Journal of Intelligent & Robotic Systems (2021) 103:71
https://doi.org/10.1007/s10846-021-01526-8

REGULAR PAPER

A Guidance System for Tactical Autonomous Unmanned Aerial Vehicles

Julius A. Marshall[1] · Robert B. Anderson[1] · Wen-Yu Chien[2] · Eric N. Johnson[2] · Andrea L'Afflitto[1]

Received: 24 September 2020 / Accepted: 1 November 2021

Abstract
This paper presents an original guidance system able to confer a tactical behavior to multi-rotor unmanned aerial vehicles (UAVs), such as quadcopters, that operate in potentially hostile, unknown, cluttered environments. By applying this guidance system, UAVs complete the assigned tasks, such as reaching a goal set, while minimizing both their exposure to opponents, whose location is unknown, and the predictability of their trajectories. A taxonomy of flight behaviors is provided to help users tuning those parameters that characterize the UAV's level of cautiousness. This guidance system is supported by an original navigation system that exploits exclusively information gathered by onboard cameras and inertial measurement units. Numerical simulations and flight tests validate the applicability of the proposed guidance system in real-time, while performing all calculations aboard the UAV.

Keywords Path planning · Trajectory planning · Unmanned aerial vehicles · Tactical behavior

1 Introduction

This paper presents a tactical guidance system for multi-rotor unmanned aerial vehicles (UAVs), such as quadcopters, to be employed in missions wherein the aircraft must minimize its exposure to opponents, whose location is unknown. The proposed guidance system allows an autonomous vehicle to reach a goal set, whose position relative to the vehicle's initial position is given, without any prior knowledge of the environment. Estimates on the UAV's state and a map of the environment are produced by a navigation system that relies exclusively on onboard sensors such as cameras, accelerometers, and gyroscopes. As highlighted in [99], UAVs equipped with guidance systems such as the one proposed in this paper could have been used by law enforcement agencies during the tragic events on October 1, 2017 in Las Vegas, NV, to aid situational awareness, reduce personnel exposure, and surprise and distract opponents.

The first element of novelty of the proposed guidance system is its ability to instill a tunable tactical behavior in autonomous vehicles operating in unknown environments. To the authors' knowledge, there is no guidance system able to attain this result without any prior knowledge on the presence of adversaries, if any. To instill a tactical behavior in UAVs, our guidance system pursues concurrently two common tactical strategies, which are inspired by the behavior of house mice [35] and the tactics of ground troops operating in potentially hostile environments [94]. The first of these strategies consists of flying sufficiently close to obstacles such as walls, pillars, and other similar environmental features [1]. Coasting obstacles minimizes the vehicle's exposure to sensors that acquire targets in direct line-of-sight, such as cameras, LiDAR systems, and RADARs. Furthermore, the obstacles' hard surfaces can distort the signal emitted by active sensors such as

✉ Andrea L'Afflitto
a.lafflitto@vt.edu

Julius A. Marshall
mjulius@vt.edu

Robert B. Anderson
andersonb@vt.edu

Wen-Yu Chien
wuc188@psu.edu

Eric N. Johnson
eric.johnson@psu.edu

[1] Grado Department of Industrial and Systems Engineering, Virginia Tech, Blacksburg, VA 24061, USA

[2] Department of Aerospace Engineering, The Pennsylvania State University, University Park, PA 16802, USA

Published online: 25 November 2021

LiDARs and RADARs and hence, contribute to the UAV's concealment. The second strategy to induce a tactical behavior is to regulate the UAV's velocity according to its distance from the obstacles' set. For instance, if the UAV is shielded by some obstacle, then it should proceed slowly to allow the navigation system to produce a more accurate obstacles' map. Alternatively, if no obstacle conceals the vehicle's position, then the UAV should proceed as fast as possible [1].

The second element of novelty of the proposed guidance system lays in the possibility given to users to set *a priori* the vehicle's level of cautiousness or recklessness. This level of cautiousness can be set by tuning nine user-defined parameters, whose roles are discussed in detail.

The third element of novelty of this paper lays in a taxonomy of flight behaviors. This classification of the UAV's behaviors is instrumental to predict the vehicle's behavior and, hence, enable users to tune those parameters that characterize the UAV's level of cautiousness. This taxonomy of flight behaviors has been deduced through software-in-the-loop simulations by varying one user-defined parameter at the time, multiple user-defined parameters, the UAV's initial conditions in a given environment, and the occupancy map. This numerical analysis shows that if these user-defined parameters are set to produce a more tactical behavior, then small variations in the user-defined parameters, the UAV's take-off position, and the obstacles' set produce reference trajectories that differ considerably from the shortest trajectory. Therefore, by imposing a tactical behavior, the UAV's reference trajectories are less predictable. Conversely, if these user-defined parameters are set to produce a reckless behavior, then the UAV's reference trajectories are more predictable in the sense that small variations in the user-defined parameters, the UAV's take-off position, and the obstacles' set produce reference trajectories that cluster around the shortest trajectory. Multiple flight tests verify the outcome of this numerical analysis.

