Aggressive Visual Perching with Quadrotors on Inclined Surfaces

Jeffrey Mao[1], Guanrui Li[1], Stephen Nogar[2], Christopher Kroninger[2], and Giuseppe Loianno[1]

***Abstract*— Autonomous Micro Aerial Vehicles (MAVs) have the potential to be employed for surveillance and monitoring tasks. By perching and staring on one or multiple locations aerial robots can save energy while concurrently increasing their overall mission time without actively flying. In this paper, we address the estimation, planning, and control problems for autonomous perching on inclined surfaces with small quadrotors using visual and inertial sensing. We focus on planning and executing dynamically feasible trajectories to navigate and perch to a desired target location with on board sensing and computation. Our planner also supports certain classes of nonlinear global constraints by leveraging an efficient algorithm that we have mathematically verified. The on board cameras and IMU are concurrently used for state estimation and to infer the relative robot/target localization. The proposed solution runs in real-time on board a limited computational unit. Experimental results validate the proposed approach by tackling aggressive perching maneuvers with flight envelopes that include large excursions from the hover position on inclined surfaces up to 90°, angular rates up to 600 deg/s, and accelerations up to 10 m/s^2.**

I. Introduction

Micro Aerial Vehicles (MAVs) have great speed and maneuverability however they tend to have very low flight time. Current solutions limit battery life to around 10-20 minutes. Fortunately for many missions such as environmental monitoring, it is unnecessary to remain in hover for the whole mission duration. In general, by perching and staring on one or multiple location, a MAV can greatly extend its mission time saving power without the need to frequently replace batteries. This motivates the need of autonomous perching solutions to conserve energy and extend the mission time.

In this paper, we tackle the autonomous perching problem on inclined surfaces with quadrotors solely using on board cameras and Inertial Measurement Unit (IMU) as shown in Fig. 1. Inclined flat surfaces like walls and rooftops are plentiful especially in urban environments and by focusing on this avenue, we can aim to greatly reduce the energy consumption of multiple types of mission. The proposed autonomous perching problem is challenging for several reasons. The maneuver, to intercept the target, requires large excursions from the hover position. In addition, the vehicle must generate and execute dynamically feasible trajectories respecting the actuator and sensor constraints despite the presence of nonholonomic and underactuation constraints. Finally, in our case, the maneuver has to be accomplished relying exclusively on on board minimalist sensor data (camera and IMU) and limited computational unit. The ability to execute these challenging maneuvers can be leveraged as well in several other scenarios including reaction to sudden changes in the operational conditions for obstacle avoidance or navigation in constrained environments.

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

This paper presents multiple contributions. First, we show how to generate and execute dynamically and physically feasible trajectories for perching on inclined surfaces. Our planning solution efficiently supports certain classes of nonlinear constraints such as maximum thrust limit through the use of an efficient bound checking algorithm that we have mathematically verified. The planner versatility to incorporate a diverse set of constraints makes it potentially able to support different perching or adhesion mechanisms. Second, our approach relies solely on on board sensing and computation for navigation and to infer the relative robot/target configuration. Finally, this is the first time that a fully autonomous quadrotor system can perch on any flat inclined surface with minimum mechanical modifications. Other works either assume the availability of a motion capture system [1] or require the landing point to be in the field of view at the starting location based on visual servoing approaches [2], [3].

II. Related Works

Prior works on perching with quadrotors on inclined surfaces focus on solving the trajectory generation and/or control problems [4], [1] relying exclusively on motion capture systems. These solutions do not address the design challenges and requirements when deploying robots equipped with embedded on board sensors and computationally limited units. Furthermore, the approach presented in [4] relies on composition of multiple control modes with linearized controllers without guaranteeing feasibility of the maneuver. Other works focus on the perching mechanism design. The approach proposed in [5] [6] use claws which limits perching to cylindrical objects of the appropriate

[1]The authors are with the New York University, Tandon School of Engineering, Brooklyn, NY 11201, USA email: {jm7752, lguanrui, loiannog}@nyu.edu.

[2]The authors are with the U.S. Army Research Laboratory, 2800 Powder Mill Road, Adelphi, MD 20783, USA. email: {stephen.m.nogar, christopher.m.kroninger}.civ@mail.mil.

This work was supported by the ARL grant DCIST CRA W911NF-17-2-0181 and the young researchers program "Rita Levi di Montalcini" 2017 grant PGR17W9W4N.

