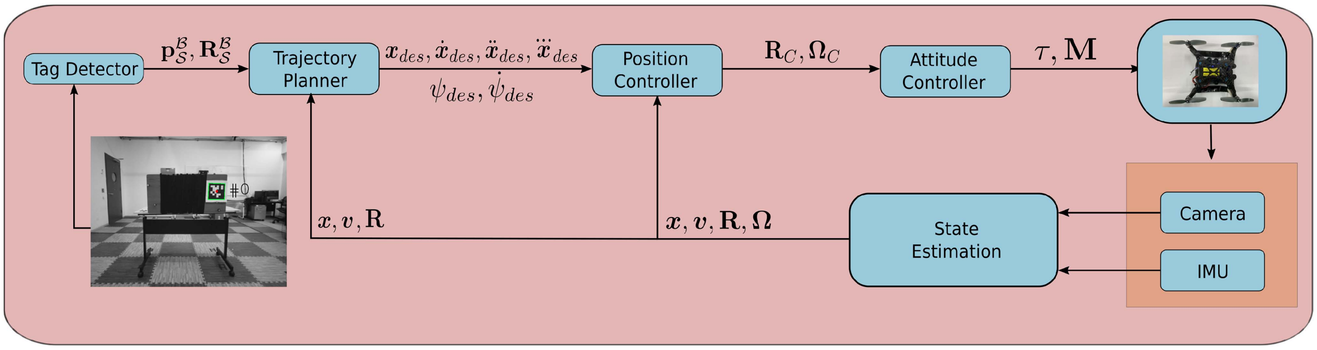

Tag Detector
$\mathbf{p}_{\mathcal{S}}^{\mathcal{B}}, \mathbf{R}_{\mathcal{S}}^{\mathcal{B}}$
Trajectory Planner
$\boldsymbol{x}_{des}, \dot{\boldsymbol{x}}_{des}, \ddot{\boldsymbol{x}}_{des}, \dddot{\boldsymbol{x}}_{des}$
$\psi_{des}, \dot{\psi}_{des}$
Position Controller
$\mathbf{R}_C, \boldsymbol{\Omega}_C$
Attitude Controller
$\tau, \mathbf{M}$
#0
$\boldsymbol{x}, \boldsymbol{v}, \mathbf{R}$
$\boldsymbol{x}, \boldsymbol{v}, \mathbf{R}, \boldsymbol{\Omega}$
State Estimation
Camera
IMU

Fig. 3. System architecture for the perching task.

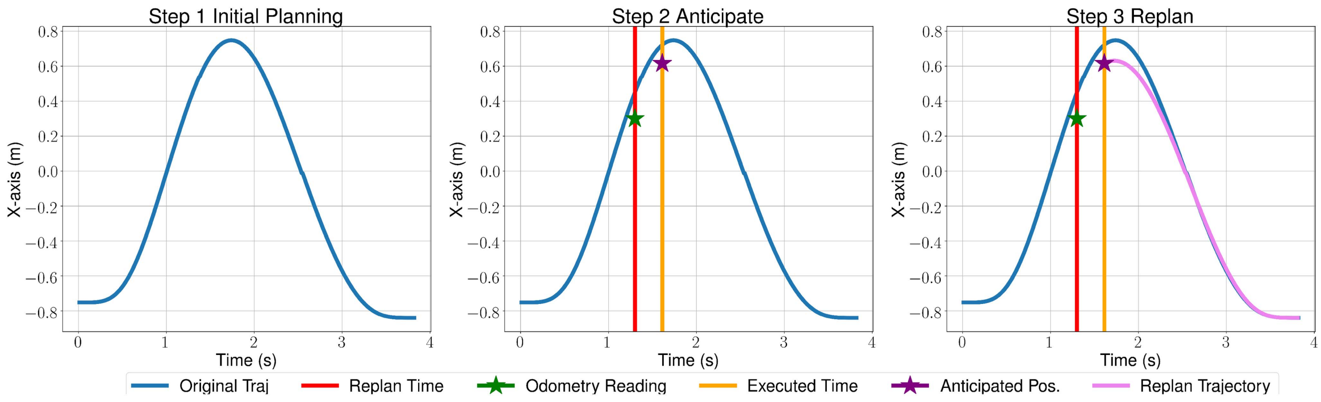


Fig. 4. Replanning steps for one axis. The horizontal axis is time, and the vertical is the X-axis distance. Blue represents the original trajectory. Red is when we start replanning. Orange is when we think the trajectory will take place. Green star is the current odometry. Purple star is the anticipated position. The pink trajectory is the new replanned trajectory.

procedure takes a specific amount of time, we have to anticipate our flat output and corresponding time derivatives in the future when we swap the new trajectory. Second, we need to enforce the target visibility to guarantee and facilitate future replanning. This translates to an FoV constraint along sparsely discretized points of your trajectory. The FoV constraint formulation and implementation as well as the additional constraints that ensure the feasibility of the perching maneuver are detailed in Section V-A. In Fig. 4, we visualize the replanning procedure for one Cartesian axis x. The same procedure is valid for the other axes. Our procedure can be broken into three steps: initial planning, anticipation of the next position, and replanning from the anticipated time and flat output along with corresponding derivatives. First, we have our initial trajectory formulated in step 1 with a set amount of time for perching. Similarly to [1], we model the trajectory on the set of vehicle's flat output space, $\{\boldsymbol{x}, \psi\} = \{x, y, z, \psi\}$, where ψ is the yaw angle of the quadrotor or the rotation around the $\mathbf{b}_3$-axis. Each flat output trajectory is independent and represented using a polynomial spline. We give an example of a single dimension's spline defined as

$$P(t) = \begin{cases} p_1\left(t - t_0\right) & \text{if } t \in [t_0, t_1] \\ p_2\left(t - t_1\right) & \text{if } t \in [t_2, t_1] \\ \vdots & \\ p_f\left(t - t_{f-1}\right) & \text{if } t \in [t_{f-1}, t_f] \end{cases}$$

$$p_i(t) = \sum_{n=0}^{N} c_{ni} t^n,\ i = 1, \ldots, f \tag{3}$$

where p_i represents the ith polynomial spline making up the full trajectory of P, $c_{ni} \in \mathbb{R}$ is the nth coefficient of p_i, and N represents the polynomial order chosen for each spline. For our polynomial spline, we generate a reasonable guess at the required time needed for each spline. This formulation allows us to declare arbitrary constraints on the quadrotor's flat outputs and corresponding derivatives as linear constraints for any arbitrary time or time range, as defined in [1]. We minimize the squared norm of the jth derivative defined as

$$\int_{t_0}^{t_f} \left\| \frac{d^j P(t)}{dt^j} \right\|^2 dt. \tag{4}$$

As in our previous work [1], we formalize this problem as a quadratic programming (QP) problem on the polynomial coefficients $\mathbf{c}$ defined as

$$\begin{aligned} \min_{\mathbf{c}} \quad & \mathbf{c}^T \mathbf{Q} \mathbf{c} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{c} = \mathbf{b} \\ & \mathbf{y} \leq \mathbf{G}\mathbf{c} \leq \mathbf{z} \end{aligned} \tag{5}$$

where the matrix $\mathbf{Q}$ is derived from the cost function in (4), and the matrices $\mathbf{A}$ and $\mathbf{b}$ are derived from the equality constraint declared on flat outputs and corresponding derivatives for some arbitrary times, and $\mathbf{y}, \mathbf{G},$ and $\mathbf{z}$ on inequality constraints, as defined in [1]. Next, in step 2, we detect our current position, which is represented by the green star and t_{rp}, replan starting time represented by the red vertical line. Then, to take into account the overall replanning time duration on the vehicle's motion, we incorporate the quadrotor's flat output and corresponding derivatives with the spatio-temporal information by inferring the displacement of the quadrotor once the replanning ends as

$$\begin{aligned}
\boldsymbol{x}_0 &= \boldsymbol{x}_n + \mathbf{P}(t_{\text{ex}}) - \mathbf{P}(t_{\text{rp}}) \\
\frac{d\boldsymbol{x}_0}{dt} &= \frac{d\boldsymbol{x}_n}{dt} + \frac{d\mathbf{P}(t_{\text{ex}})}{dt} - \frac{d\mathbf{P}(t_{\text{rp}})}{dt} \\
&\vdots \\
\frac{d^j\boldsymbol{x}_0}{dt^j} &= \frac{d^j\boldsymbol{x}_n}{dt^j} + \frac{d^j\mathbf{P}(t_{\text{ex}})}{dt^j} - \frac{d^j\mathbf{P}(t_{\text{rp}})}{dt^j}.
\end{aligned} \tag{6}$$

It should be noted that we use $\mathbf{P}$ to refer to the trajectory on the full flat output states as opposed to P for one flat output. The yaw has not been included for simplicity of notation. The quadrotor's position is represented by the purple star whereas the orange line represents the anticipated time of execution t_{ex}. The anticipation $\boldsymbol{x}_0$ is the spatial displacement that would occur in the originally planned trajectory and adding it to the odometry position $\boldsymbol{x}_n$. This is repeated for velocity, acceleration, and up to the fourth-order derivative, as shown in (6). In our final step, we plan a new perching trajectory with the anticipated flat output including all the relevant constraints detailed in Section V. The process is repeated until target interception.

V. AVP Constraints

In this section, we detail the specific constraints in (5) of the proposed AVP to intercept the target at the endpoint and maintain line of sight to the target with the robot front camera. We also derive the additional constraints required to intercept the target and show how the choice of the target's impact velocity influences the spatio-temporal target visibility during the maneuver. The above constraints represent inequality and equality constraints in the proposed optimization problem in (5). Finally, we detail the procedure to guarantee that the planning constraints are met.

A. FoV Constraint

On approach to the target, target localization improves as long as the target is in the FoV, as quantified in Fig. 9. This motivates an FoV constraint to be imposed onto the trajectory planning to maintain line of sight, so that the proposed AVP algorithm can exploit better localization. Our goal with this FoV constraint is to derive a linear constraint that is compatible with the optimization problem described in (5) and ensures that the next time the quadrotor needs the target localization for planning it is in the correct position and orientation to view the target. Inspired by [44], we formulate our FoV constraint as a cone projected from the center of the drone's front camera. We refer to Fig. 5 for a visualization. The relevant quantities for this constraint will be represented in the frame $\mathcal{I}$.

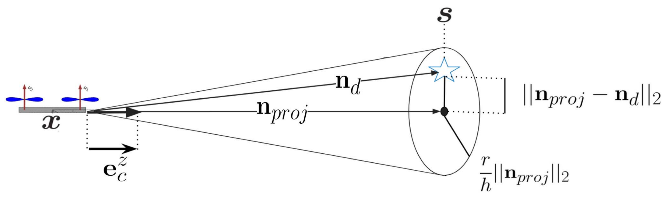


Fig. 5. FoV constraint representation. $\boldsymbol{x}$ is the quadrotor's position. $\boldsymbol{s}$ and the star represents the target's position.