Given a voxel map produced by the onboard navigation system, the UAV's reference path is computed as a sequence of unoccupied voxels that minimizes a cost function. A novel feature of the proposed guidance system lays in the fact that this cost function has been designed to capture both the UAV's need to reach the goal set and its need to coast the obstacles' set to enable a tactical behavior. Reference paths are searched at each time step by employing the A^* search algorithm. Employing a fast model predictive control approach, the aircraft reference trajectory is computed as the solution of an optimal control problem, whose waypoints are given by the vehicle's reference path. Similarly to the path planner, a cost function that captures the vehicle's need to reach the goal set and be sheltered from potential threats by coasting obstacles underlies the trajectory planner. The trajectory planning subsystem also allows to regulate the aircraft's velocity at each waypoint according to tactical needs. The algorithm that enables fast solutions of the trajectory planning problem within the model predictive control framework recasts the trajectory planning problem as a quadratic programming problem, whose underlying matrices are in block-tridiagonal form. This quadratic programming problem can be solved numerically by means of state-of-the-art algorithms that exploit these matrices' structure, fast computation of the Newton step, and an infeasible start Newton method [100]. These algorithms guarantee a time complexity that is polynomial in the dimensions of the state vector and the number of constraint equations, and is linear in the time horizon. The use of a warm start contributes to reducing the number of iterations by a factor of five or more [100].

Collision avoidance constraint sets are generated by employing a novel approach, which can be synthesized as follows. Given a voxel map of the environment, the proposed collision avoidance algorithm solves a quadratic optimization problem and computes ellipsoids that contain the vehicle's position and exclude the obstacles. Thus, convex constraint sets, whose boundaries are captured by affine functions, are generated by sampling these ellipsoids. The boundaries of these constraint sets intersect the obstacles' set at one or multiple points, and since coasting obstacles is one of the strategies pursued to enable tactical behaviors, searching for reference trajectories in some space adjacent to the obstacles' set is essential. This method to capture collision avoidance constraints allows to generate real-time solutions to the trajectory planning problem. Numerical simulations show that the proposed approach to generate collision avoidance constraints is consistently faster than two alternative, state-of-the-art algorithms for collision avoidance constraint generation, namely IRIS (Iterative Regional Inflation by Semidefinite programming) [3, 24, 25] and SFC (Safe Flight Corridors) [53].

A typical approach to guarantee safety margins, while enforcing collision avoidance constraints, is to introduce offsets of user-defined size or inflate the occupied voxels. However, the presence of large offsets may contrast with the strategy of coasting obstacles to produce a tactical behavior. Moreover, these techniques may prevent the UAV from traversing narrow passages. In this paper, to discourage the UAV's reference trajectory from approaching the boundary of its constraint set too closely, a Kreisselmeier-Steinhauser penalty function is employed to introduce soft constraints [43]. This penalty function does not disrupt the computationally efficient sparse pattern of the Hessian matrix underlying the proposed quadratic programming problem [77, 104].

Classical approaches to solve the model predictive control problem as a quadratic programming problem involve matrices that are more compact than those employed in this

paper, but whose structure is less computationally efficient [16, 55]. Among alternative approaches to efficiently solve online the trajectory planning problem as a quadratic programming algorithm it is worthwhile to recall [8, 41, 79, 80, 105], which leverage the sparsity of the underlying matrices and exploit active-set methods, interior-point methods, or the dual piecewise smooth Newton method. However, none of these works employ matrices structured in the same form as those presented in this paper.

In this paper, we compare the performance of the proposed guidance system with the performance of the MAV Voxblox planner [68, 69]. The MAV Voxblox planner is a state-of-the-art guidance system that, similarly to the proposed guidance system, comprises both an optimization-based path planner and a trajectory planner based on the model predictive control framework. This analysis is not aimed at assessing the ability of MAV Voxblox to produce tactical reference trajectories, since it is not designed for this scope. The scope of this analysis is to showcase the ability of the proposed guidance system to effectively outline reference trajectories in unknown environments over advanced algorithms employing a similar architecture. Numerical simulations performed in an unknown maze provided by [68], show that the MAV Voxblox required several attempts to traverse the densely occupied areas. The proposed guidance system did not require multiple attempts to find a solution to the trajectory planning problem both when tasked with finding reckless trajectories and when tasked with finding tactical trajectories. Furthermore, the MAV Voxblox produced reference trajectories that are longer than the reckless reference trajectories and the tactical reference trajectories produced by the proposed guidance system. However, reducing the flight time and the distance covered, compatibly with the mission constraints, is a desirable feature.