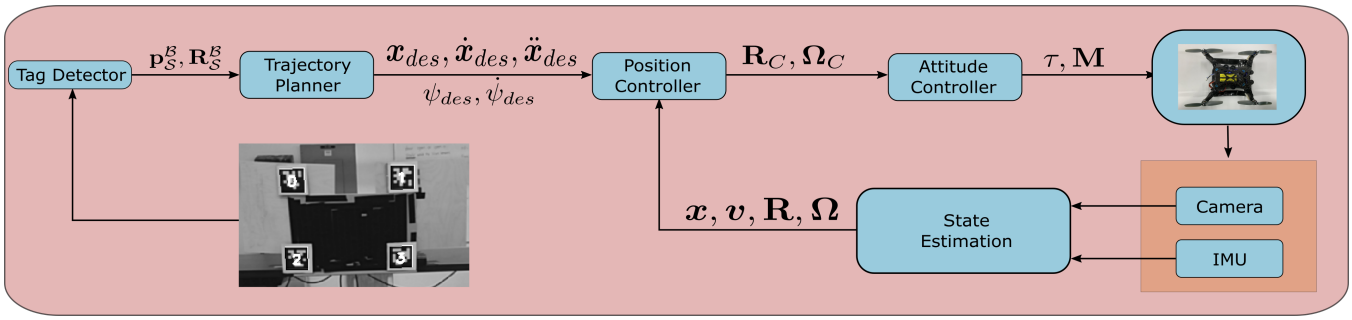

Fig. 2: System architecture for the perching task

width to fit the claw's grip. Other solutions rely on dry adhesives [7], [8], [9], suction gripper solutions [10], [11] mechanisms, or active perching mechanisms solutions [10]. However, the use of active perching mechanism solutions further increases the vehicle's payload and energy requirements, while concurrently decreasing the overall flight time. Finally, in [6] a bio-inspired trajectory planning approach is presented. However, the work relies on an actuated gripper for specific cylindrical objects and relies on a heavy 3D camera to localize perching targets. Finally, visual servoing approaches [2], [3] have shown autonomous perching results without the use of motions capture but are highly dependent on objects' shapes and require the object to initially be in the field of view.

Other works employ fixed wing solutions and focus on perching mechanism designs [12], [13]. Fixed wing solutions have lower maneuverability with respect to quadrotor solutions. Moreover, a quadrotor can hover in place and navigate in confined environments with both slow and fast agile movements. The flight time is more restricted with respect to fixed wing solutions further motivating the usefulness to provide autonomous perching solutions for quadrotors.

Compared to the aforementioned solutions, we concurrently guarantee dynamical and physical feasibility of the planned trajectories. Furthermore, our approach relies exclusively on on board sensing and estimation with a small computational unit. In the presented case, we focus on guaranteeing on board autonomous perching with dynamically and physically feasible trajectories. We leverage the differential flatness property and develop an efficient planning algorithm to generate trajectories for a quadrotor without the limitation to first order systems [14] or composition of multiple control modes relying on linearized controllers [4]. The proposed approach is agnostic with respect to the type of active/passive perching mechanism, due to the ability to support a diverse set of constraints which could conform to different attachment/adhesion mechanisms.

III. System Overview

The proposed system architecture is shown in Fig. 2 is a quadrotor running with a Qualcomm® Snapdragon™ board and 4 brushless motors. The Qualcomm® Snapdragon™ has a Qualcomm® Hexagon™ DSP, Wi-Fi, Bluetooth, GPS, one core processor along with a downward and front-facing camera with a 160° field of view along with an IMU. For perching, we employ VELCRO® material mounted in the ventral part of the vehicle.

The software framework has been developed in ROS[1] on a Linux kernel and includes a state estimation algorithm running at 500 Hz composed of a Unscented Kalman Filter (UKF) and Visual Inertial Odometry (VIO) [15] which processes images at 30 Hz. The downward facing camera is solely devoted to state estimation of the quadrotor and the front-facing camera is used to detect the perch target's position and orientation. Furthermore, the system runs on board a position and attitude controllers plus a trajectory planner to generate and execute planned path.

IV. Approach

Let the inertial frame $\mathcal{I}$ be represented by the following three axes $\begin{bmatrix}\mathbf{e}_1 & \mathbf{e}_2 & \mathbf{e}_3\end{bmatrix}$. The quadrotor body frame $\mathcal{B}$ is represented by $\begin{bmatrix}\mathbf{b}_1 & \mathbf{b}_2 & \mathbf{b}_3\end{bmatrix}$. This frame origin is located at the center of mass of the vehicle. Consequently, we denote the relative position of the quadrotor with respect to the $\mathcal{I}$ frame as $\boldsymbol{x} = \begin{bmatrix}x & y & z\end{bmatrix}^\top$ and the relative orientation as $\mathbf{R} = \begin{bmatrix}\mathbf{b}_1 & \mathbf{b}_2 & \mathbf{b}_3\end{bmatrix} \in SO(3)$. The perching target frame is denoted with $\mathcal{S}$ and is represented by the axes $\begin{bmatrix}\mathbf{s}_1 & \mathbf{s}_2 & \mathbf{s}_3\end{bmatrix}$. The relevant frames and configuration settings are depicted in Fig. 3. The perching problem from time $t = t_0$ to $t = t_f$ requires the vehicles to navigate by planning and executing a feasible trajectory (i.e., generating a sequence of $\mathbf{R}(t) \in SO(3)$ and $\boldsymbol{x}(t) \in \mathbb{R}^3$) such that $\mathcal{B} \equiv \mathcal{S}$ at $t = t_f$. The problem is decomposed in several steps. First, the vehicle visually locates the target and estimates the relative configurations 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 dynamically and physically feasible.

In Section IV-A, we briefly review the vehicle's system dynamics and control. Section IV-B describes the proposed perception pipeline including autonomous navigation and target localization. Section IV-C details our trajectory planning approach and how we leverage in this context the system dynamics and the relative configuration constraint to guarantee the correct plan and execution.