First, $\boldsymbol{x}$ and $\boldsymbol{s}$ represent the drone and perching target's location. $\mathbf{n}_d = \boldsymbol{s} - \boldsymbol{x}$ represents the vector from the drone to the target. $\mathbf{e}_c^Z$ represents the unit vector projected from the camera's center. $\mathbf{n}_{\text{proj}}$ is the projection of $\mathbf{n}_d$ on to $\mathbf{e}_c^Z$. Finally, the ratio $\frac{r}{h}$ represents the ratio between the FoV cone's radius and height. To solve for $\mathbf{n}_{\text{proj}}$, we take the projection of $\mathbf{n}_d$ on to $\mathbf{e}_c^Z$, such that the $\mathbf{e}_c^Z$ direction is maintained. This vector is $\mathbf{n}_{\text{proj}} = \left(\mathbf{n}_d^T \mathbf{e}_c^Z\right) \mathbf{e}_c^Z$. We can then define the FoV as a circle with a radius $\frac{r}{h}||\mathbf{n}_{\text{proj}}||_2$ and center at $\mathbf{n}_{\text{proj}}$. Finally, we wish to enforce the target to be inside the circular FoV. This finalizes our constraint as

$$||\mathbf{n}_d - \mathbf{n}_{\text{proj}}||_2 \leq \frac{r}{h}||\mathbf{n}_{\text{proj}}||_2. \tag{7}$$

To apply our nonlinear FoV constraint (7), to our optimization problem described in (5), we use a Taylor series expansion to linearize the inequality. Doing this linearization represents an advantage from a computational perspective compared to employing nonlinear methods, as we show in Section VI.

The constraint is a function of position, yaw, and acceleration. In other words, we can formulate (7) as $f(\boldsymbol{x}, \psi, \mathbf{a}) \leq g(\boldsymbol{x}, \psi, \mathbf{a})$. The Taylor series expansion of this constraint linearized around $(\boldsymbol{x}_0, \psi_0, \mathbf{a}_0)$ becomes

$$\begin{aligned}
&\left(\nabla_f\left(\boldsymbol{x}_0, \psi_0, \mathbf{a}_0\right) - \nabla_g\left(\boldsymbol{x}_0, \psi_0, \boldsymbol{a}_0\right)\right)\begin{bmatrix}\boldsymbol{x}\\ \psi\\ \boldsymbol{a}\end{bmatrix} \\
&\leq g(\boldsymbol{x}_0, \psi_0, \boldsymbol{a}_0) - f(\boldsymbol{x}_0, \psi_0, \boldsymbol{a}_0) \\
&\quad + \left(\nabla_f\left(\boldsymbol{x}_0, \psi_0, \boldsymbol{a}_0\right) - \nabla_g\left(\boldsymbol{x}_0, \psi_0, \boldsymbol{a}_0\right)\right)\begin{bmatrix}\boldsymbol{x}_0\\ \psi_0\\ \boldsymbol{a}_0\end{bmatrix}
\end{aligned} \tag{8}$$

where ∇_f and ∇_g represent the gradients of $f(x)$ and $g(x)$, respectively, with respect to $(\boldsymbol{x}, \psi, \boldsymbol{a})$. For our constraint, we select our linearization point $(\boldsymbol{x}_0, \psi_0, \boldsymbol{a}_0)$ as the next values from the next point we replan from to ensure that the target is still in view. These values $(\boldsymbol{x}_0, \psi_0, \boldsymbol{a}_0)$ are calculated using (6). In addition, we impose an additional constraint on our position, yaw, and acceleration during replanning such that

$$\begin{aligned}
||\boldsymbol{x} - \boldsymbol{x}_0||_2 &\leq 0.05 \\
||\boldsymbol{a} - \boldsymbol{a}_0||_2 &\leq 0.2 \\
||\psi - \psi_0||_2 &\leq 0.1
\end{aligned} \tag{9}$$

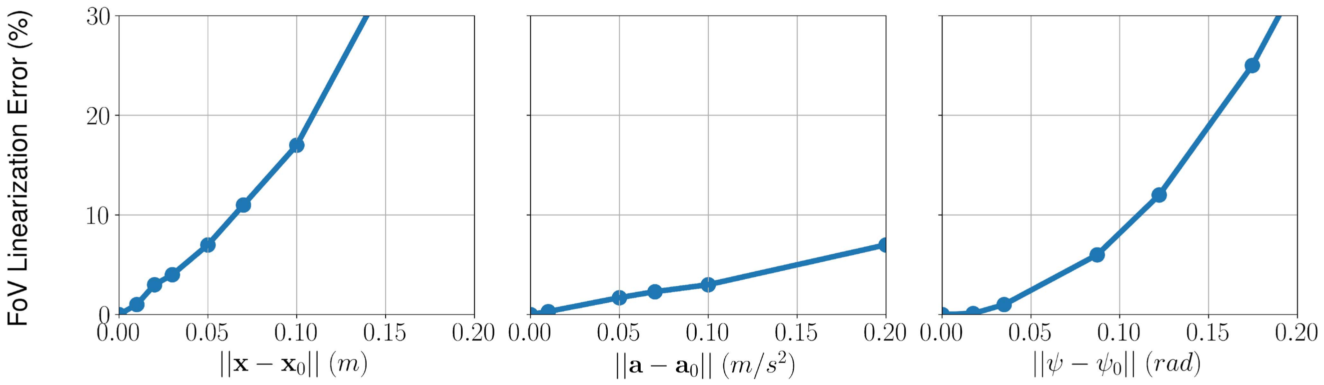


Fig. 6. Linearization error (8) of the FoV as a disturbance from the linearization zone. All plots share the same *y*-axis scale.

to ensure bounded linearization error, as shown in Fig. 6, that details the Taylor series expansion error as a function of $||\boldsymbol{x} - \boldsymbol{x}_0||_2$, $||\psi - \psi_0||_2$, and $||\boldsymbol{a} - \boldsymbol{a}_0||_2$. From this result, the Taylor series represents a good approximation of the original nonlinear constraint as long as (9) is satisfied. We repeat the linear FoV constraint (8) for n_p points spaced equally along the remaining trajectory. Each constraint (7) is linearized around the predicted value of the previously planned trajectory. This is similar to other nonlinear optimizers such as sequential quadratic programming (SQP), where the previous iteration is used as linearization points for the next nonlinear optimization iteration.

B. Perching Physical and Perception Constraints

Inspired by our previous work [14], we exploit the nonholonomic properties of the quadrotor and relate $\ddot{\boldsymbol{x}}$ and its orientation $\mathbf{R}$. From the translational dynamics (1), the thrust vector is a function of the quadrotor's acceleration

$$\tau = m||\ddot{\boldsymbol{x}} + g\mathbf{e}_3||. \tag{10}$$

Because thrust can only be actuated on the plane normal to the propeller blades, the generated force is solely on the $\mathbf{b}_3$-axis of the body frame of a quadrotor. From here, we can derive that $\mathbf{b}_3$ should be

$$\mathbf{b}_3 = \frac{\ddot{\boldsymbol{x}} + g\mathbf{e}_3}{||\ddot{\boldsymbol{x}} + g\mathbf{e}_3||}. \tag{11}$$

To achieve successful perching, the quadrotor's body frame $\mathcal{B}$ must align with the target's frame $\mathbf{R}_{\mathcal{S}}$, meaning $\mathbf{b}_3 = \mathbf{s}_3$ at the trajectory's end time t_f. The $\mathbf{s}_3$ direction on the inclined surfaces is extracted from the last column of the matrix $\mathbf{R}_{\mathcal{S}}$. This orientation $\mathbf{R}_{\mathcal{S}}$ is combining the target localization with respect to the body frame and the on-board localization of the body frame with respect to the inertial frame. Using (11), we can declare this constraint as an acceleration as

$$\ddot{\boldsymbol{x}}(t_f) = \alpha\mathbf{s}_3 - g\mathbf{e}_3 \tag{12}$$

where $\alpha = ||\ddot{\boldsymbol{x}}(t_f) + g\mathbf{e}_3|| \in \mathbb{R}$ corresponds to the thrust's norm. The planned $\ddot{\boldsymbol{x}}$ is then set as a constraint for our optimization defined in (5). We declare our quadrotor with the *Z*(ψ)-*Y*(θ)-*X*(ϕ) convention to construct the rotation from flat outputs as derived below. To control the other two axis of the vehicle, $\mathbf{b}_1$, $\mathbf{b}_2$, we relate our desired $\mathbf{R}_C$ in (2) to a desired yaw angle ψ_{des} as

$$\mathbf{R}_C = \begin{bmatrix}\mathbf{b}_{1,C} & \mathbf{b}_{2,C} & \mathbf{b}_{3,C},\end{bmatrix}$$
$$\mathbf{b}_{1,C} = \frac{\mathbf{b}_{2,\text{des}} \times \mathbf{b}_3}{||\mathbf{b}_{2,\text{des}} \times \mathbf{b}_3||},\quad \mathbf{b}_{2,C} = \mathbf{b}_3 \times \mathbf{b}_1$$
$$\mathbf{b}_{2,\text{des}} = [-\sin\psi_{\text{des}},\ \ \cos\psi_{\text{des}},\ \ 0]^\top,\ \mathbf{b}_{3,C} = \frac{\mathbf{f}}{||\mathbf{f}||}.$$

For perching on a planar surface, this ψ_{des} can be any angle that does not induce a singularity in (13). In our construction of $\mathbf{R}_C$, a singularity occurs when the rotation around the roll or $\mathbf{b}_1$-axis is at $\pm 90^\circ$. Generally, we select it such that $\mathbf{b}_{2,\text{des}}$ is parallel to $\mathbf{s}_2$, which can be know from $\mathbf{R}_{\mathcal{S}}$. The commanded angular rate $\hat{\boldsymbol{\Omega}}_C$ in (2) is then

$$\hat{\boldsymbol{\Omega}}_C = \mathbf{R}_C^\top\dot{\mathbf{R}}_C. \tag{13}$$

For practical purposes, most of the rotation should completed before target interception to stop the front of the quadrotor from bouncing off the target and failing to adhere to the target or stopping the quadrotor's vital components such as the propellers from colliding with the landing pad. This constraint is expressed by enforcing an additional acceleration range by a given q tolerance in proximity of the target. To apply the inequality constraint to our optimization, we discretize the equation as

$$(\alpha\mathbf{s}_3 - g\mathbf{e}_3) \leq \ddot{\boldsymbol{x}}(t) \leq (1+q)\,(\alpha\mathbf{s}_3 - g\mathbf{e}_3)$$
$$\forall t \in \{t_f - t_k + j * dt\},\quad j \in \mathbb{Z}\&0 \leq j < \frac{t_k}{dt} \tag{14}$$

where dt is the sampling time of our trajectory planner and t_k is the time prior to the impact that is user defined. Physically this inequality means that before impact the rotation of the quadrotor is close to the final orientation. When perching at a very steep angle, such as 90°, the endpoint velocity $\mathbf{v}$ can have a large effect on the perching path. This can either aid or greatly hinder the ability to see the target depending on if the quadrotor is ascending or descending to intercept. To characterize the effects of endpoint velocity on the FoV, we analyze how various endpoint velocities change the path and how well this trajectory maintains the target

TABLE I
PERCENT OF THE TIME–PERCENT OF DISTANCE RATIO THAT TARGET IN ITS FoV AS A FUNCTION OF $\mathbf{v}_{S1}$ AND $\mathbf{v}_{S3}$ DURING THE PERCHING MANEUVER

	$\mathbf{v}_{S1} = 0$	$\mathbf{v}_{S1} = 1$	$\mathbf{v}_{S1} = 2$
$\mathbf{v}_{S3} = \text{Null}$	65.5%/70.3%	78.1%/63.9%	85.1%/65.1%
$\mathbf{v}_{S3} = -3$	68.9%/72.5%	79.8%/62.5%	85.6%/65.8%
$\mathbf{v}_{S3} = -2$	**77.5%/83.7%**	85.1%/75.1%	53.4%/60.7%
$\mathbf{v}_{S3} = -1$	29.8%/33.4%	41.3%/48.3%	43.6%/45%
$\mathbf{v}_{S3} = 0$	22.4%/10.4%	33.9%/27.9%	38.5%/34.2%
$\mathbf{v}_{S3} = 1$	17.8%/8.04%	27.0%/20.4%	25.1%/29.7%

Notes: Velocities are in m/s.

