PANTHER: Perception-Aware Trajectory Planner in Dynamic Environments

JESUS TORDESILLAS, (Student Member, IEEE), AND JONATHAN P. HOW, (Fellow, IEEE)

Aerospace Controls Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139, USA

Corresponding author: Jesus Tordesillas (jtorde@mit.edu)

ABSTRACT This paper presents PANTHER, a real-time perception-aware (PA) trajectory planner for multirotor-UAVs (Unmanned Aerial Vehicles) in dynamic environments. PANTHER plans trajectories that avoid dynamic obstacles while also keeping them in the sensor field of view (FOV) and minimizing the blur to aid in object tracking. The rotation and translation of the UAV are jointly optimized, which allows PANTHER to fully exploit the differential flatness of multirotors to maximize the PA objective. Real-time performance is achieved by implicitly imposing the underactuated dynamics of the UAV through the Hopf fibration. PANTHER is able to keep the obstacles inside the FOV 7.9 and 1.5 times more than non-PA approaches and PA approaches that decouple translation and yaw, respectively. The projected velocity (and hence the blur) is reduced by 18% and 34%, respectively. This leads to average success rates three times larger than state-of-the-art approaches in multi-obstacle avoidance scenarios. The MINVO basis is used to impose low-conservative collision avoidance constraints in position and velocity space. Finally, extensive hardware experiments in unknown dynamic environments with all the computation running onboard are presented, with velocities of up to 5.8 m/s, and with relative velocities (with respect to the obstacles) of up to 6.3 m/s. The only sensors used are an IMU, a forward-facing depth camera, and a downward-facing monocular camera.

Video: https://youtu.be/jKmyW6v73tY
Code: https://github.com/mit-acl/panther

I. INTRODUCTION AND RELATED WORK

While the last decade has seen an increase on the number of successful deployments of multirotor-UAVs in different real-world scenarios, their applicability is often limited by two common assumptions, namely the fact that the environment is static, and/or the omnidirectional coverage of the sensor(s) of the UAV. Indeed, many UAVs have a limited FOV, and many applications (delivery, aerial videography, emergency response, etc.) have non-static environments due to the presence of cars, people, and/or other UAVs. Hence, relaxing these assumptions is critical to fully exploit the potential of the UAVs and expand the range of their possible applications.

When a UAV equipped with a limited FOV sensor is flying in an unknown environment (e.g., Fig 1), it is crucial to plan both the position and orientation of the UAV to maximize the detection and the tracking accuracy of the unknown

The associate editor coordinating the review of this manuscript and approving it for publication was Hui Xie.
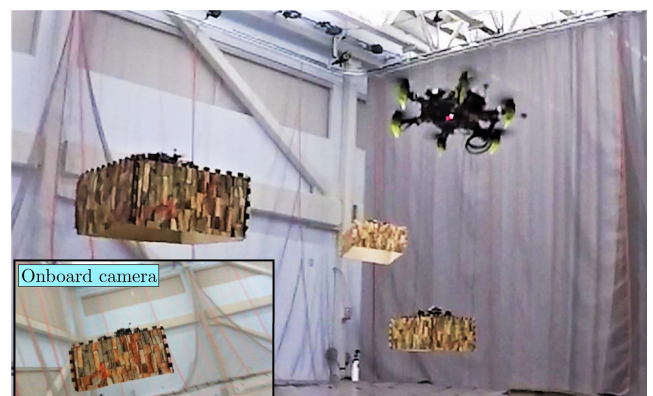
FIGURE 1. UAV planning perception-aware trajectories in a dynamic unknown environment, with relative velocities of up to 6.3 m/s. All the computation runs onboard, and the UAV does not have any prior knowledge of the trajectories or specific shape/size of the dynamic obstacles.

obstacles while at the same time doing obstacle avoidance. This perception-aware (PA) component is especially important when flying in dynamic environments, because a consistent detection of the moving obstacles is necessary to obtain a

TABLE 1. Classification of the related work, together with a (nonexhaustive) list of references.

Formulation		Ref.
(A.1) **Not PA**		[1]–[7]
(A.2) **PA with additional hardware**		[8]–[11]
(A.3) **PA planning**	**Decoupling**	[12]–[14]
	Joint opt.	[15]–[17, PANTHER]

Goal	Ref.
(B.1) **Reduce state estimation uncertainty**	[13]–[16], [18]–[27]
(B.2) **Record/chase a target**	[17], [24], [28]–[32]
(B.3) **Avoidance of dynamic obstacles**	PANTHER

good estimate of their locations and prediction of their future trajectories.

Perception-awareness for UAVs has been studied thoroughly in the literature, and, as shown in Table 1, the related work could be classified according to the formulation used and the goal itself. From the point of view of the formulation used, there are approaches that are **not PA** (A.1), which typically plan the translation and then have either a constant yaw or a yaw such that the FOV of the camera points in the direction of travel (e.g., see [1]–[7]). For instance, [5] used potential fields to avoid dynamic obstacles, but without taking into account perception-awareness, which can degrade the detection and prediction of the trajectories of the obstacles.

Other approaches are **PA by including additional hardware** (A.2): For example, by gimbal-mounting the camera, some of its degrees of freedom can be controlled independently of the rotation of the UAV [8]–[10]. Another option is to mount omni-directional sensors [11]. However, these approaches usually require additional hardware and mechanical complexity, which is typically undesirable on small UAVs.

PA planning (A.3) has received increased attention over the last few years due to its inherent ability to leverage the trajectory planned to maximize the PA objective. The related works could be subclassified according to whether or not the translation and yaw of the UAV are jointly optimized. On one hand, there are approaches that decouple translation and yaw by optimizing them separately [12]–[14]. For instance, in [12], a yaw trajectory is obtained for a fixed translational path to gain information about unknown static obstacles. For features or landmarks whose locations are known a priori, [13] optimizes the time parametrization on a fixed spatial and yaw path to maximize their visibility. In [14], translation is optimized first, and then yaw is optimized to guarantee the co-visibility of the features. While this decoupling of translation and yaw has computational advantages, it can lead to conservative results, since the translational trajectory (and consequently two degrees of freedom of the rotation as well) is fixed in the yaw optimization. Other works assume a downward-facing camera, and hence only translation (not yaw) is planned to keep a specific target in the FOV of the camera [28].

Another approach taken is to jointly optimize translation and yaw, which enables the planner to fully exploit *both* the position trajectory and the yaw angle [15]–[17].

This joint optimization leads to less conservative results than the approaches that decouple translation and yaw, but it typically comes at the expense of much higher computation times, especially when done in combination with dynamic obstacle avoidance constraints. For example, [15] proposed an on-manifold trajectory optimization approach that couples together translation with the full rotation, but the computation times required (up to 30 s) are not real time. Ref. [16] successfully presented a real-time MPC formulation that keeps the centroid of the VIO features in the center of the image while minimizing its projected velocity. However, this formulation does not include collision avoidance of static (or dynamic) obstacles, which greatly simplifies the complexity of the optimization problem. In [17], translation and yaw are optimized jointly, but only static obstacle avoidance is performed. The technical gap then is how to jointly optimize the full pose of the UAV, satisfy its underactuated dynamics, and guarantee safety in dynamic environments while maintaining real-time computational tractability.

The underactuated dynamics of the UAV (caused by the total thrust of the UAV being fixed in the body frame) makes this joint optimization especially hard, since a given spatio-temporal path fixes two degrees of freedom of the rotation, leaving only one extra degree of freedom in the rotation.[1] A typical way to impose this constraint is via the dynamic equations of the UAV. However, this comes at the expense of having differential equations as constraints in the optimization.

An alternative is to leverage the differential flatness of the UAVs [33] and make use of the map ($\mathbf{a} \in \mathbb{R}^3 \setminus [0 \ 0 \ -g]^T$, $\psi \in S^1$) $\rightarrow \mathbf{R}_b^w \in SO(3)$ that maps ψ and the acceleration $\mathbf{a}$ to the rotation of the body. However, and due to the *hedgehog theorem* in S^2 [34], [35], there is no single continuous function that defines this map for all possible accelerations $\mathbf{a}$. For the most common definitions of this map, the singularity appears for each ψ at two antipodal points in the unit sphere of possible normalized relative accelerations, which means that there is at least one singularity with a great-circle distance $\leq 90°$ with respect to the hovering condition. This closeness between the hovering condition and the singularity can limit the set of possible accelerations in aggressive flights, since an optimal solution that passes through or close to this singularity can provoke numerical instabilities and/or lead to artificial large changes in orientation. Recently, the

---

[1]Usually referred to as *yaw*, *heading*, or simply ψ.

Hopf map was leveraged in [36] to place the singularity in the inverted ("upside-down") configuration, which is independent of ψ and has the farthest possible angle away from the hovering condition. Although flying highly aggressive trajectories is not the main goal of this work, we decide to use the Hopf map (as opposed to the commonly-used maps presented in [33], [37]) since it automatically maximizes the distance to the singularity by simply changing the definition of the map. In [36], however, the Hopf fibration was only used in the controller to track predefined trajectories. It was also leveraged in [38] to find the set of charts for a previously-optimized position trajectory, which are then used for the controller and to obtain the ψ trajectory. In this work, we propose instead to embed the Hopf fibration in the joint (translation *and* yaw) coupled planning optimization as a way to directly obtain trajectories in SE(3) that, by construction, satisfy the underactuated dynamics of the UAV.

From the point of view of the ⌗goal⌗ of the perception awareness, most of the related works focus on **reducing the state estimation uncertainty** ⓑ.1, usually by keeping specific features/landmarks in the FOV, and/or choosing high-textured areas [13]–[16], [18]–[27]. These features are typically static in the world frame. Some of these approaches also leverage the Observability Gramian [25]–[27], especially when trying to ease the estimation of an unknown parameter of the dynamical system.

Further relevant work addresses the problem of having a UAV **record or chase a target** ⓑ.2 [17], [24], [28]–[32]. For example, [28] focused on tracking a moving target with a downward-facing camera, while [17] proposed a way to follow a moving target while avoiding other static obstacles in the environment. Most of these works focus therefore on chasing a static or dynamic target, not on avoiding it.

Our work differs from these two previous approaches because it proposes the use of PA planning to enhance the **avoidance of dynamic obstacles** ⓑ.3. Compared to ⓑ.1 or ⓑ.2, PA planning to avoid unknown dynamic obstacles comes with many additional challenges, such as the coupling of *both* the ego-motion and the motion of the obstacle in the visibility cost and blur of the image, the inclusion of dynamic obstacle avoidance constraints in the optimization, the need to predict the future trajectories of the obstacles, and the consideration of the uncertainty of these predicted trajectories, just to name a few.

In summary, the proposed contributions of this work are as follows:

- Real-time PA planning formulation that jointly optimizes the translation and the full rotation to maximize the visibility of unknown dynamic obstacles, while simultaneously avoiding them. Compared to non-PA approaches and PA decoupled approaches, our proposed coupled solution leads to a presence of the obstacle in the FOV 7.9 and 1.5 times more frequent, respectively.

TABLE 2. Notation used in this paper.