[1] www.ros.org

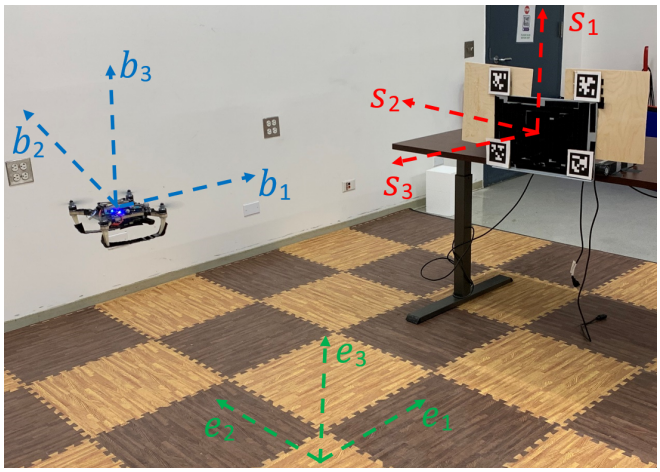

Fig. 3: Setup overview and frame convention definitions.

A. Modeling and Control

The system dynamic model in the inertial frame $\mathcal{I}$ is

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{v}, \dot{\boldsymbol{v}} = \boldsymbol{a}, m\boldsymbol{a} = \mathbf{R}\tau\mathbf{e}_3 - mg\mathbf{e}_3, \\ \dot{\mathbf{R}} &= \mathbf{R}\hat{\boldsymbol{\Omega}}, \mathbf{J}\dot{\boldsymbol{\Omega}} + \boldsymbol{\Omega} \times \mathbf{J}\boldsymbol{\Omega} = \mathbf{M}, \end{aligned} \tag{1}$$

where $\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{a} \in \mathbb{R}^3$ are the position, velocity, acceleration of the quadrotor's center of mass in Cartesian coordinates with respect to the inertial frame $\mathcal{I}$, $\mathbf{R}$ represents the orientation of the quadrotor with respect to $\mathcal{I}$. $\boldsymbol{\Omega} \in \mathbb{R}^3$ 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 $\mathcal{B}$, $g = 9.81m/s^2$ 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 $\hat{}$ 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 in our previous work [16] to achieve agile flight in indoor environments. First, $\mathbf{k}_R, \mathbf{k}_\Omega, \mathbf{k}_x, \mathbf{k}_v \in \mathbb{R}^{3\times3}$ are positive definite diagonal matrix representing the feedback gains for the errors in orientation, angular velocity, position and velocity respectively. Based on those feedbacks, the control inputs are thrust τ and moment $\mathbf{M}$ selected as

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

$\mathbf{e}_R, \mathbf{e}_\Omega, \mathbf{e}_x, \mathbf{e}_v \in \mathbb{R}^3$ are the orientation, angular velocity, position and velocity errors this is detailed in works [17], [18], and the $*_C$ are the values obtained from the differentially flat outputs. $*_{des}$ represents the differential flat outputs of the quadrotor system computed using the planning algorithm.

B. State Estimation and Perching Target Localization

For quadrotor autonomous navigation, we leverage our previous work [15], where we showed aggressive maneuvers combining visual and inertial data via VIO and UKF.

The perching localization method is unimportant for our system as long as it gives both position and orientation. For our experiments perching localization is achieved through the use of four Apriltags [19] placed on the four corners of the target. By placing these Apriltags on the corners, we can easily calculate the exact center of the landing pad by averaging these position while also leaving a large portion of VELCRO® to land on. Our software perception pipeline polls images from the front camera and runs an Apriltag localization algorithm. The quadrotor polls the front-facing camera till it recognizes all 4 Apriltags Ids, and takes the first sample image as the target configuration location $\mathbf{p}_{\mathcal{S}}^{\mathcal{B}}$ and orientation $\mathbf{R}_{\mathcal{S}}^{\mathcal{B}}$. These target configurations are represented in the $\mathcal{I}$ frame. To reduce sensitivity to the noise from the detection process, we average the tags' position and quaternions to obtain the target center position and orientation.

C. Planning for Aggressive Perching

After acquiring the target location and orientation, we plan a differentially smooth and dynamically feasible trajectory to satisfy the end goal of reaching the target and perching by exploiting the differential flatness property of the quadrotor. This allows us to shift the planning problem defined at the beginning of this Section to the flat space of the vehicle $\{\boldsymbol{x}, \psi\} = \{x, y, z, \psi\}$, where ψ is the yaw angle. We employ a set of polynomial splines P_d to represent the quadrotor trajectory

$$\begin{aligned} P_d(t) &= \begin{cases} p_{1d}\,(t - t_0) & \text{if } t \in [t_0, t_1] \\ p_{2d}\,(t - t_1) & \text{if } t \in [t_2, t_1] \\ \vdots & \\ p_{fd}\,(t - t_{f-1}) & \text{if } t \in [t_{f-1}, t_f] \end{cases}, \\ p_{id}(t) &= \sum_{n=0}^{N} c_{nid}t^n \;, i = 1, \cdots, f \end{aligned} \tag{3}$$

where f is the number of splines, N is the polynomial order, and $d \in \{1, 2, 3, 4\}$ corresponds to the dimensions of the flat space of the quadrotor system composed by the flat outputs $\{x, y, z, \psi\}$, p_{id} represents the i^{th} polynomial making up the full trajectory of P_d, $c_{nid} \in \mathbb{R}$ is the n^{th} coefficient of p_{id}. Polynomials are ideal because the function and its derivatives can be written as matrix multiplication.