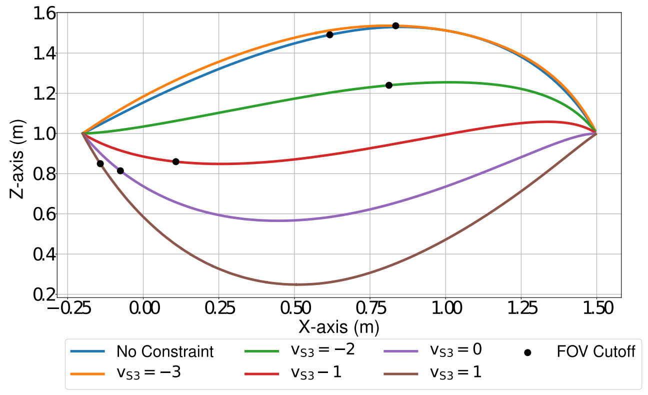


Fig. 7. Visualization of each path for various $\mathbf{v}_{S3}$ during 90° perching. $\mathbf{v}_{S1} = 1\text{m/s}$. The black dot is where the quadrotor loses sight of the target. Target's position is the endpoint of the trajectory. Velocities are in m/s.

in the FoV for a trajectory. For our experiments, we consider two main factors: the percent of the time that the trajectory respects the FoV constraint and the ratio between the arc lengths where the FoV is respected over the full path's length. The qualitative and quantitative results of this study are demonstrated in Table I and Fig. 7. Here, we consider mainly the following two components: $\mathbf{v}_{S1} = \mathbf{v} \cdot \mathbf{S}_1$ and $\mathbf{v}_{S3} = \mathbf{v} \cdot \mathbf{S}_3$, where $\mathbf{v}_{S1}$ is the velocity normal to the landing pad and $\mathbf{v}_{S3}$ is the velocity parallel to the landing pad. For the experiments in Table I, we set the same time for each of the trajectories. From Fig. 7, we notice the effect of varying $\mathbf{v}_{S3}$. As $\mathbf{v}_{S3}$ approaches negative infinity, the maximum height is higher, and for the reverse, as $\mathbf{v}_{S3}$ approaches, infinity, the minimum height of the path dramatically decreases. $\mathbf{v}_{S1}$ does not influence the maximum height or dip of perching but instead influences where the maximum or minimum dip happens. The faster the normal velocity is, the closer to the start the hump will occur. In a generic use case, $\mathbf{v}_{S1}$ will also need to be tuned to meet the perching mechanism. However, for our mechanism, any nonzero $\mathbf{v}_{S1}$ is sufficient to achieve perching. We employ the combination of $\mathbf{v}_{S1}$ and $\mathbf{v}_{S3}$ based on our simulation results below to optimize the percent of the trajectory that respects the FoV constraint. Each of the above paths is generated assuming that the FoV inequality condition is active for linearization of (8) considering $n_p = 8$ points in the future trajectory outlook.

C. Perching Actuator Constraints

Once the optimization problem in (5) is solved, we must ensure that the actuator constraint is met

$$\tau_{\min}^2 \leq \|m\ddot{\mathbf{x}} + mg\mathbf{e}_3\|_2^2 \leq \tau_{\max}^2 \tag{15}$$

Algorithm 1: GLOBAL BOUND CHECKING (GBC) [1].

```
Returns true if $H(t) < b \,\forall\, t \in [t_0, t_f]$. $H(t)$ is any
 polynomial. $b \in \mathbb{R}$ is an upper bound
1: Let $F(t) = H(t) - b$;
2: if $F(t_0) > 0$ or $F(t_f) > 0$ then
3:   return FALSE
4: end if
5: if STURM$(F(t), t_0, t_f) > 0$ then
6:   return FALSE
7: end if
8: return TRUE
```

where $\tau_{\min}$ and $\tau_{\max}$ are the minimum and maximum thrust, respectively. This condition unlike the other previous conditions must be true for the entire trajectory flight time rather than a specific time frame. As a result, it is very difficult to incorporate the QP optimization problem described in (5). Direct incorporation in the optimization problem would increase the solving time. Using our previous work [1], which is built on Sturm's theorem from [45], we perform efficient bounds check postsolving using the global bound checking summarized in Algorithm 1. Details about the proof of this algorithm are provided in the Appendix. This algorithm returns true if given some polynomial $H(t)$ and some constant b, then $H(t) < b$ for all $t \in [t_0, t_f]$. Otherwise, it returns false. Since we use polynomial splines to characterize our trajectory, the thrust is a polynomial. We apply the global bound checker after solving the QP optimization to check if (15) is respected. Should this constraint not be respected, we iteratively add time to each segment of the polynomial till the constraint is respected. This procedure takes place primarily after solving the initial planning. This is because the combined result of all the newly planned trajectories tends to follow a shorter trajectory and move at a lower average speed than a one-shot approach as we observe in the experimental section with Fig. 11.

Additional constraints can be considered such as thrust rate, angular velocity, angular acceleration, moments, and moment rates to Algorithm 1 of our work. A derivation of these quantities can be found in Appendix B. Therefore, Sturm's theorem from [45] can be applied to perform efficient bounds check. If we find the trajectory is infeasible, then we can add time to decrease the maximum of the above constraints similar to thrust and find a feasible trajectory.

D. Feasibility Conditions

Finally, there exists the possibility that the visibility constraint will not be compatible with the perching constraints. In order to verify the existence of this case, we employ the KKT conditions. The KKT conditions for our QP, defined by (5) with constraints specified in (8), (12), and (14), determine the solution if one exists. To formulate our KKT condition, we only consider inequality constraint of the form $\mathbf{y} \leq \mathbf{Gc}$. The general inequality constraint $\mathbf{Gc} \leq \mathbf{z}$ is equivalent to the linear constraint $-\mathbf{z} \leq -\mathbf{Gc}$, so the above formulation does not restrict the set of solvable problems. Solving the KKT conditions involves creating two sets of Lagrange multipliers $\boldsymbol{\lambda}_1 \in \mathbb{R}^J$ and $\boldsymbol{\lambda}_2 \in \mathbb{R}^K$, and a group of slack variables $\mathbf{s} \in \mathbb{R}^K$, where

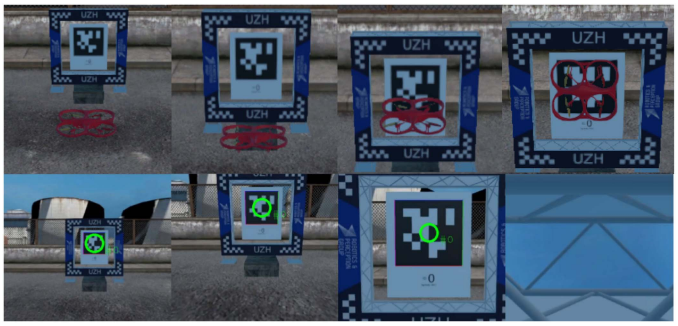


Fig. 8. Virtual perching in simulation. The top four images are the view of the quadrotor in the simulated environment. The bottom four images display the view from the camera with tag detection. The green circle denotes the target detection.

J and K are the number of equality and inequality constraints, respectively. Based on these new variables, we can formulate the KKT conditions as

$$\begin{aligned} &\mathbf{Q}\mathbf{c} - \mathbf{A}\boldsymbol{\lambda}_1 - \mathbf{G}\boldsymbol{\lambda}_2 = 0, && \boldsymbol{\lambda}_1^\top(\mathbf{y} - \mathbf{G}\mathbf{c}) = 0 \\ &\mathbf{A}\mathbf{c} - \mathbf{b} = 0, && \mathbf{G}\mathbf{c} - \mathbf{y} - \mathbf{s} = 0 \\ &\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \mathbf{s} \geq 0. \end{aligned} \tag{16}$$

We solve these conditions using the primal-dual interior-point method. Should an incompatibility between the FoV constraints and other objectives or constraints occur, we relax the problem by removing the FoV constraint and plan the trajectory by only incorporating the perching constraint.

VI. Results

In this section, we demonstrate in simulation that our quadrotor's visibility constraints provide real improvements to the visibility of the target. Next, we show the improvement in target interception gained from using multiple detection on approach as opposed to one-shot planning. Finally, we demonstrate that we can successfully perch on any target up to 90° with only on-board sensing and computation.

A. Setup

We deploy our approach both in simulation and real-world settings. For simulation, we consider a photorealistic simulator based on a variation of Flightmare [46]. This simulator is capable of rendering both the quadrotor, environment, and front camera view, as demonstrated in Fig. 8. In our setup, Flightmare Unity only handles graphic rendering of both on-board cameras and the environment. This allows us to fully test and validate our framework without any setup variation in simulation. The simulated on-board camera images are fed back into our planning software to execute our AVP perching maneuver. The physics simulation is built on our custom quadrotor simulator, which takes in the output of our position controller and simulates the motion of the quadrotor (a simple flag sends the commands directly to a real platform). The landing pad area is localized both in simulation and real-world experiments using a single large Apriltag [21]. Our landing pad is identified using the quadrotor's front camera that detects a single large Apriltag with a known offset from the main pad. In the real world, the pad is mounted on an adjustable desk that controls the height. The adhesive is attached to an adjustable stand that allows variation in the surface angle. A sample perching sequence is demonstrated in simulation in Fig. 8. The real-world experiments are conducted at the Agile Robotics and Perception Lab (ARPL) at New York University in a flying arena of dimensions $10 \times 6 \times 4\,\mathrm{m}^3$. As shown in Fig. 2, the platform is a quadrotor running with a Qualcomm Snapdragon$^{\mathrm{TM}}$ board and four brushless motors. The Qualcomm Snapdragon$^{\mathrm{TM}}$ consists of a Qualcomm Hexagon$^{\mathrm{TM}}$ DSP, Wi-Fi, Bluetooth, GPS, quad-core processor, an IMU, and two cameras: a downward facing 160° FoV and a front-facing 70° FOV. For perching, we employ VELCRO material due to its simplicity and low cost in the ventral part of the vehicle. The software framework is developed in ROS[1] Indigo on a Linux kernel. As specified in Section III, localization with respect to the inertial frame $\mathcal{I}$ is performed based on the on-board visual inertial odometry (VIO) system using the downward facing camera [14] upsampled with the IMU to 300 Hz. The front-facing camera is used to localize the target at a rate of 10 Hz. A Vicon[2] motion capture system is used to record the ground truth data for comparison at 100 Hz.

In both simulation and real-world settings, we employ the same parameters and settings for our perching maneuver. We perform AVP at a rate of 30 Hz for replanning. In our setup, perching constraints are set as $\mathbf{v}_{S3} = -2\,\mathrm{m/s}$ and $\mathbf{v}_{S1} = 0.3\,\mathrm{m/s}$. The slight increase of $\mathbf{v}_{S1} = 0.3\,\mathrm{m/s}$ from the optimal $\mathbf{v}_{S1} = 0\,\mathrm{m/s}$ is because the VELCRO requires some forward momentum to attach itself. We empirically identified that $0.3\,\mathrm{m/s}$ is a good forward momentum to attach to the landing pad. We selected a tolerance $q = 0.1$, sampling time $dt = 0.01\,\mathrm{s}$, time before impact $t_k = 0.15\,\mathrm{s}$, and $\alpha = 4.0\,\mathrm{m/s}^2$ as the hyperparameters of (12) and (14). We choose to minimize the $j = 4$, snap norm, for the cost function in (4). Regarding the FoV constraint in (9), we chose eight points to preserve a good tradeoff between computation and ensuring the FoV constraint is enforced for most of the trajectory.