Symbol	Meaning
$\text{abs}(\boldsymbol{a})$, $\boldsymbol{a} \leq \boldsymbol{b}$	Element-wise absolute value, element-wise inequality.
$\|\cdot\|$	Euclidean norm.
c_α, s_α	$\cos(\alpha)$, $\sin(\alpha)$
$\circ$	Quaternion multiplication.
g	$g \approx 9.81$ m/s^2
$\sigma(\cdot)$	Sigmoid function [39].
$\boldsymbol{e}_z$, $\mathbf{1}$	$\boldsymbol{e}_z := \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$, $\mathbf{1} := \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^T$
FOV, AABB	Field of View, Axis-Aligned Bounding Box.
SO(n), SE(n)	Special orthogonal group, Special Euclidean group.
S^n	n-sphere.
$\text{wrap}^\pi_{-\pi}(\cdot)$	Wrapping of an angle in $[-\pi, \pi)$
$N(\cdot)$	Normal distribution.
$\text{norminv}(\cdot)$	Inverse of the standard normal cumulative distribution function [40].
$\text{diag}(\cdot)$	Diagonal matrix.
$\mathcal{S}^d_{p,m}$	Set of clamped uniform splines with dimension d, degree p, and $m+1$ knots.
n ($n_{\mathbf{p}}$ and n_ψ)	$n := m - p - 1$ $n + 1$ is the number of control points of the spline.
$\mathbf{p}, \mathbf{v}, \mathbf{a}, \mathbf{j}$	Position, Velocity, Acceleration, and Jerk of the UAV, $\in \mathbb{R}^3$. All of them are of the body w.r.t. the world frame, and expressed in the world frame.
$\boldsymbol{\xi}$	Relative acceleration, expressed in the world frame: $\boldsymbol{\xi} := \begin{bmatrix} a_x & a_y & a_z + g \end{bmatrix}^T$. We will assume $\boldsymbol{\xi} \neq \mathbf{0}$.
$\psi, \dot{\psi}$	Angle (and its derivative) such that $\boldsymbol{q}^w_b = \boldsymbol{q}_{\boldsymbol{\xi}} \circ \begin{bmatrix} c_{\psi/2} & 0 & 0 & s_{\psi/2} \end{bmatrix}^T$ (see section II-D1).
$\mathbf{x}$	State vector: $\mathbf{x} := \begin{bmatrix} \mathbf{p}^T & \mathbf{v}^T & \mathbf{a}^T & \psi & \dot{\psi} \end{bmatrix}^T \in \mathbb{R}^{11}$.
$\boldsymbol{p}^a$	Point expressed in the frame a. For the definitions of this table that include the sentence *"expressed in the world frame"*, the notation of the frame is omitted.
$\tilde{\boldsymbol{p}}, \bar{\boldsymbol{p}}$	$\tilde{\boldsymbol{p}} := \begin{bmatrix} \boldsymbol{p}^T & 1 \end{bmatrix}^T$, $\bar{\boldsymbol{p}} := \frac{\boldsymbol{p}}{\|\boldsymbol{p}\|}$
$\boldsymbol{T}^a_b = \begin{bmatrix} \boldsymbol{R}^a_b & \boldsymbol{t}^a_b \\ \boldsymbol{0}^T & 1 \end{bmatrix}$	Transformation matrix: $\tilde{\boldsymbol{p}}^a = \boldsymbol{T}^a_b \tilde{\boldsymbol{p}}^b$. Analogous definition for the quaternion $\boldsymbol{q}^a_b$.
$\text{rot}(\boldsymbol{q})$	Rotation matrix associated with the quaternion $\boldsymbol{q}$.
J	Set of indexes of all the intervals $J = \{0, 1, ..., m - 2p - 1\}$.
j	Index of the interval of the trajectory, $j \in J$.
I	Set of indexes of the tracked obstacles.
i	Index of the obstacle, $i \in I$.
i^*	Index of the obstacle used in the PA term of the cost function.
$(\mathbf{p}_i)^a(t)$	Mean of the predicted position of obstacle i, expressed in frame a.
$(\mathbf{p}_i)^w(t), \boldsymbol{\sigma}_i(t)$	The predicted trajectory of the obstacle i, in the world frame, is $\sim N\left((\mathbf{p}_i)^w(t), (\text{diag}(\boldsymbol{\sigma}_i(t)))^2\right)$.
f	Focal length of the camera in meters.
θ	Opening angle of the cone that approximates the FOV.
$\begin{bmatrix} q_w & q_x & q_y & q_z \end{bmatrix}^T$	Components of a unit quaternion.
$\mathbf{1}_c$	1 if c is true, 0 otherwise.
$\text{inFOV}(\boldsymbol{T}^w_c, (\mathbf{p}_i)^w)$	$\mathbf{1}_{(\mathbf{p}_i)^w \in \text{FOV}} \approx \mathbf{1}_{((\mathbf{p}_i)^c)_z / \|(\mathbf{p}_i)^c\| \geq c_{\theta/2}} \approx \sigma\left(\gamma\left(-c_{\theta/2} + ((\mathbf{p}_i)^c)_z / \|(\mathbf{p}_i)^c\|\right)\right)$. γ is a positive parameter.
$L_{\mathbf{p}}, L_\psi$	$L_{\mathbf{p}} := \{0, 1, ..., n_{\mathbf{p}}\}$, $L_\psi := \{0, 1, ..., n_\psi\}$.
l	Index of the control point. $l \in L_{\mathbf{p}}$ for $\mathbf{p}(t)$, $l \in L_{\mathbf{p}} \setminus \{n_{\mathbf{p}}\}$ for $\mathbf{v}(t)$, $l \in L_{\mathbf{p}} \setminus \{n_{\mathbf{p}} - 1, n_{\mathbf{p}}\}$ for $\mathbf{a}(t)$, $l \in L_\psi$ for ψ and $l \in L_\psi \setminus \{n_\psi\}$ for $\dot{\psi}$.
$\boldsymbol{q}_l, \boldsymbol{v}_l, \boldsymbol{a}_l, \psi_l, \Psi_l$	Position, velocity, and acceleration control points, ($\in \mathbb{R}^3$), ψ and $\dot{\psi}$ control points ($\in \mathbb{R}$).
$\mathcal{Q}^{\text{MV}}_j$	Set of position control points of the interval j using the MINVO basis. Analogous definition for the velocity control points $\mathcal{V}^{\text{MV}}_j$.
δ	$\in [0, 1]$, percentile of the standard normal distribution (see next row).
$\mathcal{C}^{\text{MV}}_{ij}$	Set of vertexes of the convex hull of the set obtained by inflating $(\mathcal{Q}^{\text{MV}}_j)_{\text{obs } i}$ with $\text{norminv}(\delta) \cdot \boldsymbol{\sigma}_i(t_{\text{end } j})$, half of the sides of the AABB of the obstacle i and half of the sides of the AABB of the agent.
$\boldsymbol{\pi}_{ij}(\boldsymbol{n}_{ij}, d_{ij})$	Plane $\boldsymbol{n}^T_{ij}\boldsymbol{x} + d_{ij} = 0$ that separates $(\mathcal{Q}^{\text{MV}}_j)_{\text{agent}}$ from $\mathcal{C}^{\text{MV}}_{ij}$.
$\boldsymbol{h}(\cdot), \boldsymbol{st}(\cdot)$	Hopf fibration, stereographic projection.
Snapshot at $t = t_1$ (current time):	

$\boldsymbol{g}_{\text{term}}$ (●) is the terminal goal, and ● is the current position of the UAV.
—— is the trajectory the UAV is currently executing.
- - - - is the trajectory the UAV is currently optimizing, $t \in [t_{\text{in}}, t_{\text{f}}]$.
$\boldsymbol{d}$ (○) is a point in ——, used as the initial position of - - - -
$\mathcal{M}$ is a sphere of radius r around $\boldsymbol{d}$.
$\boldsymbol{g}$ (●) is the projection of $\boldsymbol{g}_{\text{term}}$ (●) onto the sphere $\mathcal{M}$.
$\boldsymbol{d}$, $\boldsymbol{g}$, and $\boldsymbol{g}_{\text{term}}$ are expressed in the world frame.

The success rates achieved are on average 2.98 times larger than other state-of-the-art approaches when flying in multi-obstacle dynamic environments.

- We show how the Hopf fibration can be embedded in the planning optimization to jointly optimize translation and yaw while implicitly imposing the underactuated dynamics that couples acceleration and orientation. This avoids the need to explicitly impose the dynamics of the UAV as differential constraints, while automatically guaranteeing the largest possible great-circle distance between the hovering condition and the differential flatness singularity. Dynamic obstacle avoidance constraints are imposed by leveraging the MINVO basis to reduce conservatism.

- Extensive set of hardware experiments in unknown dynamic environments, with everything (navigation, perception, planning, and control) executed onboard the UAV, and without any prior knowledge of the trajectories or specific shape/size of the obstacles. The UAV achieves velocities of up to 5.8 m/s and relative velocities (with respect to the obstacles) of up to 6.3 m/s. The replanning times achieved onboard are $\approx$ 53 ms.

- The code has also been released open source for the community.

This paper uses the notation shown in Table 2.

II. PANTHER

PANTHER comprises four modules: Tracker and predictor, selector of the obstacle in the PA term, planes and initial guess generator, and optimization (see Fig. 2A). A summary of how all these modules work together is as follows: First the incoming point clouds of the onboard depth sensor are clustered and tracked using the Hungarian algorithm [41] to obtain the trajectory, as a probability distribution, of each of the obstacles (section II-A). The obstacle i^* that the UAV is most likely to collide with is then selected to be included in the PA term of the cost function (section II-B). Then, a kinodynamic search-based planner (Octopus Search Algorithm [42]) is run to find an initial guess of the translational trajectory $\mathbf{p}(t)$ that avoids the probabilistic trajectories of the obstacles found before (section II-C1). This translational guess and the obstacle i^* selected are then used to run a graph search algorithm to find the $\psi(t)$ guess (section II-C2). Finally, the $\mathbf{p}(t)$ and $\psi(t)$ guesses are used for the nonconvex optimization to obtain the optimized trajectory, that is sent to the controller of the UAV (section II-D). In this framework, the coupling between rotation and acceleration is imposed implicitly using the Hopf fibration. All these modules are described in detail in the following subsections.

A. TRACKING AND PREDICTION

We create a k-d tree representation of the point clouds coming from the onboard depth sensor, and perform Euclidean clustering to group the points that are more likely to belong

to the same obstacle (see Fig. 2). For each cluster found, we compute the AABB (Axis-Aligned Bounding Box) centered on the centroid of that cluster[2]. Then, to assign each cluster to a specific track, we minimize the total assignment cost using the Hungarian algorithm [41], where the cost is the pairwise distance between the centroid of each cluster and the prediction of the tracks at the time the point cloud was produced. If this distance is above a specific threshold (usually $\approx$ 1–2 m), we create a new track for it. If a cluster is not assigned to any track (which can happen if there are more clusters than tracks), then a new track is created for it. Finally, given a sliding window history of all the observations associated with a track, we fit a polynomial for each coordinate $\{x, y, z\}$. To capture the stochasticity of the prediction problem, the predicted position at time t is then approximated by a 3D Gaussian distribution (mean from the value of the fitted polynomial and a diagonal covariance matrix obtained from the prediction intervals [44, section 5.7]).

B. SELECTION OF THE OBSTACLE IN THE PA TERM

When there are several predicted trajectories, and to maintain computational tractability, the agent needs to choose which one of them to include in the PA term of the cost function. It does so by choosing the most likely obstacle to collide with in the future, using a simple heuristic of the probability of collision based on Boole's inequality [45]:

$$i^* = \underset{i \in I}{\operatorname{argmax}} \sum_{u=0}^{U-1} P\left(\left\|(\mathbf{p}_i)^w(t_u) - \boldsymbol{\kappa}(u)\right\|_\infty \leq R\right)$$

where U is the number of samples taken, $t_u := t_{\text{in}} + \frac{u}{U}(t_f - t_{\text{in}})$ and $\boldsymbol{\kappa}(u) := \boldsymbol{d} + \frac{u}{U}(\boldsymbol{g}_{\text{term}} - \boldsymbol{d})$ is a point in a straight line from $\boldsymbol{d}$ to $\boldsymbol{g}_{\text{term}}$. Note that although only one obstacle is included in the PA objective function, all the predictions of the tracked obstacles are included in the collision avoidance constraints.

Additionally, and to address the trade-off between gathering information about the obstacle, and gathering information about the direction of travel, the UAV will include the obstacle i^* in the PA term if the angle between $(\boldsymbol{g}_{\text{term}} - \boldsymbol{d})$ and $((\boldsymbol{p}_{i^*})^w(t_{\text{in}}) - \boldsymbol{d})$ is smaller than a predefined angle α_0 (typically $\approx$ 90°). Otherwise the UAV will try to align the FOV of the camera with the direction of travel.