In the following, we formulate the trajectory planning problem as a Quadratic Programming (QP). The trajectory optimization must satisfy perching, actuators, and sensing constraints globally in a time efficient manner. We first declare a cost function as the squared norm of the j^{th} order derivative summed in all dimensions. We formulate the case for one polynomial spline in the form

$$\min_{\mathbf{c}_{id}} \quad \mathbf{c}_{id}^\top \left(\int_{t_{i-1}}^{t_i} \frac{d^j\mathbf{t}_i}{dt^j} \frac{d^j\mathbf{t}_i}{dt^j}^\top dt \right) \mathbf{c}_{id} = \min_{\mathbf{c}_{id}} \mathbf{c}_{id}^\top \mathbf{Q}_i \mathbf{c}_{id} \tag{4}$$

where $\mathbf{t}_i = \left[1, (t - t_{i-1}), (t - t_{i-1})^2, \cdots, (t - t_{i-1})^N\right]^\top \in \mathbb{R}^N$ represents the time vector and and $\mathbf{c}_{id} \in \mathbb{R}^N$ is the

vector that consists of all the coefficients of p_{id}. The term $\mathbf{Q}_i \in \mathbb{R}^{N\times N}$ is the cost matrix formed from the center integral in eq. (4). It should be noted that $\mathbf{Q}_i$ is identical for all dimensions, d, so it has no subscript d. We can then convert eq. (4) into one unified cost for dimension d in the form $\mathbf{c}_d^T\mathbf{Q}\mathbf{c}_d$ by stacking the matrices diagonally $\text{Diag}(\mathbf{Q}_1,\cdots,\mathbf{Q}_i,\cdots,\mathbf{Q}_f) = \mathbf{Q}$ and forming $\mathbf{c}_d \in \mathbb{R}^{Nf}$ as a vector of all the $\mathbf{c}_{id}$ stacked vertically. We can then formulate an equality constraint for endpoints of each spline in the form $\mathbf{b}_{id} = \mathbf{A}_{id}\mathbf{c}_{id}$. The term $\mathbf{b}_{id}$ is a vector consisting of all the user defined constraints for dimension d that are imposed at time, t_i. $\mathbf{A}_{id}$ is defined by transposing $\mathbf{t}_i$ into a row vector then stacking $\mathbf{t}_i$ and its derivatives vertically as

$$\begin{bmatrix} p_{id}\left(t_i\right) \\ \dot{p}_{id}\left(t_i\right) \\ \vdots \end{bmatrix} = \begin{bmatrix} \mathbf{t}_i\left(t_i\right) & \dot{\mathbf{t}}_i\left(t_i\right) & \dots \end{bmatrix}^\top * \mathbf{c}_{id}. \tag{5}$$

The constraints on higher order terms like velocity and accelerations are optional to declare in eq. (5). We can simply stack matrices in eq. (5) diagonally, $\text{Diag}(\mathbf{A}_{1d},\cdots,\mathbf{A}_{id},\cdots,\mathbf{A}_{fd})$, to create the combined constraint for all trajectories in a dimension. Additionally, a continuity constraint between the endpoints of all splines is enforced for all derivatives. The first order derivative case is

$$\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} \dot{\mathbf{t}}_1(t_1) & -\dot{\mathbf{t}}_2(t_1) & 0 & \dots \\ 0 & \dot{\mathbf{t}}_2(t_2) & -\dot{\mathbf{t}}_3(t_2) & \dots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & \dot{\mathbf{t}}_{f-1}(t_{f-1}) & -\dot{\mathbf{t}}_f(t_{f-1}) \end{bmatrix} \begin{bmatrix} \mathbf{c}_{1d} \\ \mathbf{c}_{2d} \\ \vdots \\ \mathbf{c}_{fd} \end{bmatrix}. \tag{6}$$

Eq. (6) constraint is replicated to the fourth order and combined with eq. (5) constraints by stacking the matrices vertically creating a unified equality constraint matrix $\mathbf{A}_d$.

We can add additional inequality constraints formatted the same as eq. (5) $\mathbf{A}_{id}$ for each spline and combine them in the same way to get $\mathbf{G}_d$. The combined constraint and cost are solved as 4 separate QP optimization in parallel for each dimension to speed up the overall solution time. For each QP dimension the optimization problem is detailed below

$$\begin{aligned} \min_{\mathbf{c}_d} \quad & \mathbf{c}_d^T\mathbf{Q}\mathbf{c}_d \\ \text{s.t.} \quad & \mathbf{A}_d\mathbf{c}_d = \mathbf{b}_d \\ & \mathbf{y}_d \le \mathbf{G_d}\mathbf{c}_d \le \mathbf{z}_d \end{aligned} \tag{7}$$

where the inequality constraint are between a lower bound $\mathbf{y}_d$ and upper bound $\mathbf{z}_d$ respectively. In the following section, we will discuss the perception, state and actuator constraints specific to executing perching tasks.