This paper is organized as follows. In Section 2, we present a literature review on guidance systems for tactical UAVs, and highlight key features of the proposed architecture by surveying multiple path planning, optimal trajectory planning, and collision avoidance techniques from a tactical mission planning perspective. In Section 3, the notation used in this paper is outlined. The proposed guidance system, the vision based navigation system generating the voxel map, and their integration are discussed in Sections 4, 5, and 6, respectively. The results of software-in-the-loop simulations to validate the proposed guidance system and deduce a taxonomy of flight behaviors are presented in Section 7. Section 8 illustrates the results of indoor flight tests, and prove that software-in-the-loop simulation are useful to predict average behaviors of the proposed guidance system. Section 9 compares the performance of the proposed guidance system and MAV Voxblox. Finally, Section 10 draws conclusions on this work.

2 Literature Review and Relevance of the Proposed Guidance System

In this section, we survey existing guidance systems for tactical UAVs and highlight distinctive features of the proposed guidance system. Successively, we examine some popular path planning and optimization-based trajectory planning techniques for UAVs and discuss their applicability to guidance systems for tactical UAVs. Finally, we review state-of-the-art techniques currently employed in trajectory planning for UAVs to generate collision-free constraint sets and discuss the relevance of the proposed algorithm to solve this problem in real-time.

2.1 Guidance Systems for Tactical UAVs

The literature on guidance systems for UAVs moving along a collision-free path of low exposure to observers, also known as autonomous tactical UAVs, is relatively under-explored. To the authors' knowledge, there is no guidance system that, similarly to the one proposed herein, comprises both a path planner and a trajectory planner, allows the user to set *a priori* the UAV's level of cautiousness or recklessness, relies on onboard sensors only, and is able to operate without prior knowledge of the environment and the opponents' location.

Recently, a guidance system for stealthy UAVs was presented in [110], where sparse A^*, D^*, and learning real-time A^* are employed as path planners, and their performances are compared to one another. However, despite the proposed guidance system, the authors in [110] assume to know that the opponents detect UAVs by employing netted RADARs. Additionally, the guidance system proposed in [110] employs a classical, but not fast, model predictive control algorithm for trajectory planning and does not exploit the obstacles' set to conceal the UAV from RADARs Additionally, this work does not account for the UAV's navigation system, and results are not verified by means of flight tests.

An alternative guidance system for UAVs attempting to reach a given location and minimize their exposure to RADAR systems is presented in [17]. However, the authors of this work assume exact knowledge of the RADAR's position. Furthermore, reference paths were computed as solutions of a mixed integer linear programming problem, and this approach is unsuitable in real-time and for large number of constraints and variables needed for realistic mission scenarios. Lastly, mixed integer linear programming is sensitive to uncertainties in the underlying models [20, 75].

To induce a stealthy behavior in UAVs, the authors of [28] produced corridor maps by computing visibility polygons that describe the field of view of the observers. Thus, the A^* algorithm was exploited to generate safe

reference paths. Very recently, a method for tracking a ground vehicle from a UAV with a low risk of being detected was developed in [33]. Finally, the authors of [58] introduced a two-dimensional guidance system for covert autonomous robots that quantifies the robot's visibility at each obstacle-free partition of a map, and generates a collision-free path that minimizes visibility. The results in [28, 33, 58], however, rely on *a priori* information about the environment, and do not explicitly consider the cases wherein the vehicle is in an unexplored area.

Moving at the highest speed possible is a technique employed by some prey animals to escape their predators [113], and multiple guidance systems have been recently designed to allow UAVs to operate at high speed in unknown, cluttered environments. Among these works, it is worthwhile recalling [29, 48, 50, 78, 85, 86, 89, 92, 111, 112] to cite a few of the latest results in this area. However, relying only on speed as a tactical strategy to win a pursuit-evasion game can be effective if sufficient conditions on the agents' initial conditions and relative velocity, such as those presented in [34, 87, 88], are met. Additionally, the speed of any commonly used ammunition and the accuracy of existing automatic pointing systems render tactical approaches for UAVs based only on speed largely ineffective. Lastly, speed is not an effective strategy in the case the UAV must be concealed from electromagnetic sensors such as cameras, RADARs, and LiDARs.