C. PLANES AND INITIAL GUESSES

1) SEPARABILITY PLANES AND INITIAL GUESS FOR POSITION

We use the Octopus Search Algorithm (OSA) [42], which is a search-based algorithm that operates directly on the control points of the position spline. It ensures collision-free constraints between the agent and the dynamic obstacles by

---

[2]Regardless of whether or not the obstacle is convex, this produces an outer convex approximation of the visible part of the obstacle.
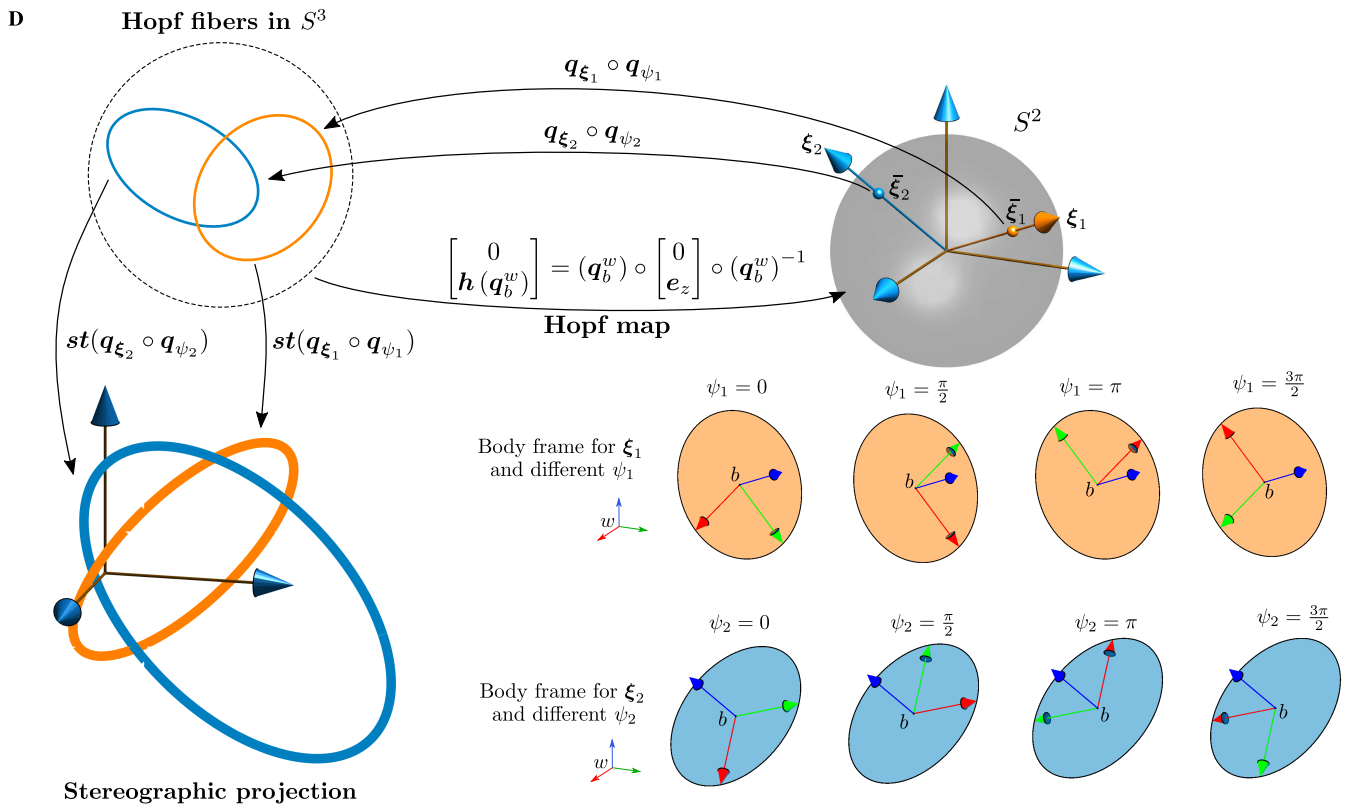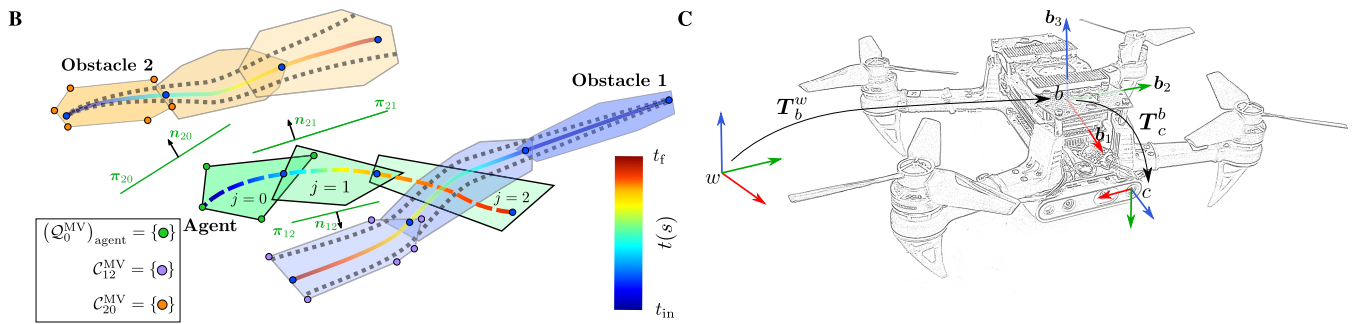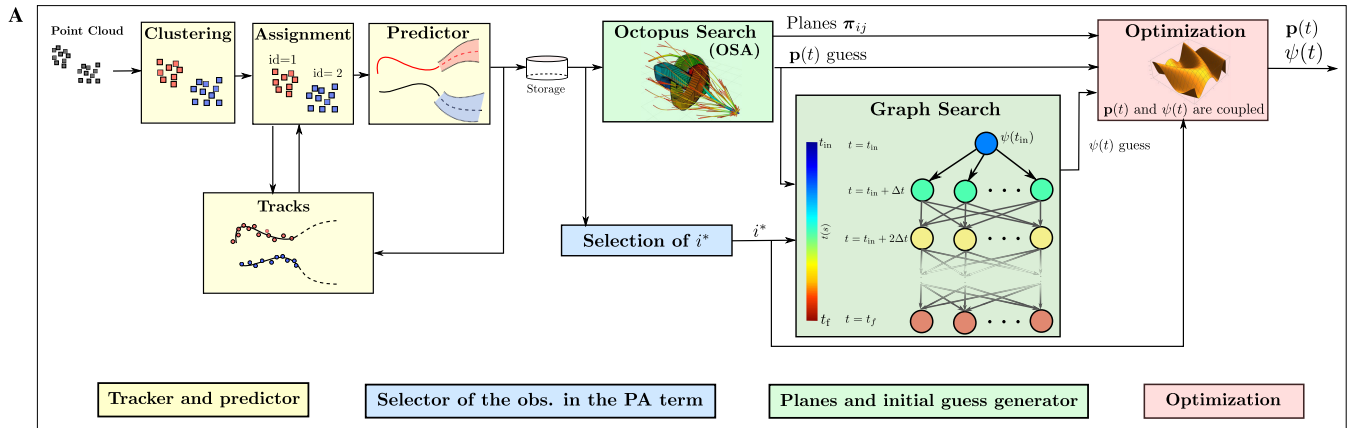
FIGURE 2. **(A)** Different modules of PANTHER. **(B)** Predicted trajectories of the obstacles and convex representation of each segment of the trajectory of the agent and the obstacles. **(C)** World, body, and camera frames. **(D)** Hopf fibration and its stereographic projection, partly inspired from [43]. Given a specific relative acceleration ξ (with $\bar{\xi} \neq -e_z$), the quaternion $q_b^w = q_\xi \circ q_\psi$ is a fiber (specifically a circle) in S^3 parameterized by ψ. On the bottom right, the body frames for different values of ψ_i for each ξ_i are shown.

finding the planes that separate the inflated MINVO polyhedral representation of each interval j of the trajectory of the obstacle i (denoted as $\mathcal{C}_{ij}^{\mathrm{MV}}$) and the MINVO polyhedral representation of that interval j of the trajectory of the agent, denoted as $\left(\mathcal{Q}_j^{\mathrm{MV}}\right)_{\text{agent}}$ (see Fig. 2B). The outputs of this algorithm are both the position control points and the planes $\boldsymbol{\pi}_{ij}$ (given by $\boldsymbol{n}_{ij}^T \boldsymbol{x} + d_{ij} = 0$) $\forall i, \forall j$. The position control points are then used as initial guess in the optimization, while the planes $\boldsymbol{\pi}_{ij}$ are held fixed in the optimization. The reader is referred to our previous work [42] for a more in-depth explanation of the OSA.

2) INITIAL GUESS FOR ψ

To obtain the initial guess for ψ, we uniformly sample the position guess spline obtained through the OSA, and for each of these position samples, we uniformly sample several values of $\psi \in [-\pi, \pi)$. Each one of these $\mathbf{p}$-ψ samples will be a *node*, and all the nodes associated with the same position sample, but with different ψ, will constitute a *layer* (see Fig. 2A). Then, we create a graph connecting with directed edges all the nodes of one layer to the nodes of the next layer [12]. Each node has therefore a time, position, acceleration, and yaw associated with it, and all the nodes of the same layer have the same time, position, and acceleration. The cost of the edge between two nodes n_1 and n_2 of the graph is then given by

$$c_\psi \left(\mathrm{wrap}_{-\pi}^{\pi}\left(\psi_{n_2} - \psi_{n_1}\right)\right)^2 + c_{\Psi_{\max}} \cdot \mathbf{1}_{\left|\frac{\mathrm{wrap}_{-\pi}^{\pi}\left(\psi_{n_2} - \psi_{n_1}\right)}{t_{n_2} - t_{n_1}}\right| > \Psi_{\max}}$$
$$+ c_{\mathrm{FOV}}\left(1 - \mathrm{inFOV}\left(\left(\boldsymbol{T}_{n_2}\left(t_{n_2}\right)\right)_c^w, \left(\boldsymbol{p}_{i*}\left(t_{n_2}\right)\right)^w\right)\right)$$

Here, c_ψ, $c_{\Psi_{\max}}$, and c_{FOV} are nonnegative weights, while ψ_{n_u}, t_{n_u}, and $\left(\boldsymbol{T}_{n_u}\left(t_{n_u}\right)\right)_c^w$ are the angle ψ, the time, and the transformation matrix associated with node n_u. Note that the edge cost is guaranteed to be nonnegative at all times. The transformation matrix can be directly obtained from the position, acceleration, and yaw of the node. The first term in the cost penalizes the distance between two ψ angles, the second term penalizes edges that do not satisfy the limit $\Psi_{\max}$, and the last one rewards the visibility of the obstacle. The units of the weights above are such that the corresponding term is dimensionless (see section III). To choose these weight values, we first set $c_{\Psi_{\max}}$ to a large value to guarantee the $\Psi_{\max}$ constraint. Then, c_ψ and c_{FOV} are selected as a trade-off between smoothness and inclusion of the obstacle $i*$ in the FOV of the UAV. The root node of the graph corresponds to the state $\boldsymbol{d}$ (see last row of Table 2). We solve the search problem using Dijkstra's algorithm [46], with early termination when the search reaches a node of the last layer. Letting Λ denote the indexes of the nodes of the path found, we shift the angles $\psi_{n_\lambda} \; \forall\lambda \in \Lambda$ (by adding or subtracting $2\pi r$, $r \in \mathbb{Z}$) such that the absolute difference between two consecutive angles is $\leq \pi$. Using $\hat{\psi}_{n_\lambda}$ to denote these shifted angles, a spline is fitted to these angles by solving the following

constrained least square problem:

$$\min_{\psi(t) \in \mathcal{S}_{2,m}^1} \sum_{\lambda \in \Lambda} \left\| \psi\left(t_{n_\lambda}\right) - \hat{\psi}_{n_\lambda} \right\|_2^2$$
$$\text{s.t. } \psi(t_{\mathrm{in}}) = \psi_{\mathrm{in}}, \quad \dot{\psi}(t_{\mathrm{in}}) = \dot{\psi}_{\mathrm{in}}, \quad \dot{\psi}(t_{\mathrm{f}}) = 0 \quad (1)$$

Note that, as this problem is a quadratic program with linear equality constraints, its solution can be easily found by simply solving the linear Karush-Kuhn-Tucker (KKT) conditions associated with it [47], [48].[3] The control points of this fitted spline are then used as the initial guess for $\psi(t)$ in the optimization.