1) Perching Perception and Physical Constraints: Inspired by [15], we exploit the differential flatness property of our model to derive a relationship between the vehicle's center of mass acceleration, $\ddot{\boldsymbol{x}}$ and rotation matrix $\mathbf{R}$. First we look at the dynamical model established in eq. (1). We can derive that the nominal thrust as

$$\tau = m||\ddot{\boldsymbol{x}} + g\mathbf{e}_3||. \tag{8}$$

Based on the nonholonomic property of a quadrotor system, the generated force is along the $\mathbf{b}_3$ axis of the body frame of a quadrotor; therefore, the $\mathbf{b}_3$ should satisfy

$$\mathbf{b}_3 = \frac{\ddot{\boldsymbol{x}} + g\mathbf{e}_3}{||\ddot{\boldsymbol{x}} + g\mathbf{e}_3||}. \tag{9}$$

Using eq. (9), we can define $\mathbf{b}_3$ at the end of the perching at t_f through an acceleration constraint as

$$\ddot{\boldsymbol{x}}(t_f) = \alpha\mathbf{s}_3 - g\mathbf{e}_3, \tag{10}$$

where $\alpha = ||\ddot{\boldsymbol{x}}(t_f) + g\mathbf{e}_3|| \in \mathbb{R}$ corresponds to the pre-defined thrust of the quadrotor selected by the user. The planned $\ddot{\boldsymbol{x}}$ is then injected in the thrust in eq. (2). 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 obtained combining the outcome of the state estimation algorithm with the target detector, which also provides $\mathbf{p}_\mathcal{S}$. These quantities allow us to fully define the target configuration with respect to the inertial frame and provide the constraints required to accomplish the perching maneuver. To enforce this maneuver at control level, the rotation matrix, we impose that in eq. (2), $\mathbf{R}_C$ should be chosen according to

$$\mathbf{R}_C = \begin{bmatrix} \mathbf{b}_{1,C} & \mathbf{b}_{2,C} & \mathbf{b}_{3,C} \end{bmatrix} \tag{11}$$

$$\mathbf{b}_{1,C} = \frac{\mathbf{b}_{2,des}\times\mathbf{b}_3}{||\mathbf{b}_{2,des}\times\mathbf{b}_3||}, \quad \mathbf{b}_{2,C} = \mathbf{b}_3\times\mathbf{b}_1,$$

$$\mathbf{b}_{2,des} = \begin{bmatrix} -\sin\psi_{des}, & \cos\psi_{des}, & 0 \end{bmatrix}^\top, \ \mathbf{b}_{3,C} = \frac{\mathbf{f}}{||\mathbf{f}||}.$$

The desired yaw angle ψ_{des} can be chosen by the user. Generally, we select it such that $\mathbf{b}_{2,des}$ is parallel to $\mathbf{s}_2$, which can be know from $\mathbf{R}_\mathcal{S}$. The commanded angular rate is then

$$\hat{\boldsymbol{\Omega}}_C = \mathbf{R}_C^\top\dot{\mathbf{R}}_C. \tag{12}$$

Next, additional constraints have to be placed on the impact velocity to ensure that the quadrotor is neither moving too quickly nor too slowly so that it does not properly adheres to the perching mechanism. We take this aspect into account by imposing additional velocity constraints in the target proximity

$$v_{min} \le \dot{\boldsymbol{x}}(t_f)\cdot\mathbf{s}_3 \le v_{max}, \tag{13}$$

where v_{min} is the minimum impact velocity and v_{max} is the maximum. Finally, the vehicle must complete most of its rotation before its impact with the surface rather than try to pivot at the target. This is to avoid the rim of the vehicle impacting the target surface. Should the vehicle begin rotating too late in its trajectory, the front end will collide with the landing pad before reaching the desired attitude. This aspect is expressed by enforcing an additional acceleration range by a given q tolerance in proximity of the target. In order to apply the inequality constraint to our optimization, we discretize the equation as

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

where dt is the sampling time of our trajectory planner and t_k is the time prior to the impact which is user defined.

Algorithm 1 GLOBAL BOUND CHECKING (GBC)

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

```
Let F(t) = H(t) − b;
if F(t_0) > 0  or  F(t_f) > 0 then
    return FALSE
end if
if STURM(F(t), t_0, t_f) > 0 then
    return FALSE
end if
return TRUE
```

2) Actuators Constraints: Our planner must respect the actuator constraints. If we refer to eq. (8), then it follows that there are lower and upper bounds τ_{min} and τ_{max} respectively that the vehicle's thrust τ should respect. We can then incorporate the actuator constraint formulated as

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

Since this constraint is nonlinear, it cannot be formulated in the QP optimization. We will describe how our system satisfies this condition in the next section.

3) Optimization Procedure: For our planner, we use the QP to solve the constraints described in eqs. (10), (13), and (14). Next, we use Algorithm 1 to verify if the constraint described in eq. (15) is met. We do not apply Algorithm 1 to the inequality constraints in eqs. (13) and (14) because they are linear constraints and can be efficiently resolved within the QP optimization. Should the global bounds be violated, we increase the trajectories' time iteratively and recursively solve the QP until eq. (15) is met. We visualize this process in Fig. 4 whereby increasing the allotted time for a minimum snap trajectory, the constraint from eq. (15) is met through iteratively expanding the time and checking the bounds.