We solve the optimization problem in (5) using the object-oriented quadratic programming (OOQP) library [47]. Our optimizer concurrently solves the KKT conditions described in (16) and raises a flag simultaneously after solving if a solution exists or not. In the case of no solution being found due to the incompatibility of the FoV constraint with other objectives or constraints, we drop the visibility constraints and resolve the problem considering only the perching constraints to guarantee the physical feasibility of the maneuver. The planning process takes 29 ms on-board our platform equipped with an Arm processing unit and 28 ms on a novel NVIDIA Xavier NX. We also evaluate the computational time when leveraging recent nonlinear solvers like ACADOS [12] with SQP [48]. Similar to [26], the proposed formulation using ACADOS solves an optimization problem considering a series of setpoints with a time difference of 1 ms between each other as opposed to finding

[1][Online]. Available: www.ros.org

[2][Online]. Available: www.vicon.com

TABLE II
COMPUTATION TIME BETWEEN DIFFERENT SOLVERS RUNNING ON VARIOUS ARCHITECTURES

Solver	Snapdragon (ms)	Xavier NX (ms)	Intel i5 (ms)
OOQP	29	28	11
ACADOS	N/A	40	13

the polynomial coefficients as in our original approach. This formulation as a discrete time nonlinear optimization represents the way ACADOS treats these problems and is consistent with other works [26] in the field, which have proposed a similar approach. While ACADOS can be used to solve for polynomial coefficients, it is unlikely to be an interesting comparison as ACADOS uses an optimizer called SQP, which requires an inner QP optimizer where one of the given options is OOQP. ACADOS performs multiple iterations until the cost converges to below $5e{-7}$ threshold. The process takes 40 ms on an NVIDIA Xavier NX. It is not possible to test ACADOS on-board our platform because of multiple library/hardware incompatibilities related to the specific arm architecture available on our quadrotor. We also execute the tests on a laptop equipped with an Intel i5-9300H CPU. We see both solvers nonlinear and linear give similar results 11 and 13 ms for OOQP and ACADOS, respectively. The results are summarized in Table II. The computational time between linear QP and nonlinear when solving the problem off-board is quite similar, whereas on-board computationally constrained platforms, the time difference is larger. This is due to multiple iterations that the SQP performs, which take a larger amount of time on-board computationally constrained platforms.

B. Target Localization

As the distance between camera and target reduces, the vision-based Apriltag target localization improves. In this experiment, we demonstrate this by flying a quadrotor level to the target and backing away horizontally to measure the localization error from the Apriltag and compare it to the ground truth pose values obtained using our Vicon motion capture system. This procedure is repeated at multiple inclines where the angle refers to the relative angle between the floor and the target. The results are recorded in Fig. 9. As seen in Fig. 9, increasing the distance causes the target localization error to exponentially increase. This motivates the need for replanning in our system. Leveraging-reduced localization error during the perching maneuver will improve perching consistency. The decrease of accuracy in target localization as the quadrotor moves further from the target is also successfully modeled in our simulator where we notice a similar accuracy drop off.

C. AVP Evaluation

1) Simulation: In our simulations, we perform a sequence of studies to quantify the impact of various constraints and parameters. First, we validate that the constraints selected for planning a perching maneuver improve the target visibility in terms of both space and time. Second, we evaluate the accuracy of the anticipation step in the AVP. Finally, we demonstrate replanning

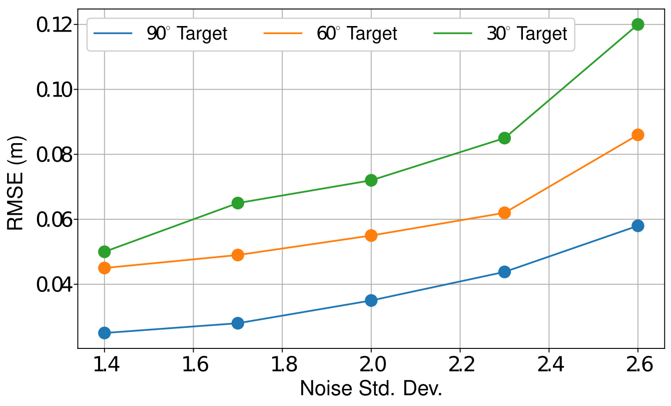


Fig. 9. AprilTag localization error as a function of distance from the target at various inclines.

TABLE III
ABLATION STUDY OF THE ROLE OF THE FOV AND BOUNDARY CONDITIONS ON THE TARGET VISIBILITY

Constraint		1.5 m (%)		3.5 m (%)		7.5 m (%)	
FoV	$\mathbf{v}_{s1s3}$	Time	Space	Time	Space	Time	Space
✗	✗	65	44	71	52	80	74
✓	✗	**72**	45	82	53	**89**	74
✗	✓	68	47	80	76	86	83
✓	✓	70	**53**	**82**	**76**	87	**88**

Notes: The FoV constraint is represented by (8), whereas $\mathbf{v}_{s1s3}$ refers to if both $\mathbf{v}_{s1}$ and $\mathbf{v}_{s3}$ are set or left floating. Time and space refer to the corresponding temporal or spatial FoV percentage during the perching maneuver. The distances such 1.5, 3.5, and 7.5 m refer to the distance the quadrotor to the landing pad along the $\mathbf{s}_1$-axis.

improves the interception with the target by comparing one shot with AVP in simulation. The advantage of performing these experiments in simulation is that more variables are controlled and only the removed or altered component in the study causes performance differences between experiments.

First, we must verify all relevant constraints: (8), $\mathbf{v}_{s1}$, and $\mathbf{v}_{s3}$ contribute to improving the visibility of the target. We first disable the constraints expressed by (8) as well as $\mathbf{v}_{s1}$ and $\mathbf{v}_{s3}$ in the optimization. Our system then executes a perching trajectory and records how long in terms of both space and time the target is recognized with our tag detector. Spatial visibility refers to the percentage of the path traveled that the target is visible. Temporal visibility refers to the percentage of time in performing the maneuver the target is visible. We repeated this experiment with every combination of the constraints (8), $\mathbf{v}_{s1}$ and $\mathbf{v}_{s3}$ at a distance of 1.5, 3.5, and 7.5 m along the $\mathbf{s}_1$-axis away from the target. In this case, the vehicle was aligned with the target on the y-axis. The results are presented in Table III . The results show that as expected the FoV constraint (8) is useful to enforce target spatio-temporal visibility. By carefully selecting $\mathbf{v}_{s1}$ and $\mathbf{v}_{s3}$, we obtain a better spatial visibility of the target compared to the case when no constraints are active. Combining these two constraints, we can achieve a stronger spatio-temporal target visibility when perching without substantially comprising on either. Additional experiments are performed to verify that the visibility both spatially and temporally remains consistent for different starting positions. We changed the relative position

TABLE IV
TEMPORAL-SPATIAL VISIBILITY OF SIMULATED PERCHING

Axis		1.5 m (%)		3.5 m (%)		7.5 m (%)	
$\mathbf{s}_2$ (m)	$\mathbf{s}_3$ (m)	Time	Space	Time	Space	Time	Space
0	0	70	53	82	71	87	88
0.5	0	70	54	81	76	85	83
2	0	68	50	82	75	86	85
4	0	-	-	85	75	87	84
-0.5	0	70	54	79	76	84	83
-2	0	69	50	82	75	87	84
-4	0	-	-	85	75	88	84
0	-0.8	-	-	81	73	86	83
0	1	-	-	80	73	87	85

Notes: $1.5/3.5/7.5$ m refer to position of the quadrotor along the $\mathbf{s}_1$-axis. The - sign represents the case when the initial position does not allow the identification of the target.

TABLE V
STUDY OF ANTICIPATION ERROR (6) IN SIMULATION

	1.5 m		3.5 m		7.5 m	
Axis	x	z	x	z	x	z
Simul. RMSE (m)	0.014	0.003	0.02	0.003	0.03	0.006
Real RMSE (m)	0.015	0.002	0.03	0.01	0.04	0.02

Notes: The distances $1.5/3.5/7.5$ m refer to the how far the quadrotor is from the target on the $\mathbf{s}_1$ component.

of the quadrotor and executed a perching maneuver with all visibility constraints active. The spatio-temporal visibility of the target is reported in Table IV . We notice that, on average, shifting the height of the quadrotor or moving it along the lateral does not have a major impact on the visibility constraint. In addition, in the aforementioned situations, the quadrotor is always able to successfully perch despite the positional shifts. We did not shift the height of the quadrotor for the 1.5 m case because the target starts outside the quadrotor's FoV in this case meaning we have no initial estimate to localize to or track before perching. This shows that our visibility constraints are fairly robust to different starting conditions.

We also evaluate the prediction accuracy of the anticipation (6). In this experiment, we consider a perching maneuver in simulation at various distances and calculated the root mean square error (RMSE) between the first predicted point and the quadrotor position when switching to a new trajectory. The results are shown in Table V . There is not much error on average between the true pose and replanning on the order of centimeters, showing that our prediction is quite accurate. We did not record the y-error in the following table since the quadrotor was aligned with the target and the error was negligible for anticipation or without anticipation. This experiment has also been conducted in the real world and a similar trend to simulation is observed.

Finally, we aim to study how AVP improves the perching maneuver with respect to one shot. In the first experiment, we artificially inject zero-mean Gaussian noise with different variances ranging from 0 to 1 in (2) on the state estimation. After injecting the noise in the state estimation, we set the quadrotor to perform a standard perching trajectory on a 90° incline in simulation from a distance of 1.7 m. This experiment is repeated for both one-shot and AVP approaches. The tracking RMSE between the planned and true paths is reported in Fig. 10. The proposed AVP approach shows increased robustness to

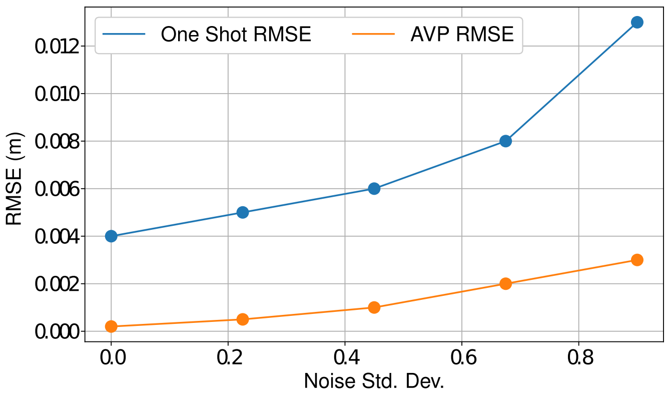


Fig. 10. Control error as a function of the error standard deviation in simulation. The quadrotor was tasked with perching on a target 1.7 m away at an incline of 90°.

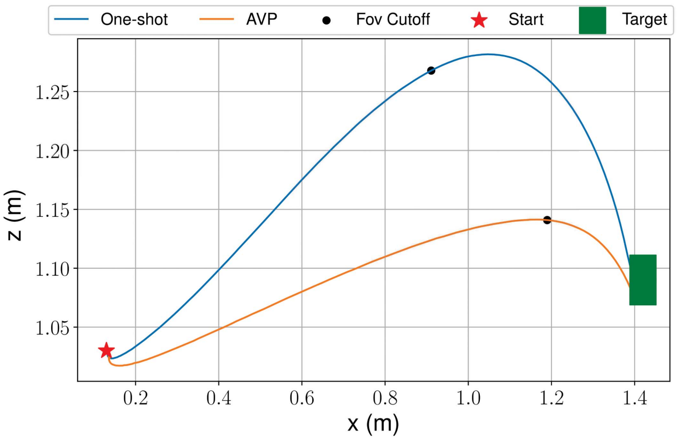


Fig. 11. Path comparison between the one-shot and the AVP on the x- and z-axis. This experiment was done trying to land on a 90° inclined surface with $\mathbf{v}_{s1} = -2$ and $\mathbf{v}_{s3} = 0.3$. Slight deviations between the end and start points are due to slight difference in AprilTag detection for real-world experiments and drift when take off. One-shot perching is done with the Vicon for its path estimation.