D. OPTIMIZATION
1) COUPLING ROTATION AND ACCELERATION WITH THE HOPF FIBRATION

In a standard multirotor-UAV, the perpendicularity of the total thrust with respect to the plane spanned by $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ (see the coordinate frames shown in Fig. 2C) makes the UAV underactuated by imposing the following constraint [36]:

$$\mathrm{rot}\left(\boldsymbol{q}_b^w\right)\boldsymbol{e}_z = \bar{\boldsymbol{\xi}} \quad (2)$$

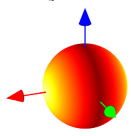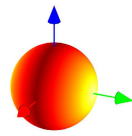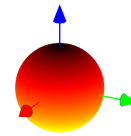where $\bar{\boldsymbol{\xi}}$ is the normalized relative acceleration expressed in the world frame (see Table 2). In a planning optimization problem where rotation and translation are jointly optimized, (2) needs to be satisfied at all times. A very common way to guarantee (2) is via direct imposition of the dynamic equations of the UAV as explicit constraints. However, these differential equations in the optimization problem typically lead to computationally-expensive problems, due to the fine sampling needed in the discretization methods (shooting or collocation).

The direct imposition of the dynamic equations can be avoided by leveraging the differential flatness map ($\mathbf{a} \in \mathbb{R}^3 \setminus [0 \; 0 \; -g]^T$, $\psi \in S^1) \rightarrow \mathbf{R}_b^w \in SO(3)$, which takes the acceleration $\mathbf{a}$ and ψ and maps them to the rotation of the body frame. Due to the *hedgehog theorem*[4] in S^2 [34], [35], this map is guaranteed to have at least one singularity when tried to be defined with a single continuous function. Several possible definitions of this differential flatness map are shown in Table 3, all of which satisfy (2) by construction. In the first two definitions, one body axis is obtained as the cross product of $\boldsymbol{b}_3 \equiv \bar{\boldsymbol{\xi}}$ with a vector lying in the xy world plane, and the remaining body axis is such that the resulting body frame is right-handed. These two definitions present a singularity whenever the normalized relative acceleration $\bar{\boldsymbol{\xi}} \in S^2$ is parallel to a vector defined by ψ which lies in the xy world plane. This means that, for a given ψ, the singularity appears for two $\bar{\boldsymbol{\xi}}$ that have a great-circle distance of 90° with respect to the hovering condition. In aggressive flights, and due to numerical instabilities and artificial large changes of orientations near the singularity, this closeness between the

---

[3]For a detailed explanation of the derivation of the resulting linear system of equations, see, e.g., [49, Example 5.1].

[4]Also known as the *hairy ball theorem* in the literature.

TABLE 3. Some commonly-used definitions for the differential flatness map ($a \in \mathbb{R}^3 \setminus [0 \ 0 \ -g]^T$, $\psi \in S^1$) $\rightarrow R_b^w \in SO(3)$. The colormap represents the great-circle distance to the closest singularity (yellow is closer), $(\cdot)_n$ denotes the normalization of a vector, and $\bar{\xi} := ([a_x \ a_y \ a_z + g]^T)_n$ is the normalized relative acceleration, expressed in the world frame. See also [13], [38], [50]–[52] for more possible definitions, which are usually rotations of the first two definitions of this table.

	Definition 1	**Definition 2**	**Definition 3 (Hopf fibration)**
Map	$b_1 = b_2 \times b_3$ $b_2 = \left(b_3 \times \begin{bmatrix} c_\psi \ s_\psi \ 0 \end{bmatrix}^T\right)_n$ $b_3 = \bar{\xi}$ $R_b^w = \begin{bmatrix} b_1 \ b_2 \ b_3 \end{bmatrix}$	$b_1 = \left(\begin{bmatrix} -s_\psi \ c_\psi \ 0 \end{bmatrix}^T \times b_3\right)_n$ $b_2 = b_3 \times b_1$ $b_3 = \bar{\xi}$ $R_b^w = \begin{bmatrix} b_1 \ b_2 \ b_3 \end{bmatrix}$	$q_b^w = \underbrace{\dfrac{1}{\sqrt{2(1+\bar{\xi}_z)}} \begin{bmatrix} 1+\bar{\xi}_z \\ -\bar{\xi}_y \\ \bar{\xi}_x \\ 0 \end{bmatrix}}_{:=q_\xi} \circ \underbrace{\begin{bmatrix} c_{\psi/2} \\ 0 \\ 0 \\ s_{\psi/2} \end{bmatrix}}_{:=q_\psi}$ $R_b^w = \mathrm{rot}\left(q_b^w\right)$
Singularity	$\bar{\xi} \parallel \begin{bmatrix} c_\psi \ s_\psi \ 0 \end{bmatrix}^T$. When $\psi = 0$: 	$\bar{\xi} \parallel \begin{bmatrix} -s_\psi \ c_\psi \ 0 \end{bmatrix}^T$. When $\psi = 0$: 	$\bar{\xi} = \begin{bmatrix} 0 \ 0 \ -1 \end{bmatrix}^T$
Notes	• Singularity=$f(\mathbf{a}, \psi)$ • For a given ψ, singularity appears for two $\bar{\xi}$ • UAV is differentially flat, with flat outputs $\{\mathbf{p}, \psi\}$ [33]	• Singularity=$f(\mathbf{a}, \psi)$ • For a given ψ, singularity appears for two $\bar{\xi}$ • UAV is differentially flat, with flat outputs $\{\mathbf{p}, \psi\}$ [37]	• Singularity=$f(\mathbf{a})$ • Singularity appears for one $\bar{\xi}$ • UAV is differentially flat, with flat outputs $\{\mathbf{p}, \psi\}$ [36]

hovering condition and the singularity can limit the set of possible accelerations for the planner.

The third definition of Table 3 leverages the Hopf fibration $h(\cdot)$, which can be defined as a map $S^3 \rightarrow S^2$ [36], [43] that takes a unit quaternion q and produces the resulting rotation of the vector $e_z := \begin{bmatrix} 0 \ 0 \ 1 \end{bmatrix}^T$ (see Fig. 2D):

$$\begin{bmatrix} 0 \\ h(q) \end{bmatrix} := q \circ \begin{bmatrix} 0 \\ e_z \end{bmatrix} \circ q^{-1}$$

Making use now of the inverse image of the Hopf fibration, we have that q_b^w will be a composition of two rotations[5]: q_ξ, that aligns b_3 with ξ, followed by q_ψ, which is a rotation around ξ by an angle ψ. Given a specific ξ (with $\bar{\xi} \neq -e_z$), the quaternion $q_b^w = q_\xi \circ q_\psi$ will then be a fiber (specifically a circle) in S^3 parametrized by ψ [43]. The main advantage of the Hopf fibration over the previous two definitions is that the singularity only occurs when the UAV is inverted (i.e., when $\bar{\xi} = -e_z$), which is the orientation that has the largest possible great-circle distance from the hovering configuration, and hence much less likely to happen. Although the goal of this paper is not to plan highly aggressive trajectories, and hence any of the singularities shown in Table 3 are unlikely to be reached, we use the Hopf map to automatically ensure the maximum distance to the singularity. Note also that, with the Hopf fibration, a second chart could be used to cover the inversion point $\bar{\xi} = -e_z$, but the use of multiple charts,

[5]Note that q_ψ, q_ξ, and q_b^w are guaranteed to be unit quaternions by construction.

while computationally cheap in the controller level [36] or in an intermediate check step in a decoupled $\mathbf{p}$–ψ optimization [38], would significantly increase the computation time when embedded in the $\mathbf{p}$–ψ joint planning optimization. This fact, together with the improbability of an upside-down configuration as being PA optimal, led us to the inclusion of only the first chart.

Our work differs from other works that have used the Hopf fibration for UAVs [36], [38], [53] as follows: Ref. [36] uses the Hopf fibration only in a controller to track predefined trajectories. In [38], the Hopf fibration is used in the planner to find the charts in a step after the $\mathbf{p}$ optimization and before the ψ optimization, and [53] does not optimize ψ. We instead propose to embed the Hopf fibration map directly on the $\mathbf{p}$–ψ *joint* optimization, as a way to directly obtain trajectories in SE(3) that are dynamically feasible by construction, and with the crucial advantage of not needing to explicitly impose the dynamic equations as constraints in the optimization.

2) COST FUNCTION

A PA term in the objective function should maximize the presence in the FOV of the predicted position of the obstacle $i^* \in I$. However, this alone is not enough to guarantee good PA trajectories, since a fast moving projected obstacle in the image plane may cause significant blur, which can lead to stereo matching failure and consequently tracking failure. To take into account both the presence in the FOV and the blur, we use $\frac{\mathrm{inFOV}(\cdot)}{\epsilon_1 + \epsilon_2 \|\dot{s}\|^2}$ as the running reward, where

$\dot{s}$ is the projected velocity in the image plane, and where ϵ_1 and ϵ_2 are nonnegative parameters such that $\epsilon_1 + \epsilon_2 > 0$. Note how this reward is high if the predicted position of the obstacle is in the FOV with a small projected velocity, and it is approximately zero if the predicted position of the obstacle is not in the FOV, regardless of the value of the projected velocity. The position in the image plane of the projection of the obstacle can be obtained using the pinhole camera model as $s(t) := \frac{f}{\left[(\tilde{\mathbf{p}}_{i*}(t))^c\right]_z} \left[\left(\tilde{\mathbf{p}}_{i*}(t)\right)^c\right]_{x:y}$ (where each component of s is expressed in meters, not in pixels), and

$$\left(\tilde{\mathbf{p}}_{i*}(t)\right)^c := \mathbf{T}_b^c \mathbf{T}_w^b(t) \left(\tilde{\mathbf{p}}_{i*}(t)\right)^w$$

$$\mathbf{T}_w^b(t) := \begin{bmatrix} \mathrm{rot}\big(\mathbf{q}_{\xi}(t) \circ \mathbf{q}_{\psi}(t)\big) & \mathbf{p}(t) \\ \mathbf{0}^T & 1 \end{bmatrix}^{-1}$$

As detailed in Table 2, the discontinuity of the function inFOV($\cdot$) is addressed by approximating it with a sigmoid function.

In addition to the PA term explained above, we also add two terms in the cost function to maximize the smoothness in position (by minimizing jerk) and ψ (by minimizing $\ddot{\psi}$), and a terminal cost that penalizes the distance between $\mathbf{p}(t_f)$ and $\mathbf{g}$.

3) COLLISION AVOIDANCE AND DYNAMIC LIMITS CONSTRAINTS

For the obstacle avoidance of dynamic obstacles, we first create a polyhedral outer representation of both the trajectory of the agent and of the obstacle (see Fig. 2B): For the agent, we make use of the MINVO basis [54] (a polynomial basis that finds the simplex with minimum volume enclosing a polynomial curve) to obtain the set of control points $\left(\mathcal{Q}_j^{\mathrm{MV}}\right)_{\mathrm{agent}}$ whose convex hull encloses each segment j of the agent. Similarly, for each obstacle i, we first compute the MINVO control points of the segment j of the predicted mean $(\mathbf{p}_i)^w(t)$, and then we inflate it with norminv(δ) $\cdot \boldsymbol{\sigma}_i(t_{\mathrm{end}\,j})$, half of the sides the AABB (axis-aligned bounding box) of the obstacle i and half of the sides of the AABB of the agent. Here, $\delta \in [0, 1]$ is the percentile of the standard normal distribution, and hence it encodes the desired level of conservativeness in the inflation. The resulting polyhedron is denoted as $\mathcal{C}_{ij}^{\mathrm{MV}}$.

To ensure safety between the agent and the obstacle i, we then impose linear separability constraints (via planes) between $\left(\mathcal{Q}_j^{\mathrm{MV}}\right)_{\mathrm{agent}}$ and $\mathcal{C}_{ij}^{\mathrm{MV}}$. The separating planes are found during the initial guess search for the position spline (see section II-C1), and are held fixed in the optimization. The MINVO basis is used in a similar way to impose low-conservative constraints in the velocity space. In the acceleration and jerk spaces, the MINVO control points are the same as the B-Spline control points. These constraints on $\mathbf{v}$, $\mathbf{a}$, $\mathbf{j}$, and $\dot{\psi}$ serve as a conservative approximation of the real actuator constraints of the motors of the UAV, while allowing us to reduce the complexity of the optimization problem.