Inspired by [20], Algorithm 1 checks if given any polynomial, $H(t)$, then is $H(t) < b \quad \forall \quad t \in [t_o, t_f]$ true. For our algorithm, $H(t)$ is set to the squared norm of the thrust described in eq. (15). Algorithm 1 works by leveraging the function STURM [21] which returns the number of roots of any arbitrary polynomial, $H(t)$, in a bound $t \in [t_0, t_f]$. A proof of algorithm 1 is included in the appendix along with a more detailed description of STURM's implementation. Using algorithm 1, we can ensure that our generated trajectory does not violate eq. (15) without checking each point of our trajectory. This time reduction is quantified in Table. I.

V. EXPERIMENTAL RESULTS

The experiments are conducted in the new indoor testbed with a flying space of $10 \times 5 \times 4$ m^3 at the ARPL lab at New York University. The ground truth data is collected using a Vicon[2] motion capture system at 100 Hz. Navigation is performed solely based on the on board VIO system running at 500 Hz. Our landing pad is mounted on an adjustable desk that allows us to control height. Also, the adhesive is attached to an adjustable stand which allows us to control the surface angle along a specific axis. We selected a tolerance, $q = 0.1$, sampling time, $dt = 0.01$ s, time before impact, $t_k = 0.15$ s and $\alpha = 3.3$ m/s^2 as the hyperparameters of eqs. (10) and (14). We choose to minimize the $j = 4$,snap norm, for eq. (4). Finally, we set the impact velocity bound of eq. (13) as the minimum impact velocity, $v_{min} = 0.4$m/s, and maximum impact velocity, $v_{max} = 0.6$m/s.

[2]www.vicon.com

Optimizer	Num. Waypoints	Computation Time (ms)
QP+GBC	3	4
	10	18
NLOPT	3	527
	10	1253

TABLE I: Computation time of our approach as a function of the number of waypoints.

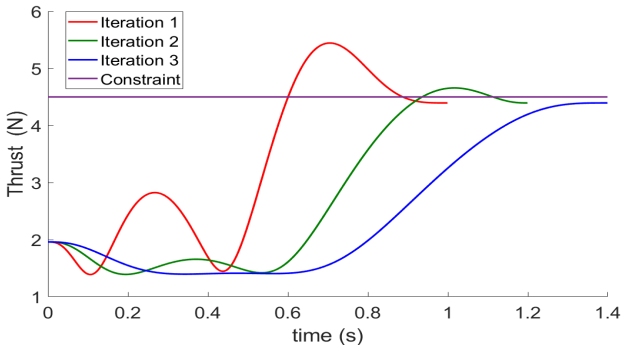

Fig. 4: Iterations through the GBC reducing thrust for perching trajectory.

First, we verify that our optimization procedure respects the maximum thrust bound in addition to the linear inequality constraints. This trajectory times spanned an initial $t_0 = 0$ s and end $t_f = 1$ s. We then apply a bound on the maximum thrust, $\tau_{max} = 4.5$ N. Visualized in Fig. 4, the thrust shrinks for each iteration as time increases. This process repeats till our condition is respected, where $t_f = 1.4$ s.

Next, we evaluate the computational time and scalability of our trajectory generator by analyzing the time taken to resolve a 3 and 10 spline problem. Computational results are reported in Table I. To ensure consistency of our results, we obtained the results by re-running the optimizer 5 times and averaging the outcome. We leverage the C++ OOQP library [22] to solve our QP formulation. For the following experiment, we compared whether using one iteration of the nonlinear optimizer package NLOPT [23] to guarantee thrust, or checking and reformulating multiple iterations of the QP problem was faster. The following experiments all use a polynomial order $N = 14$. We provide results for this specific scenario in Table I. The nonlinear optimization takes on the order of a 100 times longer to solve than performing a QP algorithm and repeating a check afterwards. This indicates that our current approach of doing multiple QP optimizations is better than trying to solve a single more complex nonlinear optimization.

We then proceed to evaluate our approach for perching on an inclined surface as shown in Fig. 1 considering two challenging inclination angles of 60° and 90°. In Fig. 5, we

rather large errors...

	Component	1.7 m		3 m	
		Tracking	Estimation	Tracking	Estimation
60°	x (m)	0.0500	0.0145	0.0378	0.0401
	y (m)	0.0325	0.0851	0.0732	0.0714
	z (m)	0.1509	0.0714	0.1321	0.0452
90°	x (m)	0.0503	0.0314	0.0948	0.0660
	y (m)	0.0539	0.0533	0.0360	0.0045
	z (m)	0.1081	0.0405	0.0911	0.0485

TABLE II: Tracking and Estimation RMSE for different surface inclinations and distances.

present the trajectory planning, control tracking, and localization results for the most challenging perching maneuver at $90°$. We do not report the results on y-axis in Fig. 5 because the motion along that axis does not have significant variations during the perching task. We also plot the thrust vector $\mathbf{b}_3$ in Fig. 5 (right) to further show that the constraint during perching is correctly enforced during the execution of the maneuver. As demonstrated, our additional constraints imposed through the eqs. (10) and (14) enforce the rotation to complete slightly before reaching the target. To further show that the proposed maneuver is aggressive and challenging, we present the angular rates achieved during $90°$ perching on the three Cartesian axes in Fig 6. Our vehicle achieves angular rates close to $600\ \mathrm{deg/s}$, which to the best of our knowledge has never been achieved in the past for a small scale vehicle using on board computation and visual perception for both state estimation and target localization.