2.2 Guidance Systems for UAVs Based on Path and Trajectory Planners

As discussed in [78], guidance systems for UAVs often embed both a path planner and a trajectory planner. The path planner is usually tasked with finding a collision-free sequence of waypoints that connect the vehicle's current location to the goal set. Path planners rarely account for the UAV's dynamics and hence, path following may result in a dynamically infeasible problem. In quadcopters, propellers are parallel to the vehicle's yaw axis and hence, the thrust force needed to translate the aircraft must be realized by pitching and rolling the vehicle. To overcome this limitation, the trajectory planner is usually tasked with computing a dynamically feasible time-parameterized curve that interpolates these waypoints. However, computing reference trajectories that interpolate all of the waypoints outlined by the path planer may be impossible for existing trajectory planners in problems of practical interest. For instance, optimization-based trajectory planners usually operate on convex sets since convexity of the underlying cost function over some convex set guarantees the existence of at least a solution. However, there may not exist a convex set that contains both the UAV's current position and all waypoints, and excludes all obstacles. For these reasons, in general, the trajectory planning problem is solved locally by interpolating multiple waypoints generated by the path planner.

A guidance system for quadcopters that, similarly to the one presented in this paper, is based on a path planner and a trajectory planner, was proposed in [24, 30, 53, 54]. Although this guidance system was not designed to support tactical missions, it is worthwhile discussing some of its main differences with the proposed work. The authors in [24, 30, 53, 54] proposed a path planner based on both the A^* algorithm and the jump point search algorithm to prune voxel elements. However, the rules to prune voxel elements may prevent the UAV from coasting the obstacles' set. Furthermore, their cost-to-come function does not allow rewarding paths coasting the obstacles' set more closely. Therefore, this framework is unsuitable for tactical missions. The trajectory planner presented in [24, 30, 53, 54] employs a linear receding horizon algorithm to control the output-feedback linearized equations of motion of the UAV, which is useful if aggressive maneuvers need to be performed, and the vehicle's inertial and aerodynamic properties are sufficiently well-known. In this paper, instead, we employ the linearized equations of motion deduced by applying Taylor's theorem since the UAV is not tasked to perform aggressive maneuvers. However, the proposed framework can be employed to control the output-feedback linearized equations of motion of the quadcopter. In [53], collision avoidance constraints are captured as hard constraints by polyhedral sets, which are computed from ellipsoids that exclude the obstacles' set along the reference path. The proposed algorithm to compute collision avoidance constraints involves a single ellipsoid and can be solved as a linear program using interior point methods. Furthermore, applying the framework presented in [24, 30, 53, 54], the risk of collisions with the obstacles' set can be mitigated by introducing constant offsets to enforce user-defined safety margins. However, this approach does not prevent sudden increases in the control effort due to the activation of the shifted hard constraints [9, Ch. 4]. In this paper, we rely on the Kreisselmeier-Steinhauser function [43], and not constant offsets defined *a priori*, to introduce soft constraints and discourage the UAV's trajectory from coasting obstacles too closely. In addition to the advantage of allowing user-defined safety margins, soft constraints also prevent sudden increases in the control effort associated with the constraints' activation.

Some recent guidance systems based on both path and trajectory planners are presented in [21, 62, 67, 69, 71, 82], to cite a few. In the following, we survey several path and trajectory planners for UAVs and discuss their possible use for the design of guidance systems for tactical UAVs.

2.3 Path Planning for Autonomous UAVs

The A^* framework has been chosen to support the proposed path planning algorithm for its ability to generate paths that are optimal with respect to a user-defined cost function, for its fast convergence, and for its low time complexity [51]. For these appealing features, this approach has been already employed by other researchers, such as [23, 73, 98] to name a few. However, to the authors' knowledge there is no tactical path planning algorithm based on A^*. To motivate our choice for the path planning algorithm, in the following we briefly survey multiple approaches to the path planning problem for UAVs.

The probabilistic roadmap method [12, 56] provides an advantageous path planning technique because it is sampling-based. Furthermore, this method guarantees low processing time in high-dimensional problems by ignoring points in the obstacle region. However, the probabilistic roadmap is not based on optimization criteria [56] and hence, can not be employed in tactical path planning to assess the risk associated with each feasible path. Similarly, the rapidly exploring random tree method (RRT) and its numerous variations [42, 66] provide computationally advantageous approaches. However, RRT may suffer from biases produced by the random nodes' generator [66].