4) OPTIMIZATION PROBLEM

Including the initial state and the final hovering condition, the optimization problem is[6]:

$$
\begin{aligned}
\min_{\mathbf{p}(t)\in\mathcal{S}_{3,m}^3,\,\psi(t)\in\mathcal{S}_{2,m}^1} \quad & \alpha_{\mathbf{j}} \int_{t_{\mathrm{in}}}^{t_f} \|\mathbf{j}\|^2\, dt + \alpha_{\psi} \int_{t_{\mathrm{in}}}^{t_f} \left(\dddot{\psi}\right)^2 dt \\
& - \alpha_{\mathrm{FOV}} \int_{t_{\mathrm{in}}}^{t_f} \frac{\mathrm{inFOV}(T_c^w, (\mathbf{p}_{i*})^w)}{\epsilon_1 + \epsilon_2 \|\dot{s}\|^2} dt \\
& + \alpha_{\mathbf{g}} \left\|\mathbf{p}(t_f) - \mathbf{g}\right\|^2
\end{aligned}
$$

$$
\begin{aligned}
\text{s.t.} \quad & \mathbf{x}(t_{\mathrm{in}}) = \mathbf{x}_{\mathrm{in}},\ \ \mathbf{v}(t_f) = \mathbf{0},\ \ \mathbf{a}(t_f) = \mathbf{0},\ \ \dot{\psi}(t_f) = 0 \\
& \mathbf{n}_{ij}^T \mathbf{q} + d_{ij} < 0 \quad \forall \mathbf{q} \in \left(\mathcal{Q}_j^{\mathrm{MV}}\right)_{\mathrm{agent}}, \forall i \in I,\ \ \forall j \in J \\
& \mathrm{abs}\,(\mathbf{v}) \leq \mathbf{v}_{\max} \quad \forall \mathbf{v} \in \left(\mathcal{V}_j^{\mathrm{MV}}\right)_{\mathrm{agent}},\ \ \forall j \in J \\
& \mathrm{abs}\,(\mathbf{a}_l) \leq \mathbf{a}_{\max} \quad \forall l \in L_{\mathbf{p}}\backslash\{n_{\mathbf{p}}-1, n_{\mathbf{p}}\} \\
& \mathrm{abs}\,(\mathbf{j}_l) \leq \mathbf{j}_{\max} \quad \forall l \in L_{\mathbf{p}}\backslash\{n_{\mathbf{p}}-2, n_{\mathbf{p}}-1, n_{\mathbf{p}}\} \\
& \mathrm{abs}\,(\Psi_l) \leq \Psi_{\max} \quad \forall l \in L_{\psi}\backslash\{n_{\psi}\}
\end{aligned}
$$

Here, $\mathbf{x} := \begin{bmatrix} \mathbf{p}^T & \mathbf{v}^T & \mathbf{a}^T & \psi & \dot{\psi} \end{bmatrix}^T$, $\{\alpha_{\mathbf{j}},\ \alpha_{\psi},\ \alpha_{\mathrm{FOV}},\ \alpha_{\mathbf{g}}\}$ are nonnegative weights, and the decision variables are the control points of the splines $\mathbf{p}(t)$ and $\psi(t)$. The degrees chosen for the splines $\mathbf{p}(t)$ and $\psi(t)$ are, respectively, 3 and 2, which are a good trade-off between computation time and dynamic feasibility for a UAV [33]. The units of the weights are such that the corresponding term is dimensionless (see section III). An empirical method to select these weight values is as follows: First set $\alpha_{\mathbf{g}}$ to a large value to ensure that the final location is near $\mathbf{g}$. Then, α_{FOV}, together with ϵ_1 and ϵ_2, are tuned to obtain a good presence of the obstacle i^* in the FOV. Finally, $\alpha_{\mathbf{j}}$ and α_{ψ} are progressively increased to improve the smoothness of $\mathbf{p}(t)$ and $\psi(t)$, without significantly deteriorating the FOV cost.

To solve this optimization problem, we utilize the Interior Point Optimizer Ipopt [55][7] interfaced through CasADi [57] with MA27 and MA57 [58] as the linear solvers of Ipopt. All these optimization tools were installed and run onboard the UAV in the real-world experiments (section III-B). We approximate the PA term of the cost function using the composite Simpson's rule for numerical integration [59].

III. RESULTS AND DISCUSSION
A. SIMULATION EXPERIMENTS

All the simulation experiments are run in an `AlienWare Aurora r8` desktop running Ubuntu 20.04 and equipped

---

[6]Time dependence of the variables in the cost function has been omitted for simplicity.

[7]We classify an Ipopt solution as successful when Ipopt returns `Solve_Succeeded` (locally optimal solution) or `Solved_To_Acceptable_Level` (solution satisfying the acceptable tolerance level). For more details, see [56].
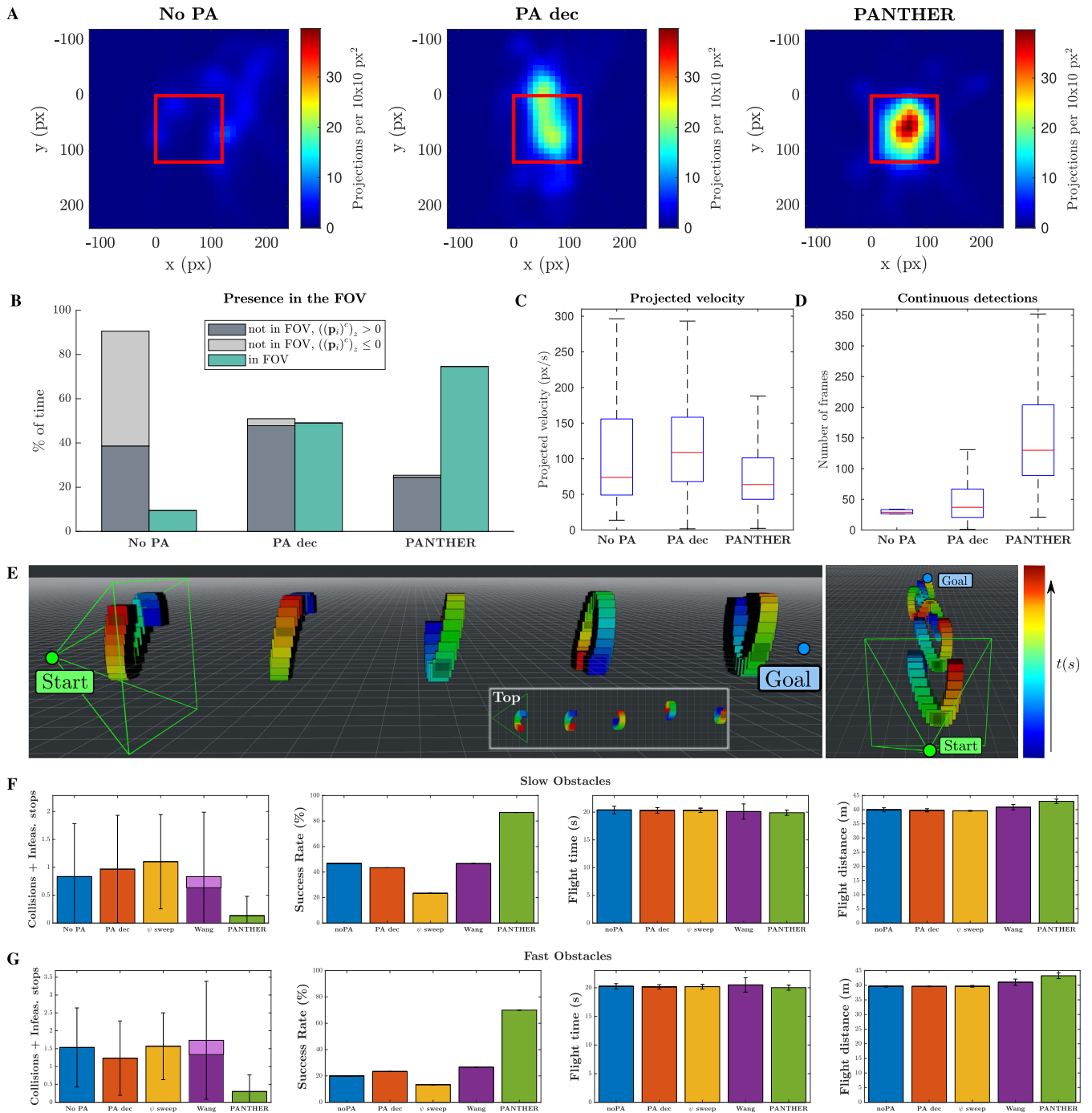
FIGURE 3. (A) Projections of the obstacle onto the image plane in the single-obstacle simulation experiments. The red square is the image plane, so any projection out of this region is not in the FOV of the camera. (B) Percentage of the time the obstacle was not in the FOV but in front of the camera (■), not in the FOV and behind the camera (■), and in the FOV (■). (C) Velocity of the projection of the centroid of the obstacle onto the image plane. Higher projected velocities produce larger blur in the image. (D) Number of frames for each continuous detection. (E) Corridor simulation with five dynamic obstacles following random trefoil-knot trajectories [60]. The green pyramid represents the FOV of the camera. (F, G) Results for the corridor simulations with slow and fast obstacles, respectively. The algorithms considered are *no PA* (■), *PA dec* (■), ψ *sweep* (■), *Wang* [6] (■), and PANTHER (■). In the left plot of both subfigures, ■ represents the number of infeasible stops of algorithm [6]. The other algorithms have zero infeasible stops.

with an Intel® Core™ i9-9900K CPU, 3.60GHz × 16 and 62.6 GiB. Moreover, and to focus the comparisons on the properties of the trajectories obtained by the planner, we assume, for all the algorithms benchmarked in simulation, that the UAV can perfectly track the trajectories obtained by the planner.

1) SINGLE OBSTACLE

We first test PANTHER in an environment with a box-shaped obstacle of size $0.2 \times 0.2 \times 0.2$ m^3 that follows a trefoil-knot [60] trajectory. During 60 s, the UAV is commanded to continuously fly between two different locations whose centroid is the area where the obstacle is moving. The camera has an image size of 120×120 px^2, a limited FOV of $60° \times 60°$, and runs at a rate of 60 Hz. The weights used for this simulation are $c_{\Psi_{max}} = 10^6$, $c_{FOV} = 1$, $c_\psi = 0$ rad^{-2}, $\alpha_j = 10^{-6}$ s^5/m^2, $\alpha_\psi = 0$ s^3/rad^2, $\alpha_{FOV} = 20$, $\alpha_g = 70$ m^{-2}, $\epsilon_1 = 0.3$, and $\epsilon_2 = 0.45$ s^2/m^2. To focus this comparison on the capabilities of the planner, we let the agent perfectly know the trajectory of the obstacle in these simulations. We compare the following three approaches:

1) **No PA:** ψ is held constant and only the smoothness in position and terminal goal costs are optimized. Works that do not plan ψ include, e.g., [1], [5], [7].
2) **PA with position and ψ decoupled**: Translation **p** is optimized first (as in the method *no PA*) and then it is held fixed while ψ is optimized with the PA term. We will refer to this algorithm as **PA dec**. This decoupling is done in, e.g., [12]–[14].
3) **PANTHER** (ours): Joint optimization of **p** and ψ.

As will be explained in section II-D2, two important metrics that characterize a good PA trajectory are the presence of the obstacle in the FOV and the norm of the projected velocity, which quantifies the blur. The percentage of time the obstacle was in the FOV of the camera is shown in Fig. 3B. PANTHER is able to keep the obstacle inside the FOV 7.9 and 1.5 times more than the algorithms *no PA* and *PA dec*, respectively. As *PA dec* decouples position and ψ in the optimization, the UAV lacks the ability to modify the spatial path (only ψ) to generate a better overall trajectory.

To qualitatively show the area of the projection, we apply a Gaussian filter to the histogram of the projection of the centroid of the obstacle onto each 10×10 px^2 cell of the image plane. The results are shown in Fig. 3A, where we can see that PANTHER is able to keep the obstacle inside the FOV limits much better, and more frequently, than methods *no PA* and *PA dec*.

The velocity of the projection of the centroid of the obstacle onto the image plane is shown in Fig. 3C, which highlights that PANTHER is able to obtain a 18% and 34% decrease in the mean of the norm of the projected velocity with respect to *no PA* and *PA dec*, respectively, achieving, therefore, a much less blurred projection of the obstacle than those two methods.