Finally to validate the consistency, performance, and robustness of our algorithm, we use 2 different surface inclinations at two target distances of $1.7\ \mathrm{m}$ and $3\ \mathrm{m}$ respectively. $3\ \mathrm{m}$ is the maximum distance to reliably detect the Apriltags on our quadrotor. Table II shows the relevant trajectory tracking and state estimation RMSE metrics of these experiments. Furthermore, we repeat the procedure 5 times for each distance and surface inclinations and record the successful perching rate as seen in Table III. A successful perching is defined as the quadrotor adheres to the target and remains attached. The lower success rate at $60°$ is due to the Apriltag accuracy falling for a worse viewing angle and further target. The tracking error is mostly located toward the end of the trajectory as depicted in Fig. 5. Once the vehicle has angular velocity prior to perching, it is very difficult to control its position. However, we still see our controller succeeding in the perch in these conditions. Overall, we see that our approach can generate reliable perching maneuver based on on board perception with cm level accuracy both in term of localization and control tracking of the maneuver. Furthermore, these success rates and tracking errors are agnostic with respect to the tested starting positions as well as surface inclination.

VI. Conclusion

In this paper, we presented the planning, control, and perception methodologies to achieve autonomous visual perching with small quadrotors relying exclusively on on board computation and sensing. Our results show that we can generate aggressive and challenging perching maneuvers up to $90°$ inclined surfaces, angular rates up to $600\ \mathrm{deg/s}$, and accelerations up to $10\ \mathrm{m/s^2}$. These results show the agility and robustness of our real-time autonomous perching.

Target Distance (m)	Surface Angle	Success rate
1.7	60°	4/5
	90°	4/5
3	60°	3/5
	90°	4/5

TABLE III: Success rate statistics as a function of the inclined surface and the target distance.

Future works will focus on two research directions. First, we aim to leverage consecutive target detection across the entire planned maneuver to have a receding-horizon planning strategy. Continuous target information introduces the possibility to refine the trajectory in real-time to compensate for possible drifts or unmodelled effects during flight and consequently increases the resilience and precision for target interception. In this context, it is interesting to study the trade-off among the dynamic feasibility, aggressive behavior of the perching maneuver, and maximizing the visibility of the target. Second, we will work on enforcing the planning objectives and constraints at the control level by formulating a nonlinear model predictive control problem where the sensors and actuator constraints can be embedded as additional terms in the optimization cost function or as constraints.

Appendix

A. Sturm's Theorem

Sturm's theorem [21] 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, eq. (16), between $S(t_0)$ and $S(t_f)$

$$S(t) = \begin{cases} S_0(t) = H(t) \\ S_1(t) = \dot{H}(t) \\ S_{i+1}(t) = -Rm(S_{i-1}, S_i) \\ \vdots \\ S_N(t) = -Rm(S_{N-2}, S_{N-1}) \in \mathbb{R} \end{cases}, \quad (16)$$

where $Rm(S_{i-1}, S_i)$ gives the algebraic remainder of $\frac{S_{i-1}}{S_i}$.

B. Proof of Algorithm 1

First, we leverage the intermediate value theorem. The intermediate value theorem state that given a continuous function $H(t)$ whose domain contains the values $[t_0, t_f]$ then $\forall \quad i \in [H(t_0), H(t_f)]$ 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's prove that our algorithm works by contradiction. Assume, 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

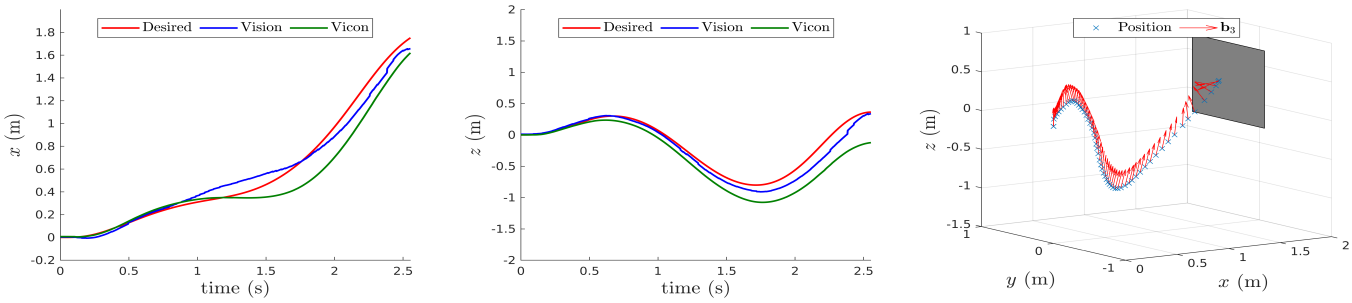

Fig. 5: Trajectory tracking and localization results for 90° surface inclination from a distance of 1.7 m. The blue crosses represent the quadrotor position, whereas the arrows represent the thrust vector.