TABLE VI
ERROR OF THE QUADROTOR'S FINAL POSITION WITH RESPECT TO THE TARGET IN TERMS OF THE TWO AXES

Axis	1.5 m		3.5 m		7.5 m	
	One shot	AVP	One shot	AVP	One shot	AVP
$\mathbf{s}_2$ (m)	0.03	0.02	0.04	0.02	0.1	0.03
$\mathbf{s}_3$ (m)	0.04	0.05	0.06	0.05	0.25	0.09

Notes: 1.5/3.5/7.5 m refers to the initial starting distance of the quadrotor with respect to the target.

injected noise in terms of tracking error compared to the one-shot approach. These trials were repeated five times each. In addition, we demonstrated that AVP can improve target interception by calculating the final perched RMSE between the center of the target and the quadrotor or $||\mathbf{x} - \mathbf{s}||$. This process is repeated for various distances, as shown in Table VI . We use an average of five trials in simulation to calculate the average RMSE for the center error. For shorter distances, the difference in error is not significant. However, if we increase the distance, it becomes

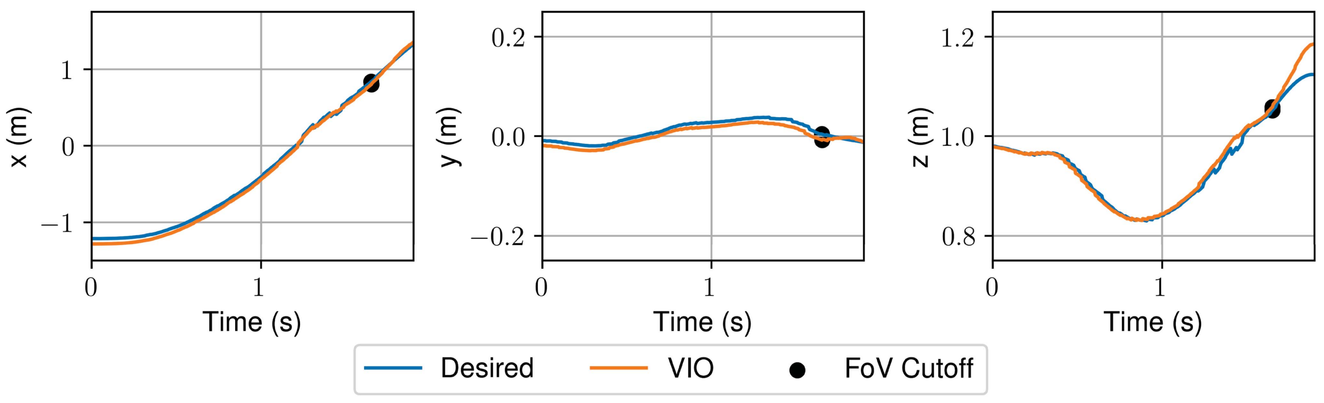


Fig. 12. Trajectory tracking and localization visualization for perching on a 90° surface inclination from a distance of 3 m. FoV cutoff is when the quadrotor loses track of the target.

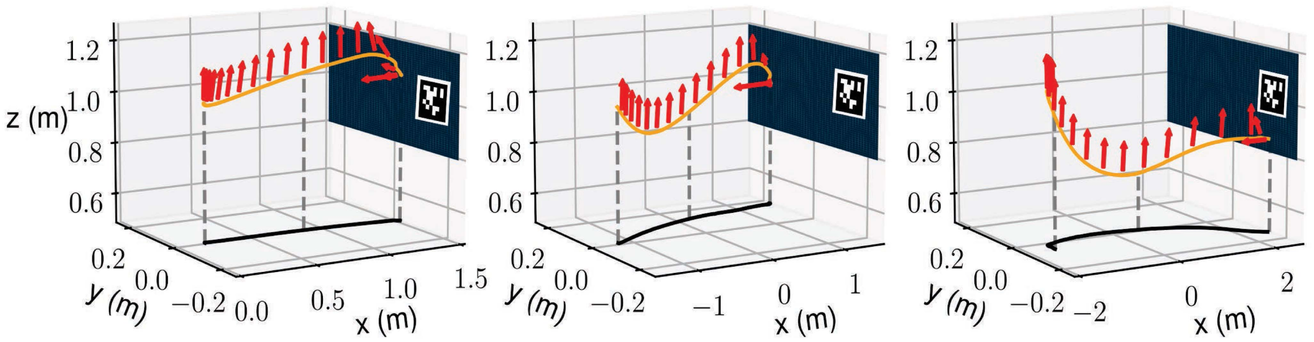


Fig. 13. 3-D path perching trajectory during from various distance, 1.5/3/4.5 m from left to right. The red arrow is the $\mathbf{b}_3$-axis of the quadrotor. The orange path is the quadrotor's path. The blue board is the target. The arrows represent the thrust vector. The blue surface represents the target.

clear that the AVP can better localize the target with repeated planning and intercept closer to the center especially at 7.5 m.

2) *Real-World Experiments:* We demonstrate the ability of our quadrotor, relying on on-board VIO and visual target localization, to perform aggressive perching maneuvers. The entire control, planning, and target localization pipeline runs on-board the vehicle. In this way, we verify that our system can autonomously perch in real-world settings. We additionally seek to prove that the AVP strategy provides increased accuracy and precision, guaranteeing reliable target interception compared to one-shot planning, especially when increasing the vehicle–target relative distance and for motions with high angular rates (up to 750°/s, as demonstrated in Fig. 15).

Our first set of experiments in the real world aims to compare a one-shot planning to AVP. We start by considering the perching problem relying on the Vicon motion capture system for quadrotor localization and employing the camera for target identification and localization. The use of the motion capture system for localization limits the variance between experiments introduced by VIO to obtain a more fair comparison between the one-shot planning and the proposed AVP method. In this way, we can identify the role and benefits or consecutive visual target detection. This experiment is repeated five times for each different target distance. The average results of the tracking error across all trials are presented in Table VII . We define

TABLE VII
TRACKING RMSE COMPARISON BETWEEN AVP AND ONE-SHOT PLANNING FOR 90° TARGET INCLINATION

Axis	1.5 m		3.5 m		4.5 m	
	One-shot	AVP	One-shot	AVP	One-shot	AVP
x (m)	0.08	0.07	0.05	0.05	0.04	0.03
y (m)	0.03	0.02	0.01	0.01	0.01	0.02
z (m)	0.04	0.03	0.03	0.01	0.06	0.02
$\mathbf{s}_2$ (m)	0.02	0.03	0.05	0.02	-	0.02
$\mathbf{s}_3$ (m)	0.11	0.05	0.15	0.05	-	0.06

Notes: All other axis have a distance of zero when starting. $\mathbf{s}_2$ and $\mathbf{s}_3$ refer to the endpoint error with respect to the target center. - refers to no available data due to unsuccessful perching.

tracking error as the difference between the quadrotor's desired position and the true location of the quadrotor. In addition, the final two rows $\mathbf{s}_2$ and $\mathbf{s}_3$ of the table quantify the distance of the vehicle with respect to the center of the target once perching is finalized. We notice a substantial improvement to target interception, especially once increasing the vehicle–target relative distance. This behavior is demonstrated across several trials in the attached multimedia material. One-shot planning is also unable to intercept a target distant 4.5 m along the $\mathbf{s}_3$. This is because for large distances visual target detection and localization become unreliable and therefore consecutive target interceptions once approaching the target to provide a clear

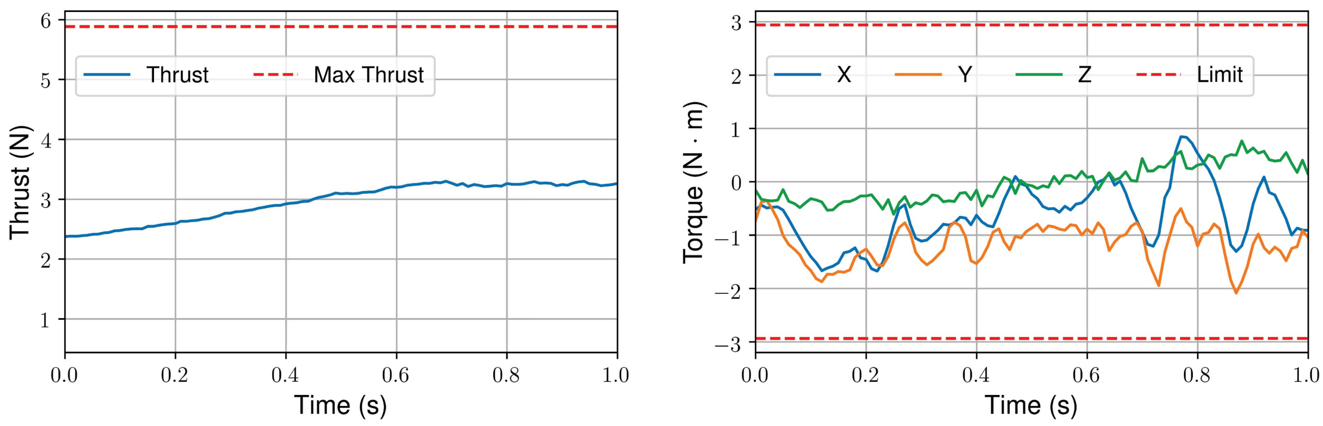


Fig. 14. Quadrotor's thrust and moments during perching. Values are calculated directly from motor RPMs readings.

TABLE VIII
LATERAL TARGET INTERCEPTION ERROR USING VICON FOR VEHICLE LOCALIZATION AND VISION FOR TARGET LOCALIZATION WITH AVP

$\mathbf{s}_1$ (m)	$\mathbf{s}_2$ (m)	$\mathbf{s}_3$ (m)	Error $\mathbf{s}_2$ (m)	Error $\mathbf{s}_3$ (m)
2.5	1	-0.1	0.03	0.05
2.5	-1	-0.1	0.03	0.04
3.5	0	0	0.02	0.05

benefit. In a separate table, we also show the target interception error when performing lateral movements in Table VIII for the AVP interception. Here, we observe that our target interception remains consistent with and without lateral movements.

An interesting property that comes out of these trials between one-shot and AVP is that the executed trajectories can be quite different even with accurate state estimation. This is demonstrated in Fig. 11 for a path with a 90° perching maneuver. We notice that the executed path for the one-shot trajectory case is quite different compared to the AVP trajectory (a similar behavior is experienced in simulation). Overall, in our tests, the trajectory generated by our AVP planner path is shorter.