Finally, and as a continuous detection of the dynamic obstacle is crucial to achieve a good tracking and prediction, we show in Fig. 3D the boxplot of the number of frames of each continuous detection for the different algorithms. A continuous detection is defined as a set of consecutive frames for which the obstacle stayed in the FOV of the camera. On average, PANTHER is able to achieve continuous detections of 155 frames, while the mean number of frames

per continuous detection for methods *no PA* and *PA dec* are 39 and 46 frames, respectively.

2) SEVERAL OBSTACLES

We now test PANTHER in a simulation with several obstacles. The environment consists of a corridor of length of 39 m along the x direction with five dynamic obstacles that move following random trefoil-knot trajectories [60], see Fig. 3E. In all these simulations, the agent only has access to the size, current position, and velocity of the obstacles that are inside the FOV of the camera. The FOV of the camera is $70° \times 70°$, and has a sensing range of 5 m. The dynamic limits are $v_{max} = 2.6 \cdot \mathbf{1}$ m/s, $a_{max} = 15.5 \cdot \mathbf{1}$ m/s^2, $j_{max} = 50.0 \cdot \mathbf{1}$ m/s^3, and $\Psi_{max} = \pi$ rad/s. The weights used for PANTHER in these simulations are $c_{\Psi_{max}} = 10^6$, $c_{FOV} = 1$, $c_\psi = 0$ rad^{-2}, $\alpha_j = 10^{-7}$ s^5/m^2, $\alpha_\psi = 0$ s^3/rad^2, $\alpha_{FOV} = 40$, $\alpha_g = 25$ m^{-2}, $\epsilon_1 = 0.3$, and $\epsilon_2 = 10^{-5}$ s^2/m^2. The UAV is constrained to remain in $y \in [-4, 4]$ m and $z \in [-4, 4]$ m at all times.

For the benchmark, we use the algorithms explained before (*no PA*, *PA dec*, and PANTHER), and the two additional algorithms:

- Algorithm [6], proposed by Wang *et al.* This approach is not perception aware, but ψ tries to make the FOV of the camera point to the direction of travel. We will refer to this algorithm as **Wang**. Note also that this algorithm does not have constraints on j_{max} and that it has a different ψ convention (it uses definition 1 of Table 3).
- **ψ sweep**: ψ follows a sinusoidal trajectory that varies in $[-90°, 90°]$ as follows:

$$\psi(t) = \frac{\pi}{2} \sin\left(\frac{\Psi_{max}}{\pi/2} t\right)$$

We test two scenarios with different maximum velocities of the obstacles. In the slow scenario, the obstacles move with velocities up to 2.12 m/s, while in the fast scenario, the obstacles move with velocities up to 4.07 m/s. In the results, we compare the number of collisions, infeasible stops, success rate, flight time, and flight distance. An infeasible stop happens when the drone passes instantly from a nonstop condition ($\mathbf{v} \neq \mathbf{0}$ or $\mathbf{a} \neq \mathbf{0}$) to a stop condition ($\mathbf{v} = \mathbf{0}$ and $\mathbf{a} = \mathbf{0}$). A run is considered successful if the UAV is able to reach the end of the corridor while not colliding with any of the obstacles. To make these simulations closer to real-world applications, where no prior information about the trajectories of the obstacles may be available, a simple constant velocity model is used in the predictor. The obstacles themselves are moving along trefoil-knot trajectories [60].

The results, for 30 different runs per algorithm, are shown in Figs. 3F and 3G for the slow and fast environments, respectively. In the slow scenario, PANTHER is able to succeed 87% of the runs, while the other algorithms have a success rate below 47%. None of the algorithms present
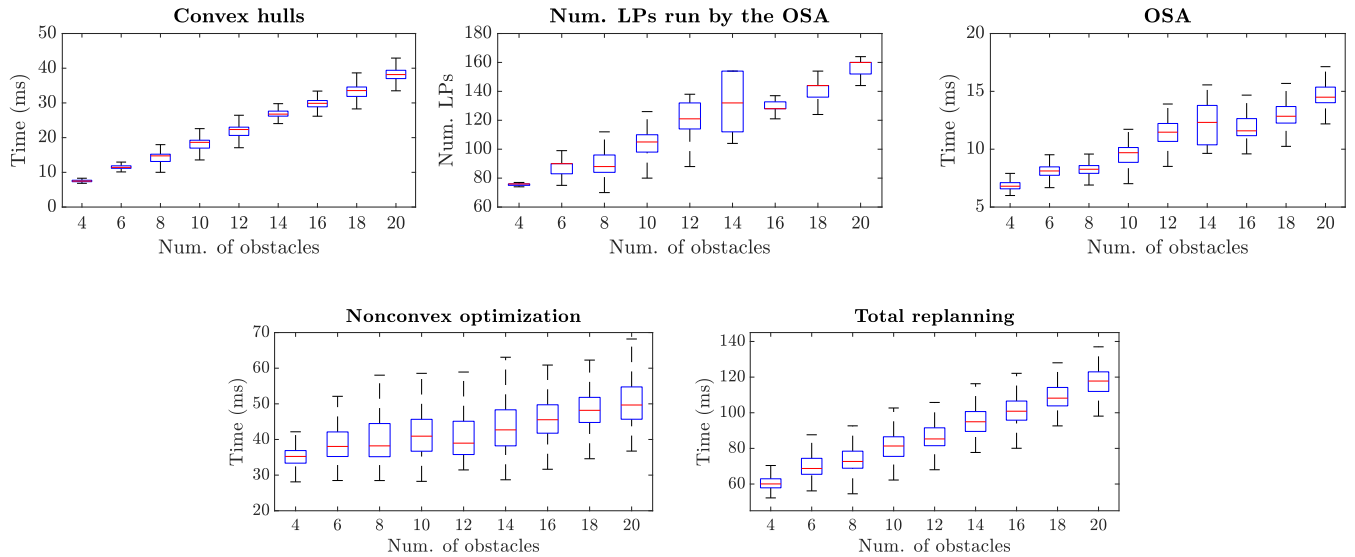
FIGURE 4. Computational analysis of different parts of the replanning step of PANTHER as a function of the number of obstacles. From left to right, and top to bottom: computation time of the generation of the convex hulls, number of linear programs (LPs) run by the OSA, computation time of the OSA, computation time of the nonconvex optimization, and total replanning time.

infeasible stops except *Wang*, that has a mean of 0.2 infeasible stops per run (light purple in Fig. 3F). In the fast scenario, PANTHER succeeds 70% of the runs, while all the other algorithms have a success rate below 27%. In terms of flight times and flight distances, most of the algorithms achieve very similar results in both scenarios, with a total flight time of approximately 20 s, and an approximate total flight distance of 41 m. The total flight distance for PANTHER is approximately 3 m more than the rest of the algorithms. This is expected, because PANTHER has the ability to modify the spatial path to maximize the visibility of the obstacles. Even with this longer flight distance, the flight time of PANTHER is very similar (and sometimes even shorter) than the rest of the algorithms.

3) COMPUTATIONAL ANALYSIS OF THE REPLANNING STEP AS A FUNCTION OF THE NUMBER OF OBSTACLES

We now compare the computational cost of different parts of the replanning step of PANTHER. As the computational cost of each part highly depends on the specific position of the obstacles relative to the UAV, we perform a Monte Carlo analysis by randomly deploying obstacles (which follow trefoil-knot trajectories) in the spherical shell [61] limited by two spheres of radii 2 m and 5 m. The starting location $\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$ m and $g_{\text{term}} = \begin{bmatrix} 6 & 0 & 1 \end{bmatrix}^T$ m are held fixed for every replanning iteration. The number of obstacles tested are {4, 6, ..., 18, 20}, and, for each number of obstacles, we run 10 simulations of 5.0 s each. For these simulations, the UAV includes all the deployed obstacles in the planning problem (i.e., the set I contains the indexes of all the obstacles deployed), and we let the UAV know the trajectory of the obstacles perfectly. The weights used are the same as the ones

used in section III-A2. The results are shown in Fig. 4, where can see that the computation time required for the convex hull generation, the OSA, the optimization, and the total replanning time change approximately linearly with the number of obstacles. Similarly, the number of linear programs run by the OSA also changes approximately linearly with the number of obstacles. The average solve time of one of these linear programs is 0.09 ms.

To obtain the ψ initial guess (section II-C2), the average runtime of the Dijkstra's algorithm on the ψ graph is 0.137 ms, and the average runtime to fit a spline to the ψ samples (Eq. 1) is 0.048 ms.

These results above show the computational analysis for the different parts of the replanning step of PANTHER (convex hull computation, generation of the $\mathbf{p}(t)$ and $\psi(t)$ initial guesses, and nonconvex optimization). For the computational cost of the tracker and predictor using real point clouds, see section III-B. The well-known results regarding the complexity analysis of the Hungarian algorithm are given in [41], [62].

B. REAL-WORLD EXPERIMENTS

We run an extensive set of hardware experiments, where a UAV needs to go from a starting point to a goal location while avoiding unknown dynamic obstacles. The UAV used is equipped with a Qualcomm® SnapDragon Flight, an Intel® NUC i7DNK, and an Intel® RealSense Depth camera D435i. The tracker, planner, and the camera run on the Intel® NUC, while the control and state estimation run on the Qualcomm® SnapDragon Flight. Note that the main onboard computer (Intel® NUC) has similar computational power to the onboard hardware used in the recent
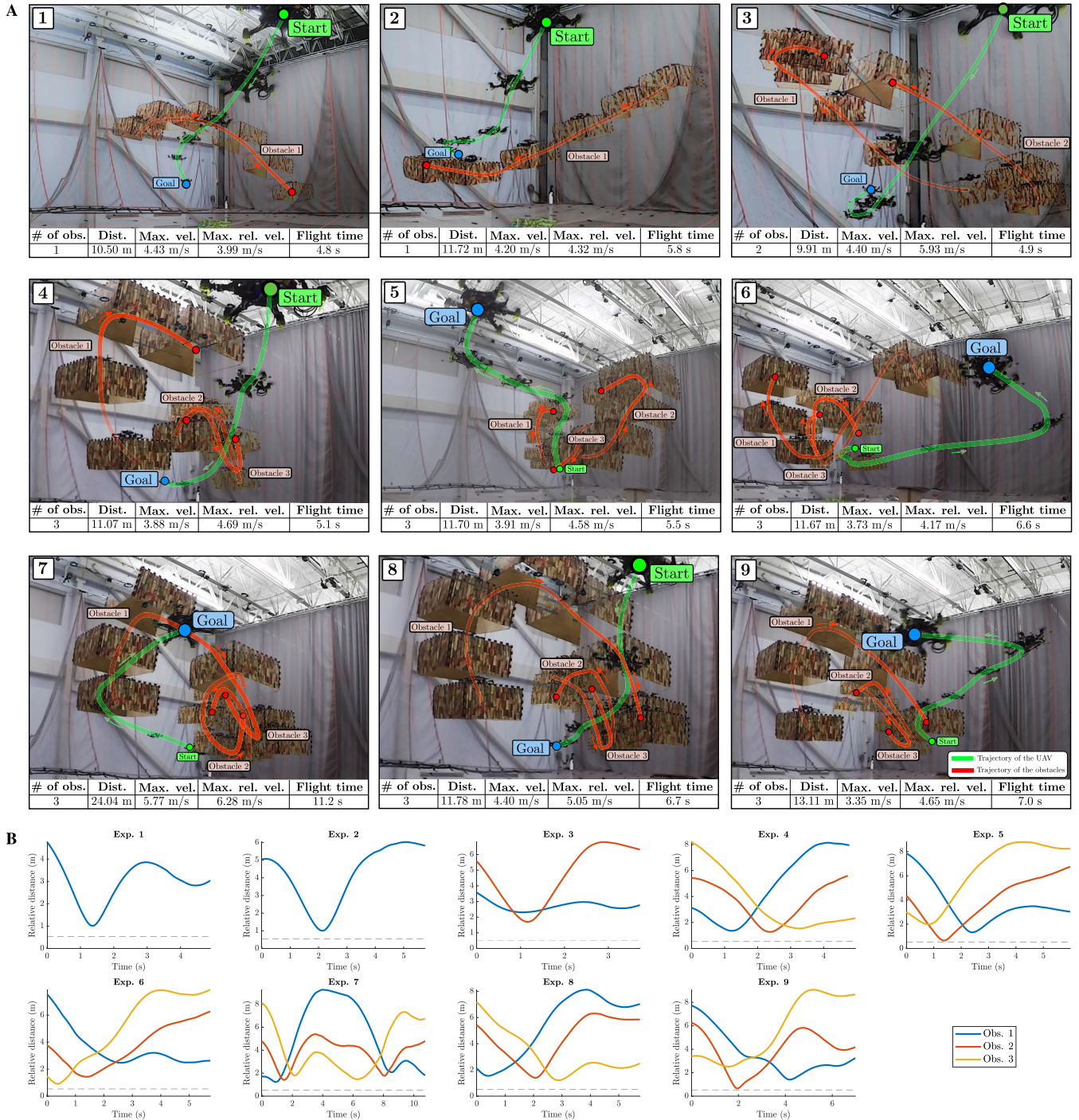
FIGURE 5. (A) Composite images of all the nine experiments. For visualization purposes, only the second half of Experiment 7 is shown. The table below every image shows the number of obstacles, flight distance, maximum velocity, maximum relative velocity (with respect to the obstacles), and flight time of each experiment. The number of obstacles is one, two, and three for the experiments 1–3, and 4–9 respectively. (B) Relative distances between the agent and each one of the obstacles. Any relative distance above the dashed line guarantees safety.