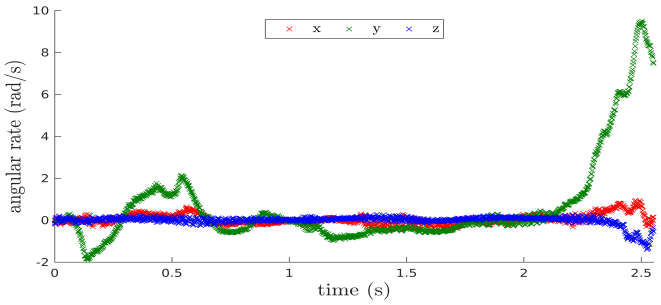

Fig. 6: Angular rate during a 90° perching maneuver.

$[H(t_0), H(t_i)]$. 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 $H(t_j) = b$ 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.

REFERENCES

[1] J. Thomas, M. Pope, G. Loianno, E. Hawkes, M. Estrada, H. Jiang, M. Cutkosky, and V. Kumar, "Aggressive flight for perching on inclined surfaces," *Journal of Mechanisms and Robotics*, vol. 8, 12 2015.

[2] J. Thomas, G. Loianno, K. Daniilidis, and V. Kumar, "Visual servoing of quadrotors for perching by hanging from cylindrical objects," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 57–64, 2016.

[3] H. Zhang, B. Cheng, and J. Zhao, "Optimal trajectory generation for time-to-contact based aerial robotic perching," *Bioinspiration & Biomimetics*, vol. 14, 10 2018.

[4] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," *The International Journal of Robotics Research*, vol. 31, no. 5, pp. 664–674, 2012.

[5] W. Chi, K. H. Low, K. H. Hoon, and J. Tang, "An optimized perching mechanism for autonomous perching with a quadrotor," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 3109–3115.

[6] K. M. Popek, M. S. Johannes, K. C. Wolfe, R. A. Hegeman, J. M. Hatch, J. L. Moore, K. D. Katyal, B. Y. Yeh, and R. J. Bamberger, "Autonomous grasping robotic aerial system for perching (agrasp)," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1–9.

[7] A. Kalantari, K. Mahajan, D. Ruffatto, and M. Spenko, "Autonomous perching and take-off on vertical walls for a quadrotor micro air vehicle," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 4669–4674.

[8] E. W. Hawkes, D. L. Christensen, E. V. Eason, M. A. Estrada, M. Heverly, E. Hilgemann, H. Jiang, M. T. Pope, A. Parness, and M. R. Cutkosky, "Dynamic surface grasping with directional adhesion," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013, pp. 5487–5493.

[9] L. Daler, A. Klaptocz, A. Briod, M. Sitti, and D. Floreano, "A perching mechanism for flying robots using a fibre-based adhesive," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013, pp. 4433–4438.

[10] H. Tsukagoshi, M. Watanabe, T. Hamada, D. Ashlih, and R. Iizuka, "Aerial manipulator with perching and door-opening capability," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 4663–4668.

[11] C. C. Kessens, J. Thomas, J. P. Desai, and V. Kumar, "Versatile aerial grasping using self-sealing suction," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 3249–3254.

[12] J. Moore, R. Cory, and R. Tedrake, "Robust post-stall perching with a simple fixed-wing glider using LQR-trees," *Bioinspiration & Biomimetics*, vol. 9, no. 2, May 2014.

[13] A. L. Desbiens, A. T. Asbeck, and M. R. Cutkosky, "Landing, perching and taking off from vertical surfaces," *The International Journal of Robotics Research*, vol. 30, no. 3, pp. 355–370, 2011.

[14] Z. Zhang, P. Xie, and O. Ma, "Bio-inspired trajectory generation for uav perching," in *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, 2013, pp. 997–1002.

[15] 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 Robotics and Automation Letters*, vol. PP, pp. 1–1, 11 2016.

[16] 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 Robotics and Automation Letters*, vol. 2, no. 2, pp. 404–411, April 2017.

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

[18] G. Loianno, Y. Mulgaonkar, C. Brunner, D. Ahuja, A. Ramanandan, M. Chari, S. Diaz, and V. Kumar, "Autonomous flight and cooperative control for reconstruction using aerial robots powered by smartphones," *The International Journal of Robotics Research*, vol. 37, no. 11, pp. 1341–1358, 2018.

[19] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2011, pp. 3400–3407.

[20] Z. Wang, X. Zhou, C. Xu, J. Chu, and F. Gao, "Alternating minimization based trajectory generation for quadrotor aggressive flight," 2020.

[21] P. Sturm, *Collected Works of Charles François Sturm*. Springer; 2009th edition, 01 2009, vol. 6, pp. 345–390.

[22] E. M. Gertz and S. J. Wright, "Object-oriented software for quadratic programming," *ACM Trans. Math. Softw.*, vol. 29, no. 1, p. 58–81, Mar. 2003. [Online]. Available: https://doi.org/10.1145/641876.641880

[23] S. G. Johnson, "The nlopt nonlinear-optimization package," https://github.com/stevengj/nlopt, 2007.