Finally, we perform perching using vision for target and vehicle localization using VIO both running on-board concurrently with planning and control. A visualization of the estimation error is shown in Fig. 12 for a three-meter flight at 90°. We also visualize the paths and thrust executed by the quadrotor using VIO feedback in Fig. 13. The executed path respects the actuators and orientation constraints described in Section V-A, as shown in Figs. 13 and 14 . The vehicle correctly rotates before target interception and aligns its thrust direction with the target's normal surface. As observed in Fig. 13, the further the distance the worse the target interception becomes and the VIO drift also starts to be relevant. The estimation error is the difference between the Vicon ground truth and the VIO. We quantify the average result of the estimation error in ten flight trials in Table IX . We notice that as expected less aggressive maneuvers like perching on 60° presents lower tracking and estimation errors compared to more aggressive maneuvers required to perch at 90°. This is mainly due to two factors. First, by increasing the surface inclination, the vehicle will have to perform a larger rotational excursion producing an earlier relaxation of the FoV

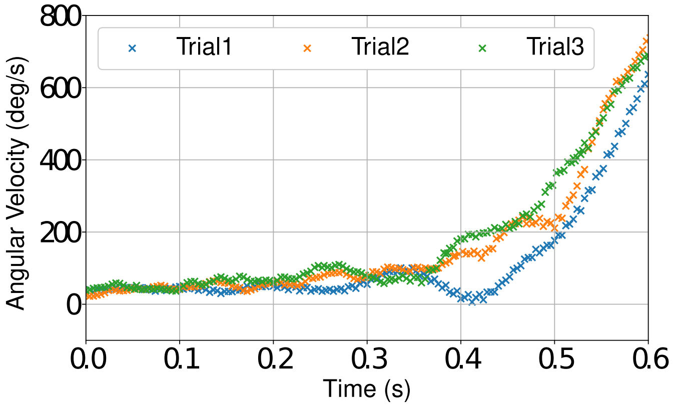


Fig. 15. Angular rate along the $\mathbf{b}_2$-axis during a series of 90° perching maneuvers.

TABLE IX
ESTIMATION RMSE FOR PERCHING USING VIO AND AVP

	Axis	1.5 m		3.5 m		4.5 m	
		One-shot	AVP	One-shot	AVP	One-shot	AVP
60°	x (m)	0.03	0.04	0.07	0.07	0.1	0.06
	y (m)	0.02	0.02	0.03	0.02	0.03	0.03
	z (m)	0.05	0.02	0.04	0.02	0.05	0.09
90°	x (m)	0.03	0.02	0.04	0.04	0.12	0.1
	y (m)	0.02	0.04	0.03	0.05	0.05	0.05
	z (m)	0.06	0.03	0.06	0.04	0.08	0.04

Notes: This is the error difference between the motion capture system and the VIO state estimation.

constraint. Second, the larger inclination will also increase the difficulty of concurrently guaranteeing a given attitude at a specified location in space due to the system underactuation. Therefore, by increasing the surface inclination, the vehicle has to gain an increased momentum and acceleration to perch. Additionally, we compare the target interception between one-shot and AVP using VIO for localization in Table X. The estimation error is small and similar in both cases due to the limited VIO drifts in the proposed traveled distances. Similar to our previous experiments with Vicon, one-shot perching is unable to intercept the target placed at 4.5 m distance and, on average, the AVP performs better than one-shot. We also verify that our trajectory is inside the thrust and moment bounds demonstrated in Fig. 14.

TABLE X
TARGET INTERCEPTION PERCHING ERROR AT 90° SURFACE INCLINATION CONSIDERING VIO FOR VEHICLE LOCALIZATION AND VISION FOR TARGET LOCALIZATION

Axis	1.5 m		3.5 m		4.5 m	
	One-shot	AVP	One-shot	AVP	One-shot	AVP
s_2 (m)	0.03	0.05	0.08	0.07	–	0.07
s_3 (m)	0.10	0.02	0.12	0.06	–	0.18

Finally, we compare our success rate of the current interception with our past work over ten trials for each surface inclination. For short distances, such as $1.5\,\mathrm{m}$, we do not see much of a difference from 80% for one-shot and 90% for AVP. For $3.5\,\mathrm{m}$, we notice a more substantial difference in success rate 70% for one-shot and 90% for AVP. At further distances, such as $4.5\,\mathrm{m}$, one-shot is unable to guarantee successful perching. Conversely, AVP guarantees a 60% success rate. Overall, the AVP can reduce and compensate for the effects of inaccurate target localization and control or estimation errors enabling successful and reliable perching compared to the one-shot solution.

VII. CONCLUSION

In this article, we proposed an AVP strategy that exploits in an active fashion midflight information, such as the vehicle odometry and new target localization, to improve the robustness to localization and control errors of our perching task compared to a one-shot approach. We theoretically and experimentally showed how to efficiently formulate and execute this problem as a QP optimization that incorporates actuator and FoV, several perching constraints, and boundary conditions. The KKT conditions further guarantee the feasibility of the planned maneuver. We experimentally analyze the effects of different boundary conditions on the spatio-temporal target visibility during the perching maneuver and identify the best parameters to maximize it. The proposed AVP consistently shows superior performances compared to a one-shot planning approach. This algorithm is still lightweight to run on-board concurrently with the target localization, on-board VIO, and control during flight.

Future works will focus on intercepting a moving target fully exploiting the potential of our planner to adapt in midflight. Additionally, it is relevant to improve our controller to incorporate other high-speed dynamic effects or disturbances, such as drag. We would also like to exploit past perching iterations to further refine the maneuver in a learning-based fashion. Finally, we would also like to explore the possibility to consider a more general object or surface detection algorithm to remove the need for a fiducial marker. This would also facilitate the execution of perching experiments in outdoor settings.

APPENDIX

A. Sturm's Theorem

Sturm's theorem [45] states that the number of roots for a polynomial $H(t)$ in an interval $[t_0, t_f]$ is equal to the difference in sign changes of the Sturm's sequence, (17), between $S(t_0)$ and $S(t_f)$

$$S(t) = \begin{cases} S_0(t) = H(t) \\ S_1(t) = \dot{H}(t) \\ S_{i+1}(t) = -Rm\left(S_{i-1}, S_i\right) \\ \vdots \\ S_N(t) = -Rm\left(S_{N-2}, S_{N-1}\right) \in \mathbb{R} \end{cases} \tag{17}$$

where $Rm(S_{i-1}, S_i)$ gives the algebraic remainder of $\dfrac{S_{i-1}}{S_i}$. To evaluate the number of roots between $t \in [0, 2]$ for $H(t) = t^4 + t^3 - t - 1$. First, calculate the Sturm sequence

$$S(t) = \begin{cases} S_0(t) = t^4 + t^3 - t - 1 \\ S_1(t) = 4t^3 + 3t^2 - 1 \\ S_2(t) = 0.1875t^2 + 0.75t + 0.9375 \\ S_3(t) = -32t - 64 \\ S_4(t) = -0.1875. \end{cases} \tag{18}$$

We evaluate this sequence's signs at $t = 0$ as $[-, -, +, -, -]$. This sequence has two sign changes. Next, we calculate the sequence at $t = 2$ as $[+, +, +, -, -]$. There is one sign change. Subtracting the number of sign changes at $t = 0$ from $t = 1$, we find one root for $H(t)$ in the domain $[0, 2]$.

Proof: To prove that the Sturm's theorem guarantees that the values are in the bound, we leverage the intermediate value theorem. The intermediate value theorem states that given a continuous function $H(t)$ whose domain contains the values $[t_0, t_f]$, then $\forall\ i \in [H(t_0), H(t_f)]$ there must exist a corresponding $t_i \in [t_0, t_f]$ such that $i = H(t_i)$. Since our trajectory is continuous, this theorem holds in our case. Now let us prove that our algorithm works by contradiction. Assume that there exists a $t_i \in [t_0, t_1]$ such that $H(t_i) > b$, where b is the global bound, and all conditions of Algorithm 1, $H(t_0) < b$, $H(t_f) < b$, and $H(t_i) - b \neq 0\, \forall\, t_i \in [t_0, t_f]$ are true.

If this is the case, then we can apply the intermediate value theorem and construct a domain $[t_0, t_i]$ and a range $[H(t_0), H(t_i)]$. We know that $H(t_0) < b$ and $H(t_i) > b$, then $b \in [H(t_0), H(t_i)]$. Therefore, based on the intermediate value theorem, there must exist a $t_j \in [t_0, t_i]$ such that $H(t_j) = b$. However, we see that $H(t_j) = b$ is a contradiction with respect to the condition $H(t_j) - b \neq 0\, \forall\, t_j \in [t_0, t_f]$. As $[t_0, t_i] \subset [t_0, t_f]$ by construction, this condition also holds true for all $t_j \in [t_0, t_i]$. Since this is a contradiction, there can exist no such number $t_i \in [t_0, t_1]$ such that $H(t_i) > b$ if our algorithm returns true. ■

B. Flat Output Polynomial Bounds

In this section, we derive upper bounds for the thrust rate, angular velocity, angular accelerations, and moments as functions of the flat outputs represented with polynomials. These condition can eventually be added and complement additional bounds we described in Section V-C especially if the motor low-level control setup is not reactive enough. The boundaries can be verified using the Sturm's on the derived polynomial bounds and implemented with minimal modifications. The following boundaries on thrust rate and moments are functions of the third-

and fourth-order derivatives of the trajectory, namely the jerk and snap. Therefore, since the third- or fourth-order derivative norm is already being minimized as our cost, adding additional time to our quadrotor will enable us to obtain a feasible trajectory since we are indirectly minimizing angular velocity, thrust rate, and moments by reducing the quadrotor jerk.

1) Thrust Rate: In our previous work [49], we showed that the thrust is a function of the flat outputs. Let m be the mass of the vehicle, $\mathbf{j}$ the jerk that represents the third-order derivative of the positional trajectory, the rotation of the quadrotor defined by (13) convention as $Z(\psi)$-$Y(\theta)$-$X(\phi)$, and $\dot{\tau}$ the thrust rate. It holds that

$$m\mathbf{b}_3 \cdot \mathbf{j} = \dot{\tau}. \tag{19}$$

We can then construct an upper bound of the dot product using the geometric definition

$$m\mathbf{b}_3 \cdot \mathbf{j} = m(||\mathbf{b}_3||)(||\mathbf{j}||)\cos\theta \le m(||\mathbf{b}_3||)(||\mathbf{j}||) \tag{20}$$

where θ is the angle between $\mathbf{b}_3$ and $\mathbf{j}$. By construction $||\mathbf{b}_3|| = 1$; therefore, we can create an upper bounded polynomial such that

$$\dot{\tau}^2 \le (m\,||\mathbf{j}||)^2 \le \dot{\tau}^2_{\max}. \tag{21}$$

We choose to use the squared norm to avoid the square root in the norm function. The function $(m||\mathbf{j}|)^2$ remains a polynomial as it is constructed solely through derivatives and multiplication, which means that the trajectory still remains in the group of polynomial functions.

2) Angular Velocity: Angular velocity is constructed from our formulation of the rotation matrix defined by (13). We consider our angular velocity vector components as $\mathbf{\Omega} = [\omega_1\,\omega_2\,\omega_3]^\top$

$$\begin{bmatrix}\omega_1\\ \omega_2\end{bmatrix} = \frac{m}{\tau}\begin{bmatrix}-\mathbf{b}_2^T\\ \mathbf{b}_1^T\end{bmatrix}\mathbf{j}. \tag{22}$$

By construction $\mathbf{b}_1$ and $\mathbf{b}_2$ are orthogonal. Therefore, we can form the following inequality based on the minimum thrust τ:

$$\omega_1^2 + \omega_2^2 \le 2\frac{m}{\tau}\,||\mathbf{j}||^2. \tag{23}$$

While this eq. (23) may not be a polynomial, an upper bounded polynomial can be found by simply rounding the function up as both thrust and norm of jerk are positive thus allowing us to apply Sturm's Theorem.