literature (e.g., [1], [6], [7], [12]). For the controller, we run the approach presented in [36], [63] at 100 Hz to generate the desired orientation and angular rates from $\mathbf{p}(t)$ and $\psi(t)$. The commanded thrusts for the motors are then found from these attitude commands using a geometric controller [64], which is run at IMU rate (500 Hz). For state estimation, we

use a visual inertial odometry (VIO) package [65] running at 30 Hz that leverages an extended Kalman filter to fuse the IMU measurements of the SnapDragon and the images of its downward-facing camera. To obtain a high-rate state estimate, we then integrate forward the IMU (which runs at 500 Hz) between consecutive VIO estimates.

Experiment 3

Experiment 6
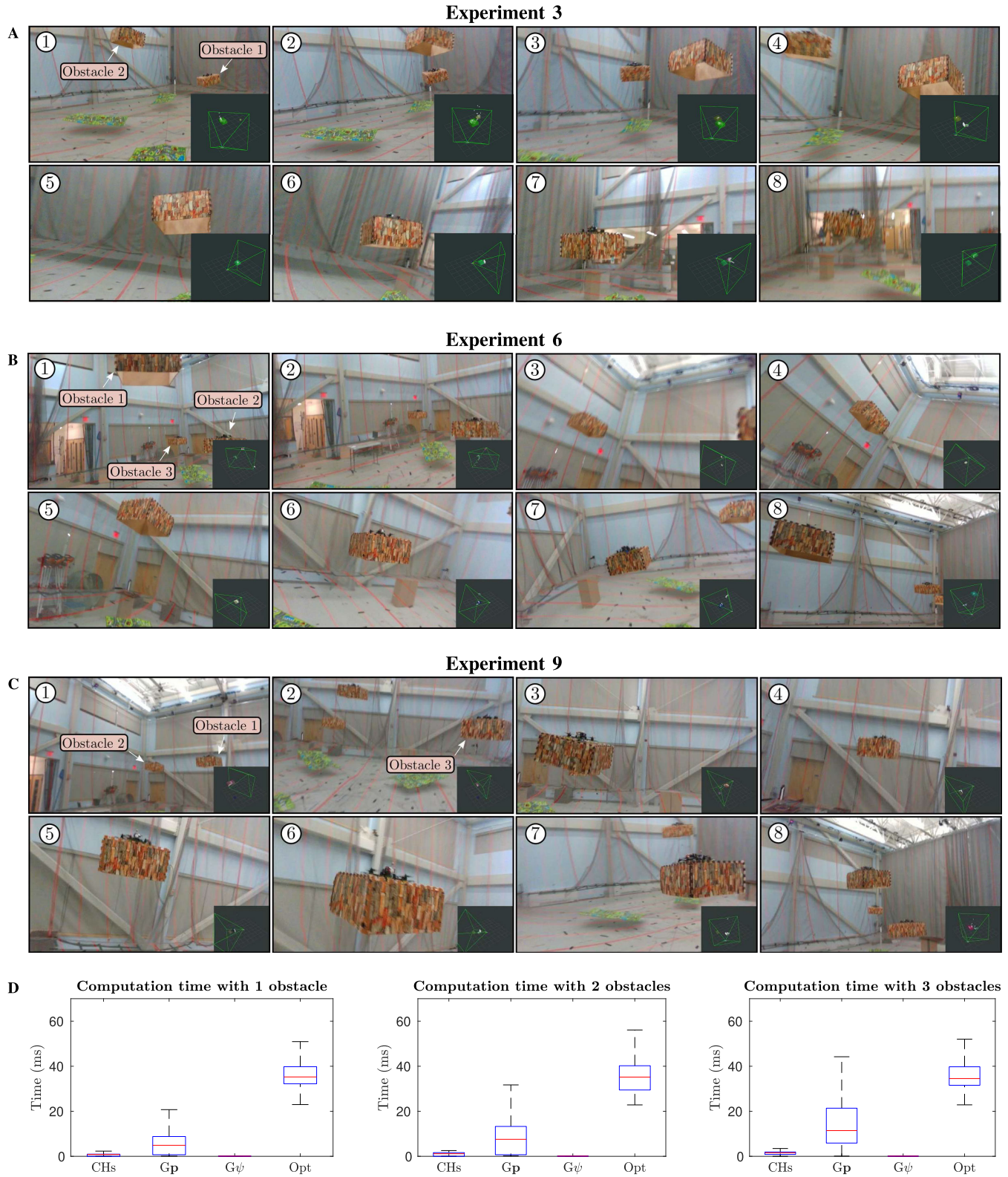
Experiment 9

FIGURE 6. Snapshots of the onboard camera in experiments 3 (A), 6 (B), and 9 (C). (D) Computation times for each part of a replanning step, measured on the onboard Intel® NUC i7DNK. The tracker, predictor, and the depth camera were also running on this computer at the same time these times were measured. The notation used is: CHs (convex hull computation for the polyhedral outer representations), G$\mathbf{p}$ (generation of the planes and the guess for the position $\mathbf{p}(t)$), Gψ (generation of the guess for $\psi(t)$) and Opt (Optimization time).

The IMU of the RealSense camera is not used. All the computation of this UAV is running onboard, and it does not have any prior knowledge of the trajectories and specific shape/size of the obstacles. The weights used for these experiments are $c_{\Psi_{max}} = 10^6$, $c_{FOV} = 1$, $c_\psi = 0$ rad^{-2}, $\alpha_\mathbf{j} = 0.05$ s^5/m^2, $\alpha_\psi = 0.1$ s^3/rad^2, $\alpha_{FOV} = 1$, $\alpha_\mathbf{g} = 2 \cdot 10^4$ m^{-2}, $\epsilon_1 = 0.1$, and $\epsilon_2 = 1$ s^2/m^2.

To generate the dynamic obstacles, we use three other UAVs with a Qualcomm® SnapDragon Flight, and equip them with a box-shaped frame of $\approx 0.6 \times 0.6 \times 0.3$ m^3. The obstacles are following trefoil-knot trajectories [60].

A total of 9 experiments were performed (see attached video). The composite images of the trajectories flown by the agent and by the obstacles, together with the number of obstacles, distance flown, maximum velocity, maximum relative velocity with respect to the obstacles, and total flight time of each one of the experiments are shown in Fig. 5A. Experiments 1 and 2 were done with one obstacle, experiment 3 with two obstacles, and experiments 4-9 with three obstacles. The maximum velocity achieved by the agent, 5.77 m/s, happened in experiment 7. In that same experiment, the maximum relative velocity (6.28 m/s) with respect to the obstacles is also achieved. The relative distances between the UAV and the obstacles are shown in Fig. 5B. Any relative distance above the dashed horizontal line guarantees safety between the agent and the corresponding obstacle. For experiments 3, 6, and 9, different snapshots of the onboard camera are shown in Figs. 6A, 6B, and 6C, respectively. Note how the planned trajectories try to keep an obstacle in the FOV at all times to aid in obstacle tracking and prediction.

The computation times are shown in Fig. 6D. All these computation times were measured onboard, with the UAV flying, and with the depth camera node and the tracker running on the same computer (Intel® NUC i7DNK). The mean total replanning times are 48.70, 51.66, and 58.59 ms for the experiments with 1, 2, and 3 obstacles respectively. The point cloud of the camera is generated at 90 Hz, and the tracker (clustering, assignment, and prediction) is able to process each point cloud in ≈ 8.6 ms.

IV. CONCLUSION

This work derived PANTHER, a perception-aware (PA) trajectory planner in dynamic environments. PANTHER is able to couple together the translation and the full rotation in the optimization, leading to PA trajectories computed in real time that maximize the presence of the obstacles in the FOV while minimizing their projected velocity. Extensive hardware experiments in unknown dynamic environments, with all the computation running onboard, and with relative velocities of up to 6.3 m/s have shown its effectiveness.

Our approach has also some limitations. Specifically, in the hardware experiments we observed the importance of the choice of the obstacle to include in the optimization (i.e., the choice of i^*, see Table 2 and section II-B): when should the UAV include a specific (already tracked) obstacle

in the PA term of the optimization, in order to predict its trajectory more accurately to be able to avoid it, and when should the UAV turn around to explore unknown space? This highlights the trade-off between exploration and exploitation: too much focus on exploitation may lead to collision with obstacles that were never detected, and too much focus on exploration may lead to a very poor trajectory prediction, and hence to a collision as well. Optimally solving this trade-off is a promising direction for future work.

Another possible direction of future work is to solve the trade-off between visibility and time optimality. This would entail adding the time minimization in the optimization problem of section II-D4, and would also allow to highlight the advantages of the Hopf fibration when flying aggressive trajectories that pass close to the singularity produced by the commonly-used maps presented in [33], [37] (first two definitions of Table 3).

Finally, another interesting research direction is how to incorporate disturbances in the planning problem, while still guaranteeing that the tracking error of the UAV remains bounded [66]. The incorporation of such disturbance information is especially important when flying outdoors under windy conditions, since a large deviation between the planned trajectory and the actual trajectory can provoke a collision with the obstacles.

REFERENCES

[1] J. Chen, T. Liu, and S. Shen, "Online generation of collision-free trajectories for quadrotor flight in unknown cluttered environments," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2016, pp. 1476–1483.

[2] F. Gao, L. Wang, B. Zhou, X. Zhou, J. Pan, and S. Shen, "Teach-repeat-replan: A complete and robust system for aggressive flight in complex environments," *IEEE Trans. Robot.*, vol. 36, no. 5, pp. 1526–1545, Oct. 2020.

[3] J. Tordesillas, B. T. Lopez, and J. P. How, "FASTER: Fast and safe trajectory planner for flights in unknown environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Nov. 2019, pp. 1934–1940.

[4] J. Lin, H. Zhu, and J. Alonso-Mora, "Robust vision-based obstacle avoidance for micro aerial vehicles in dynamic environments," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2020, pp. 2682–2688.

[5] D. Falanga, K. Kleber, and D. Scaramuzza, "Dynamic obstacle avoidance for quadrotors with event cameras," *Sci. Robot.*, vol. 5, no. 40, p. eaaz9712, Mar. 2020.

[6] Y. Wang, J. Ji, Q. Wang, C. Xu, and F. Gao, "Autonomous flights in dynamic environments with onboard vision," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2021, pp. 1966–1973.

[7] N. J. Sanket, C. M. Parameshwara, C. D. Singh, A. V. Kuruttukulam, C. Fermüller, D. Scaramuzza, and Y. Aloimonos, "EVDodgeNet: Deep dynamic obstacle dodging with event cameras," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2020, pp. 10651–10657.

[8] T. Nägeli, J. Alonso-Mora, A. Domahidi, D. Rus, and O. Hilliges, "Real-time motion planning for aerial videography with dynamic obstacle avoidance and viewpoint optimization," *IEEE Robot. Automat. Lett.*, vol. 2, no. 3, pp. 1696–1703, Jul. 2017.

[9] R. Bonatti, Y. Zhang, S. Choudhury, W. Wang, and S. Scherer, "Autonomous drone cinematographer: Using artistic principles to create smooth, safe, occlusion-free trajectories for aerial filming," in *Proc. Int. Symp. Exp. Robot.* Cham, Switzerland: Springer, 2018, pp. 119–129.

[10] G. Chen, W. Dong, X. Sheng, X. Zhu, and H. Ding, "Bio-inspired obstacle avoidance for flying robots with active sensing," 2020, *arXiv:2010.04977*.

[11] W. Ding, W. Gao, K. Wang, and S. Shen, "An efficient B-spline-based kinodynamic replanning framework for quadrotors," *IEEE Trans. Robot.*, vol. 35, no. 6, pp. 1287–1306, Dec. 2019.

[12] B. Zhou, J. Pan, F. Gao, and S. Shen, "RAPTOR: Robust and perception-aware trajectory replanning for quadrotor fast flight," 2020, *arXiv:2007.03465*.

[13] I. Spasojevic, V. Murali, and S. Karaman, "Perception-aware time optimal path parameterization for quadrotors," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2020, pp. 3213–3219.

[14] V. Murali, I. Spasojevic, W. Guerra, and S. Karaman, "Perception-aware trajectory generation for aggressive quadrotor flight using differential flatness," in *Proc. Amer. Control Conf. (ACC)*, Jul. 2019, pp. 3936–3943.

[15] 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, nos. 2–3, pp. 303–320, Mar. 2020.

[16] D. Falanga, P. Foehn, P. Lu, and D. Scaramuzza, "PAMPC: Perception-aware model predictive control for quadrotors," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 1–8.

[17] 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.

[18] I. Spasojevic, V. Murali, and S. Karaman, "Joint feature selection and time optimal path parametrization for high speed vision-aided navigation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 5931–5938.

[19] L. Bartolomei, L. Teixeira, and M. Chli, "Perception-aware path planning for UAVs using semantic segmentation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 5808–5815.

[20] K. Lee, J. Gibson, and E. A. Theodorou, "Aggressive perception-aware navigation using deep optical flow dynamics and PixelMPC," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 1207–1214, Apr. 2020.

[21] Z. Zhang and D. Scaramuzza, "Perception-aware receding horizon navigation for MAVs," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2018, pp. 2534–2541.

[22] G. Costante, C. Forster, J. Delmerico, P. Valigi, and D. Scaramuzza, "Perception-aware path planning," 2016, *arXiv:1605.04151*.

[23] M. W. Achtelik, S. Lynen, S. Weiss, M. Chli, and R. Siegwart, "Motion- and uncertainty-aware path planning for micro aerial vehicles," *J. Field Robot.*, vol. 31, no. 4, pp. 676–698, Jul. 2014.

[24] B. Penin, R. Spica, P. R. Giordano, and F. Chaumette, "Vision-based minimum-time trajectory generation for a quadrotor UAV," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 6199–6206.

[25] J. A. Preiss, K. Hausman, G. S. Sukhatme, and S. Weiss, "Simultaneous self-calibration and navigation using trajectory optimization," *Int. J. Robot. Res.*, vol. 37, nos. 13–14, pp. 1573–1594, Dec. 2018.

[26] K. M. Frey, T. J. Steiner, and J. P. How, "Towards online observability-aware trajectory optimization for landmark-based estimators," 2019, *arXiv:1908.03790*.

[27] P. Salaris, M. Cognetti, R. Spica, and P. R. Giordano, "Online optimal perception-aware trajectory generation," *IEEE Trans. Robot.*, vol. 35, no. 6, pp. 1307–1322, Dec. 2019.

[28] J. Thomas, J. Welde, G. Loianno, K. Daniilidis, and V. Kumar, "Autonomous flight for detection, localization, and tracking of moving targets with a small quadrotor," *IEEE Robot. Autom. Lett.*, vol. 2, no. 3, pp. 1762–1769, Jul. 2017.

[29] B. F. Jeon, D. Shim, and H. J. Kim, "Detection-aware trajectory generation for a drone cinematographer," 2020, *arXiv:2009.01565*.

[30] G. Li, A. Tunchez, and G. Loianno, "PCMPC: Perception-constrained model predictive control for quadrotos with suspended loads using a single camera and IMU," 2021, *arXiv:2107.10888*.

[31] J. Chen and S. Shen, "Using a quadrotor to track a moving target with arbitrary relative motion patterns," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 5310–5317.

[32] B. F. Jeon and H. J. Kim, "Online trajectory generation of a MAV for chasing a moving target in 3D dense environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Nov. 2019, pp. 1115–1121.

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

[34] I. Bendixson, "Sur les courbes définies par des équations différentielles," *Acta Mathematica*, vol. 24, no. 1, p. 1, 1901.

[35] L. E. J. Brouwer, "Über abbildung von mannigfaltigkeiten," *Mathematische Annalen*, vol. 71, no. 1, pp. 97–115, 1911.

[36] M. Watterson and V. Kumar, "Control of quadrotors using the Hopf fibration on $SO(3)$," in *Robotics Research*. Cham, Switzerland: Springer, 2020, pp. 199–215.

[37] M. Faessler, A. Franchi, and D. Scaramuzza, "Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories," *IEEE Robot. Autom. Lett.*, vol. 3, no. 2, pp. 620–626, Apr. 2018.

[38] M. Watterson, A. Zahra, and V. Kumar, "Geometric control and trajectory optimization for bidirectional thrust quadrotors," in *Proc. Int. Symp. Exp. Robot.* Cham, Switzerland: Springer, 2018, pp. 165–176.

[39] W. M. Weisstein and W. Eric. (2020). *Sigmoid Function*. Accessed: Feb. 20, 2021. [Online]. Available: https://mathworld.wolfram.com/SigmoidFunction.html

[40] Mathworks. (2020). *Normal Inverse Cumulative Distribution Function*. Accessed: Feb. 22, 2021. [Online]. Available: https://www.mathworks.com/help/stats/norminv.html

[41] H. W. Kuhn, "The Hungarian method for the assignment problem," *Nav. Res. Logistics Quart.*, vol. 2, nos. 1–2, pp. 83–97, 1955.

[42] J. Tordesillas and J. P. How, "MADER: Trajectory planner in multi-agent and dynamic environments," *IEEE Trans. Robot.*, vol. 38, no. 1, pp. 463–476, Feb. 2022.

[43] D. W. Lyons, "An elementary introduction to the Hopf fibration," *Math. Mag.*, vol. 76, no. 2, pp. 87–98, 2003.

[44] R. J. Hyndman and G. Athanasopoulos, *Forecasting: Principles and Practice*. OTexts, 2018. [Online]. Available: https://otexts.com/fpp2/

[45] L. Janson, E. Schmerling, and M. Pavone, "Monte Carlo motion planning for robot trajectory optimization under uncertainty," in *Robotics Research*. Cham, Switzerland: Springer, 2018, pp. 343–361.

[46] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numer. Math.*, vol. 1, no. 1, pp. 269–271, Dec. 1959.

[47] H. Kuhn and A. Tucker, *Nonlinear Programming*, vol. 13. Berkeley, CA, USA: Univ. California Press, 1951, p. 54.

[48] W. Karush, "Minima of functions of several variables with inequalities as side constraints," M.S. thesis, Dept. Math., Univ. Chicago, Chicago, IL, USA, 1939.

[49] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[50] B. Morrell, M. Rigter, G. Merewether, R. Reid, R. Thakker, T. Tzanetos, V. Rajur, and G. Chamitoff, "Differential flatness transformations for aggressive quadrotor flight," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 5204–5210.

[51] R. Spica, P. R. Giordano, M. Ryll, H. H. Bülthoff, and A. Franchi, "An open-source hardware/software architecture for quadrotor UAVs," *IFAC Proc. Volumes*, vol. 46, no. 30, pp. 198–205, 2013.

[52] R. E. Allen and M. Pavone, "A real-time framework for kinodynamic planning in dynamic environments with application to quadrotor obstacle avoidance," *Robot. Auton. Syst.*, vol. 115, pp. 174–193, May 2019.

[53] J. Welde, J. Paulos, and V. Kumar, "Dynamically feasible task space planning for underactuated aerial manipulators," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 3232–3239, Apr. 2021.

[54] J. Tordesillas and J. P. How, "MINVO basis: Finding simplexes with minimum volume enclosing polynomial curves," 2020, *arXiv:2010.10726*.

[55] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.*, vol. 106, no. 1, pp. 25–57, May 2006.

[56] *Ipopt Documentation*. Accessed: Jan. 20, 2022. [Online]. Available: https://coin-or.github.io/Ipopt/OUTPUT.html

[57] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi—A software framework for nonlinear optimization and optimal control," *Math. Program. Comput.*, vol. 11, no. 1, pp. 1–36, 2019.

[58] HSL. (2013). *A Collection of Fortran Codes for Large Scale Scientific Computation*. [Online]. Available: http://www.hsl.rl.ac.U.K

[59] W. M. Weisstein and W. Eric. (2020). *Simpson's Rule*. Accessed: Feb. 17, 2021. [Online]. Available: https://mathworld.wolfram.com/SimpsonsRule.html

[60] Wolfram MathWorld. (Feb. 2021). *Trefoil Knot*. Accessed: Feb. 15, 2021. [Online]. Available: https://mathworld.wolfram.com/TrefoilKnot.html

[61] Wolfram MathWorld. (Jan. 2022). *Trefoil Knot*. Accessed: Jan. 24, 2022. [Online]. Available: https://mathworld.wolfram.com/SphericalShell.html

[62] J. Edmonds and R. M. Karp, "Theoretical improvements in algorithmic efficiency for network flow problems," *J. ACM*, vol. 19, no. 2, pp. 248–264, 1972.

[63] B. T. Lopez, "Low-latency trajectory planning for high-speed navigation in unknown environments," Massachusetts Inst. Technol., Cambridge, MA, USA, 2016.

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

[65] *Qualcomm Machine Vision SDK*. Accessed: Feb. 15, 2021. [Online]. Available: https://developer.qualcomm.com/software/machine-vision-sdk

[66] B. T. Lopez, J.-J.-E. Slotine, and J. P. How, "Dynamic tube MPC for nonlinear systems," in *Proc. Amer. Control Conf. (ACC)*, Jul. 2019, pp. 1655–1662.

JONATHAN P. HOW (Fellow, IEEE) received the B.A.Sc. degree from the University of Toronto, in 1987, and the S.M. and Ph.D. degrees in aeronautics and astronautics from the Massachusetts Institute of Technology (MIT), in 1990 and 1993, respectively. Prior to joining MIT, in 2000, he was an Assistant Professor at Stanford University. He is currently the Richard C. Maclaurin Professor of aeronautics and astronautics with MIT. He is a fellow of AIAA and was elected to the National Academy of Engineering, in 2021. Some of his awards include the IEEE CSS Distinguished Member Award, in 2020; the AIAA Intelligent Systems Award, in 2020; the IROS Best Paper Award on Cognitive Robotics, in 2019; and the AIAA Best Paper in Conference Awards, from 2011 to 2013. He was the Editor-in-Chief of *IEEE Control Systems Magazine*, from 2015 to 2019.
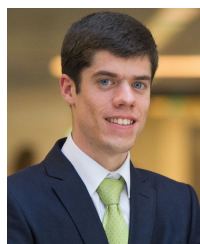
JESUS TORDESILLAS (Student Member, IEEE) received the B.S. and M.S. degrees in electronic engineering and robotics from the Technical University of Madrid, Spain, in 2016 and 2018, respectively. He then received the M.S. degree in aeronautics and astronautics from the Massachusetts Institute of Technology (MIT), in 2019, where he is currently pursuing the Ph.D. degree with the Aeronautics and Astronautics Department, as a member of the Aerospace Controls Laboratory (MIT) under the supervision of Jonathan P. How. His research interests include path planning for UAVs in unknown environments and optimization. He was a finalist for the Best Paper Award on Search and Rescue Robotics in IROS 2019.