The third component of the angular velocity requires us to declare the Euler angle convention. We can derive the third component through the flat output as

$$\omega_3 = \frac{\cos\theta}{\cos\phi}\dot{\psi} - \omega_2\tan\phi. \tag{24}$$

Equation (24) has a singularity when the roll or $\phi = \pi/2$ unlike the formulation for the Euler angles in our previous work [49], which has the singularity on the pitch at the same value. Generally, as in the presented case, perching is a maneuver that relies on pitching more than rolling. This is because the front camera is placed along the forward motion direction of the quadrotor to facilitate to maintain the line of sight of the target. As a result, the quadrotor will be primarily pitching during the interception maneuver. An interception through rolling would make it impossible for a front camera to maintain the target in the FoV. As such, we set an assumption that the roll is bounded to the following angles:

$$-\frac{\pi}{4} \le \phi \le \frac{\pi}{4}. \tag{25}$$

Following this assumption, we can construct this bound on the angular velocity

$$\omega_3^2 \le \frac{1}{2}\left(\dot{\psi}\cos\theta - \omega_2\sin\phi\right)^2. \tag{26}$$

The expression $\dot{\psi}\cos\theta - \omega_2\sin\phi$ is upper bounded with the following equation:

$$\left(\dot{\psi}\cos\theta - \omega_2\sin\phi\right)^2 \le \dot{\psi}^2 + \omega_2^2 + 2|\dot{\psi}\omega_2|. \tag{27}$$

From this result, we can construct an upper bound on w_3^2 from (27) by substituting the absolute value function with $|x| < 0.1x^2 + 2.5$; this bound was picked because of its implementation simplicity and being fairly similar near the end of the angular velocity range

$$\begin{aligned}\omega_3^2 &\le \frac{1}{2}\left(\dot{\psi}^2 + \omega_2^2\right) + |\dot{\psi}\omega_2|\\ \omega_3^2 &\le \frac{1}{2}\left(\dot{\psi}^2 + \frac{m}{\tau}\,||\mathbf{j}||^2 + 0.2\frac{m}{\tau}\,||\mathbf{j}||^2\,\dot{\psi}^2\right) + 2.5.\end{aligned} \tag{28}$$

A closer bound on absolute value can be solved with a polynomial at the cost of an increased complexity. Combining (23) and (28), we can construct our upper bound as

$$\omega_1^2 + \omega_2^2 + \omega_3^2 \le \frac{5\,m}{2\tau}\,||\mathbf{j}||^2 + \frac{1}{2}\dot{\psi}^2 + \frac{m}{10\tau}\,||\mathbf{j}||^2\,\dot{\psi}^2 + 2.5. \tag{29}$$

The above upper bound function can be constructed by multiplication and derivatives of a polynomial with other polynomials and scalars. As a result, this means the function also can be perfectly represented by a polynomial. This means that the Sturm's algorithm can be applied to this formulation.

3) Angular Acceleration: The angular acceleration can be derived in a similar manner by taking another derivative. From (1), we can take two derivatives on the translational and given dynamics to form the following equality and substituting $\frac{d^4\mathbf{x}}{dt^4} = \mathbf{s}$:

$$m\mathbf{s} = 2\dot{\tau}\mathbf{R}\hat{\mathbf{\Omega}}\mathbf{e}_3 + \tau\mathbf{R}\hat{\mathbf{\Omega}}^2\mathbf{e}_3 + \tau\mathbf{R}\dot{\hat{\mathbf{\Omega}}}\mathbf{e}_3 + \ddot{\tau}\mathbf{R}\mathbf{e}_3. \tag{30}$$

We can formulate the first two components of the angular acceleration $\dot{\omega}_1$ and $\dot{\omega}_2$, by solving for $\dot{\hat{\mathbf{\Omega}}}\mathbf{e}_3$

$$\begin{aligned}\begin{bmatrix}\dot{\omega}_1\\ \dot{\omega}_2\end{bmatrix} &= \frac{m}{\tau}\begin{bmatrix}-\mathbf{b}_2^T\\ \mathbf{b}_1^T\end{bmatrix}\mathbf{s}\\ &\quad + \frac{2\dot{\tau}m}{\tau^2}\begin{bmatrix}\mathbf{b}_2^T\\ -\mathbf{b}_1^T\end{bmatrix}\mathbf{j} + \omega_3\frac{m}{\tau}\begin{bmatrix}-\mathbf{b}_1^T\\ \mathbf{b}_2^T\end{bmatrix}\mathbf{j}.\end{aligned} \tag{31}$$

We can construct an upper bound in the form of a polynomial as

$$\dot{\omega}_1^2 + \dot{\omega}_2^2 \le 2\frac{m}{\tau}\,||\mathbf{s}||^2 + 4\frac{m\dot{\tau}}{\tau^2}\,||\mathbf{j}||^2 + \frac{m\omega_3}{\tau}\,||\mathbf{j}||^2. \tag{32}$$

Moving forward, we can substitute ω_3 with a variation of (26) to form the following inequality:

$$\dot{\omega}_1^2 + \dot{\omega}_2^2 \leq 2\frac{m}{\tau}\,||\mathbf{s}||^2 + 4\frac{m\dot{\tau}}{\tau^2}\,||\mathbf{j}||^2 + \frac{m}{\tau}\,||\mathbf{j}||^2\left(|\dot{\psi}| + \frac{m}{\tau}\,||\mathbf{j}||\right). \tag{33}$$

While absolute value is not a polynomial, it can be substituted with a polynomial bound that follows it fairly similarly. We can further derive $\dot{\omega}_3$ from the relationship of our rotation matrix construction described in (13) considering a *Z-Y-X* convention

$$\dot{\omega}_3 = \omega_3 \tan\phi + \frac{1}{\cos\phi}\left(\ddot{\psi}\cos\theta - \dot{\psi}\dot{\theta}\sin\theta - \dot{\omega}_2\sin\phi - \omega_2\dot{\phi}\cos\phi\right). \tag{34}$$

A simplified version factoring out the Euler angle derivatives for pitch $\dot{\theta}$ and roll $\dot{\phi}$ can be generated if we also set the same roll limit described in (25)

$$\dot{\omega}_3 \leq \frac{\sqrt{2}}{2}\left(\left|\ddot{\psi}\right| + \sqrt{\dot{\omega}_2^2 + (\omega_1\omega_2)^2} + \dot{\psi}^2 + \left|\dot{\psi}\omega_2\right|\right) + |\omega_3|. \tag{35}$$

We can let (32) be represented as $F(\dot{\psi}, \mathbf{a}, \mathbf{j}, \mathbf{s})$ and (35) be represented as $G(\ddot{\psi}, \dot{\psi}, \mathbf{a}, \mathbf{j}, \mathbf{s})$, and substituting in, respectively, (22) and (24), we obtain

$$\dot{\omega}_1^2 + \dot{\omega}_2^2 + \dot{\omega}_3^2 \leq F\left(\psi, \dot{\psi}, \mathbf{a}, \mathbf{j}, \mathbf{s}\right) + G\left(\psi, \dot{\psi}, \ddot{\psi}, \mathbf{a}, \mathbf{j}, \mathbf{s}\right)^2. \tag{36}$$

4) Moments: We can derive an upper bound on the moments based on (1). By considering a linear relationship between force and moments with motor speeds [41], by deriving a boundary on the moment, then it is simple to derive a boundary on the motor speeds. Without loss of generality, to make the derivation simpler, we assume that the components are solely diagonal for our inertial matrix. This is also a common assumption on most quadrotors. There exists a more complex version for a nondiagonal inertial matrix. Let the matrix components of the inertial matrix be derived as

$$\mathbf{J} = \begin{bmatrix} I_{xx}^2 & 0 & 0 \\ 0 & I_{yy}^2 & 0 \\ 0 & 0 & I_{zz}^2 \end{bmatrix}. \tag{37}$$

First, let $\dot{\mathbf{\Omega}} \leq H\left(\ddot{\psi}, \dot{\psi}, \mathbf{j}, \mathbf{s}\right)$, where $H\left(\ddot{\psi}, \dot{\psi}, \mathbf{j}, \mathbf{s}\right)$ is the inequality derived in the previous section in (29), it holds that

$$\mathbf{M} = \mathbf{J}\dot{\mathbf{\Omega}} + \mathbf{\Omega} \times \mathbf{J}\mathbf{\Omega}. \tag{38}$$

Leveraging the definition of our inertial matrix, we can formulate this cross product as a product of a constant matrix and a vector as

$$\mathbf{\Omega} \times \mathbf{J}\mathbf{\Omega} = \mathbf{I}\begin{bmatrix} \omega_2\omega_3 \\ \omega_1\omega_3 \\ \omega_1\omega_2 \end{bmatrix} \tag{39}$$

with

$$\mathbf{I} = \begin{bmatrix} \left(I_{zz}^2 - I_{yy}^2\right) & 0 & 0 \\ 0 & \left(I_{xx}^2 - I_{zz}^2\right) & 0 \\ 0 & 0 & \left(I_{yy}^2 - I_{xx}^2\right) \end{bmatrix}. \tag{40}$$

The constant matrix $\mathbf{J}_2$ can be precomputed from the real inertia values. By substituting (23) and (28), we can form the following bound:

$$\mathbf{\Omega} \times \mathbf{J}\mathbf{\Omega} \leq \mathbf{I}\begin{bmatrix} \frac{\sqrt{2}m}{2\tau}||\mathbf{j}||\left(|\dot{\psi}| + \frac{m}{\tau}\,||\mathbf{j}||\right) \\ \frac{\sqrt{2}m}{2\tau}||\mathbf{j}||\left(|\dot{\psi}| + \frac{m}{\tau}\,||\mathbf{j}||\right) \\ \frac{m^2}{\tau^2}||\mathbf{j}||^2 \end{bmatrix}. \tag{41}$$

We can combine the above inequality $\mathbf{\Omega} \times \mathbf{J}\mathbf{\Omega} \leq K(\dot{\psi}, \mathbf{a}, \mathbf{j}, \mathbf{s})$ with (29) to create a bound on the moments as

$$\mathbf{M} \leq \mathbf{J}H\left(\psi, \dot{\psi}, \ddot{\psi}, \mathbf{a}, \mathbf{j}, \mathbf{s}\right) + K\left(\psi, \dot{\psi}, \mathbf{a}, \mathbf{j}, \mathbf{s}\right). \tag{42}$$

The relationship between moments and motor speeds is linear, which can be solved knowing the vehicle's hyperparameters. This can be applied to motor speed to generate bounds. The above inequalities are combinations of the flat outputs to formulate the bounds on thrust rate, angular velocity, angular acceleration, and moments. Normally, we can also apply this to our global bound checker as additional checks for robots where the low-level motor control setup is not reactive enough. It is also notable that (29) and (42) are functions of the jerk and snap of the polynomial. This implies that if we increase the time and minimize the cost on jerk/snap, the following costs will be reduced and will indirectly pull the planned trajectory within the bounds.

5) Moment Rate: Considering that the moment is linearly related to the quadrotor's motor speed, the moment rate is therefore linearly related to the motor's acceleration. In practice and supported by claims from past works [50], the quadrotor motor acceleration or moment rates can generally be neglected in planning because motor speed control is almost instantaneous compared to its rigid body motion. For less aggressive platforms, where motor acceleration is a concern, then similar to (42) another bound can be established by taking an additional derivative of (38) and solving again in terms of the flat outputs. As (42) is bounded by up to the fourth-order derivative of the position or the quadrotor's angular acceleration, the moment rate would be bounded as a function of the flat outputs up to the fifth-order derivative or the angular jerk. Several works [13], [20], [26], [50] do not consider either the fifth-order derivative of the flat outputs or the angular jerk in their planning for the aforementioned reasons.

References

[1] J. Mao, G. Li, S. M. Nogar, C. M. Kroninger, and G. Loianno, "Aggressive visual perching with quadrotors on inclined surfaces," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 5242–5248.
[2] C. Richter, A. Bry, and N. Roy, *Polynomial Trajectory Planning for Aggressive Quadrotor Flight in Dense Indoor Environments*. Cham, Switzerland: Springer, 2016, pp. 649–666.
[3] Z. Wang, X. Zhou, C. Xu, and F. Gao, "Alternating minimization based trajectory generation for quadrotor aggressive flight," *IEEE Robot. Autom. Lett.*, vol. 5, no. 3, pp. 4836–4843, Jul. 2020.

[4] F. Gao, W. Wu, J. Pan, B. Zhou, and S. Shen, "Optimal time allocation for quadrotor trajectory generation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 4715–4722.

[5] D. Burke, A. Chapman, and I. Shames, "Generating minimum-snap quadrotor trajectories really fast," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 1487–1492.

[6] M. Watterson, S. Liu, K. Sun, T. Smith, and V. Kumar, "Trajectory optimization on manifolds with applications to quadrotor systems," *Int. J. Robot. Res.*, vol. 39, no. 2/3, pp. 303–320, 2020.

[7] S. Liu, K. Mohta, N. Atanasov, and V. Kumar, "Search-based motion planning for aggressive flight in SE(3)," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 2439–2446, Jul. 2018.

[8] V. Agrawal and F. Dellaert, "Continuous-time state and dynamics estimation using a pseudo-spectral parameterization," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 426–432.

[9] P. Foehn, A. Romero, and D. Scaramuzza, "Time-optimal planning for quadrotor waypoint flight," *Sci. Robot.*, vol. 6, no. 56, 2021, Art. no. eabh1221.

[10] F. Morbidi, R. Cano, and D. Lara, "Minimum-energy path generation for a quadrotor UAV," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2016, pp. 1492–1498.

[11] J. L. Paneque et al., "Perception-aware perching on powerlines with multirotors," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 3077–3084, Apr. 2022.

[12] R. Verschueren et al., "acados–A modular open-source framework for fast embedded optimal control," *Math. Program. Comput.*, vol. 14, no. 1, pp. 147–183, 2022.

[13] J. Thomas et al., "Aggressive flight for perching on inclined surfaces," *J. Mechanisms Robot.*, vol. 8, 2015, Art. no. 051007.

[14] G. Loianno, C. Brunner, G. Mcgrath, and V. Kumar, "Estimation, control, and planning for aggressive flight with a small quadrotor with a single camera and IMU," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 404–411, Apr. 2017.

[15] Y. Chen and N. O. Pérez-Arancibia, "Controller synthesis and performance optimization for aerobatic quadrotor flight," *IEEE Trans. Control Syst. Technol.*, vol. 28, no. 6, pp. 2204–2219, Nov. 2020.

[16] S. Lupashin, A. Schöllig, M. Sherback, and R. D'Andrea, "A simple learning strategy for high-speed quadrocopter multi-flips," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2010, pp. 1642–1648.

[17] E. Kaufmann, A. Loquercio, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Deep drone acrobatics," *Robot.: Sci. Syst.*, 2020.

[18] B. Habas, B. AlAttar, B. Davis, J. W. Langelaan, and B. Cheng, "Optimal inverted landing in a small aerial robot with varied approach velocities and landing gear designs," in *Proc. Int. Conf. Robot. Automat. (ICRA)*, 2022, pp. 2003–2009, doi: 10.1109/ICRA46639.2022.9812409.

[19] J. Thomas, G. Loianno, K. Daniilidis, and V. Kumar, "Visual servoing of quadrotors for perching by hanging from cylindrical objects," *IEEE Robot. Autom. Lett.*, vol. 1, no. 1, pp. 57–64, Jan. 2016.

[20] H. Zhang, B. Cheng, and J. Zhao, "Optimal trajectory generation for time-to-contact based aerial robotic perching," *Bioinspiration Biomimetics*, vol. 14, 2018, Art. no. 016008.

[21] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2011, pp. 3400–3407.

[22] V. Usenko, L. Von Stumberg, A. Pangercic, and D. Cremers, "Real-time trajectory replanning for MAVs using uniform b-splines and a 3D circular buffer," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 215–222.

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

[24] B. Penin, P. R. Giordano, and F. Chaumette, "Vision-based reactive planning for aggressive target tracking while avoiding collisions and occlusions," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 3725–3732, Oct. 2018.

[25] D. Falanga, A. Zanchettin, A. Simovic, J. Delmerico, and D. Scaramuzza, "Vision-based autonomous quadrotor landing on a moving platform," in *Proc. IEEE Int. Symp. Saf., Secur. Rescue Robot.*, 2017, pp. 200–207.

[26] J. Ji, T. Yang, C. Xu, and F. Gao, "Real-time trajectory planning for aerial perching," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 10516–10522.

[27] W. Chi, K. H. Low, K. H. Hoon, and J. Tang, "An optimized perching mechanism for autonomous perching with a quadrotor," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2014, pp. 3109–3115.

[28] K. M. Popek et al., "Autonomous grasping robotic aerial system for perching (agrasp)," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 1–9.

[29] W. Roderick, M. Cutkosky, and D. Lentink, "Bird-inspired dynamic grasping and perching in arboreal environments," *Sci. Robot.*, vol. 6, no. 61, 2021, Art. no. eabj7562.

[30] A. Kalantari, K. Mahajan, D. Ruffatto, and M. Spenko, "Autonomous perching and take-off on vertical walls for a quadrotor micro air vehicle," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 4669–4674.

[31] E. W. Hawkes et al., "Dynamic surface grasping with directional adhesion," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2013, pp. 5487–5493.

[32] L. Daler, A. Klaptocz, A. Briod, M. Sitti, and D. Floreano, "A perching mechanism for flying robots using a fibre-based adhesive," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2013, pp. 4433–4438.

[33] C. C. Kessens, J. Thomas, J. P. Desai, and V. Kumar, "Versatile aerial grasping using self-sealing suction," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2016, pp. 3249–3254.

[34] S. Backus, J. Izraelevitz, J. Quan, R. Jitosho, E. Slavick, and A. Kalantari, "Design and testing of an ultra-light weight perching system for sloped or vertical rough surfaces on mars," in *Proc. IEEE Aerosp. Conf.*, 2020, pp. 1–12.

[35] H. W. Wopereis, T. D. van der Molen, T. H. Post, S. Stramigioli, and M. Fumagalli, "Mechanism for perching on smooth surfaces using aerial impacts," in *Proc. IEEE Int. Symp. Saf., Secur., Rescue Robot.*, 2016, pp. 154–159.

[36] S. Park, D. S. Drew, S. Follmer, and J. Rivas-Davila, "Lightweight high voltage generator for untethered electroadhesive perching of micro air vehicles," *IEEE Robot. Autom. Lett.*, vol. 5, no. 3, pp. 4485–4492, Jul. 2020.

[37] M. T. Pope et al., "A multimodal robot for perching and climbing on vertical outdoor surfaces," *IEEE Trans. Robot.*, vol. 33, no. 1, pp. 38–48, Feb. 2017.

[38] A. L. Desbiens, A. T. Asbeck, and M. R. Cutkosky, "Landing, perching and taking off from vertical surfaces," *Int. J. Robot. Res.*, vol. 30, no. 3, pp. 355–370, 2011.

[39] J. Moore, R. Cory, and R. Tedrake, "Robust post-stall perching with a simple fixed-wing glider using LQR-trees," *Bioinspiration Biomimetics*, vol. 9, 2014, Art. no. 25013.

[40] D. Mehanovic, J. Bass, T. Courteau, D. Rancourt, and A. L. Desbiens, "Autonomous thrust-assisted perching of a fixed-wing UAV on vertical surfaces," in *Proc. Conf. Biomimetic Biohybrid Syst.*, 2017, pp. 302–314.

[41] G. Loianno, C. Brunner, G. McGrath, and V. Kumar, "Estimation, control, and planning for aggressive flight with a small quadrotor with a single camera and IMU," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 404–411, Apr. 2017.

[42] T. Lee, M. Leok, and N. H. McClamroch, "Nonlinear Robust Tracking Control of a Quadrotor UAV on SE(3)," *Asian J. Control*, vol. 15, no. 2, pp. 391–408, 2013.

[43] G. Loianno et al., "Smartphones power flying robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2015, pp. 1256–1263.

[44] G. Li, A. Tunchez, and G. Loianno, "PCMPC: Perception-constrained model predictive control for quadrotors with suspended loads using a single camera and IMU," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 2012–2018.

[45] C. Sturm, "Collected works of Charles François Sturm," *Inst. France Sci. Math. Phys.*, vol. 6, pp. 345–390, 2009.

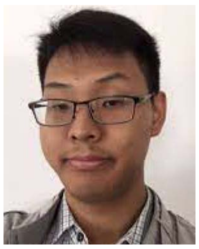
[46] Y. Song, S. Naji, E. Kaufmann, A. Loquercio, and D. Scaramuzza, "Flightmare: A flexible quadrotor simulator," in *Proc. Conf. Robot Learn.*, 2020, pp. 1147–1157.

[47] E. M. Gertz and S. J. Wright, "Object-oriented software for quadratic programming," *ACM Trans. Math. Softw.*, vol. 29, no. 1, pp. 58–81, Mar. 2003.

[48] D. Kraft, "A software package for sequential quadratic programming," *Forschungsbericht- Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt Köln: Forschungsbericht*, 1988.

[49] G. Loianno et al., "A swarm of flying smartphones," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 1681–1688.

[50] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2011, pp. 2520–2525.



Jeffrey Mao (Graduate Student Member, IEEE) received the M.S. degree in electrical and computer engineering in 2019 from New York University (NYU), New York, NY, USA, where he is currently working toward the Ph.D. degree with the Agile Robotics and Perception Lab under the guidance of Prof. Loianno

Prior to joining NYU, he was with Intel, working on wearable embedded system devices. His research interest include aerial robotics, trajectory planning, navigation, Gaussian processes, and perception.

Stephen Nogar received the Ph.D. degree in aerospace engineering from Ohio State University, Columbus, OH, USA, in 2015.

He is currently an Autonomy Team Lead with DEVCOM Army Research Laboratory, Adelphi, MD, USA. His research interests include dynamics, control, perception, and multiagent teaming for autonomous systems.

Christopher M. Kroninger received the M.S. degree in aerospace engineering from Pennsylvania State University, State College, PA, USA, in 2008.

He is currently a Program Manager with the U.S. Army DEVCOM Army Research Laboratory, Adelphi, MD, USA. His research focuses on single and multiagent autonomy to include heterogeneous team control, agent-task assignment, and vision-based navigation.

Giuseppe Loianno (Member, IEEE) received the Ph.D. degree in robotics from the University of Naples "Federico II," Naples, Italy, in 2014.

He is currently an Assistant Professor with New York University, New York, NY, USA, and the Director of the Agile Robotics and Perception Lab working on autonomous robots. Prior to joining NYU, he was a Postdoctoral Researcher, Research Scientist, and Team Leader with GRASP Lab, University of Pennsylvania, Philadelphia, PA, USA. He has authored or coauthored more than 70 conference papers, journal papers, and book chapters. His research interests include perception, learning, and control for autonomous robots.

Dr. Loianno was the recipient of the NSF CAREER Award in 2022, DARPA Young Faculty Award in 2022, IROS Toshio Fukuda Young Professional Award in 2022, Conference Editorial Board Best Associate Editor Award at ICRA 2022, and Best Reviewer Award at ICRA 2016, and he was selected as Rising Star in AI from KAUST in 2023. He is also the Co-Chair of the IEEE RAS Technical Committee on Aerial Robotics and Unmanned Aerial Vehicles. He was the General Chair of the IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR) in 2021 as well as Program Chair in 2019, 2020, and 2022. His work has been featured in a large number of renowned international news and magazines.