Search algorithms based on the use of Voronoi diagrams [18, 22], while easy to implement, are inefficient in high dimensions, and require complex data structures and long pre-processing times [64]. Furthermore, Voronoi diagrams inherently find paths of *maximum* distance from obstacles [59], and, in general, this behavior produces non-tactical paths.

Some bio-inspired algorithms [70, 107, 108], such as evolutionary algorithms, allow to solve non-deterministic polynomial-time-hard (NP-hard) multi-objective path planning problems, but suffer from high time complexity. Alternative bio-inspired path planning algorithms, such as those based on neural networks, require training with given data [84] and hence, may not be suitable for mission scenarios involving complex, unknown environments. Lastly, the computational cost of bio-inspired path planners still hinders real-time path planning on small UAVs in three-dimensional environments [81, 106].

Mixed integer linear programming [7] and disjunctive convex programming [11] allow to find reference paths in the presence of multiple collision avoidance constraint sets. However, the computational times of these techniques are in the orders of tens of seconds in the presence of large numbers of obstacles [11], and hence, may not be suitable to design path planners for tactical UAVs.

The D^* and the D^*-lite algorithms [39, 40] provide appealing alternatives to A^* for their speed of execution and good performance in the presence of rapidly changing obstacles' sets. Future work directions of the proposed path planning algorithm involve the use of the D^*-lite framework.

2.4 Optimal Trajectory Planning for Autonomous UAVs

In this paper, a fast, linear model predictive control framework has been chosen to compute reference trajectories for its ability to rapidly generate trajectories that are dynamically feasible and optimal with respect to some user-defined cost function. To motivate our choice for the trajectory planning algorithm, in the following we briefly survey multiple approaches to the trajectory planning problem for UAVs.

Direct and indirect multiple shooting methods [76], direct and indirect transcription methods [38, 95], and level set methods [45, 83] have been employed to compute reference trajectories as solutions of optimal control problems. These methods, however, are computationally expensive and, considering the state-of-the-art of single board computers for Class I UAVs, are slow in producing reference trajectories and reference yaw angles in complex scenarios.

The optimal trajectory planning problem has been addressed by means of the receding horizon [74] and the model predictive control techniques [44]. Earlier implementations of model predictive control and receding horizon control were applicable to slow processes only due to their high computational costs [44, Ch. 1]. However, recent advances, such as those exploited in this paper, produce fast solutions to linear-quadratic optimal control problems [14, 77]. Recently, dynamically feasible collision-free trajectories in the receding horizon control framework have been produced using an innovative B-spline and a nonuniform kinodynamic search algorithm with control point optimization based on quadratically constrained quadratic programming [90].

More recently, trajectory planning methods based on nonlinear-nonquadratic model predictive control approaches have been proposed [2]. However, if a multi-rotor UAV needs to perform aggressive maneuvers, then the use of nonlinear model predictive control algorithms is computationally justified by the high-precision of the resulting reference trajectories [37]. In this paper, a tactical behavior is not induced by performing aggressive maneuvers but, by enabling more cautious strategies.

A very recent approach to optimal trajectory planning for UAVs operating in spaces densely packed with obstacles has been proposed in [102]. According to this approach, first the free boundary conditions are optimized while holding the UAV's travel time fixed, and then the UAV's travel time is optimized while holding the boundary conditions fixed.

2.5 Computing Convex Collision-Free Constraint Sets

In the proposed trajectory planning subsystem, collision avoidance is enforced by searching convex spaces, which do not intersect the obstacles' set and whose boundaries are piece-wise affine. These convex sets are obtained by sampling ellipsoids computed as solutions of a quadratic discrimination problem. Alternative approaches are given by IRIS [3, 24, 25] and SFC [53].

Assuming that the point cloud generated by the UAV's navigation system to identify obstacles has been partitioned into multiple convex obstacles, the IRIS algorithm has been designed for quickly computing large polytopic and ellipsoidal regions of obstacle-free space around a UAV through a series of convex optimizations. Although both the algorithm presented in this paper and IRIS rely on ellipsoids to identify collision-free partitions of the environment, our algorithm does not require to identify convex clusters of obstacle points. Furthermore, a key step in IRIS is to identify those obstacle points that are closest to an ellipsoid containing the UAV. This computationally onerous step is not required by the proposed algorithm. Finally, as shown by means of a numerical analysis, the proposed algorithm is faster than IRIS in computing convex obstacle-free sets containing the UAV.

The SFC algorithm [53] computes collections of convex overlapping polyhedra that model free space and contain some path connecting the UAV's current position to the goal position. More recently, the SFC algorithm has been extended to trajectory planning on Riemannian manifolds and has been applied to the UAV trajectory planning problem [103]. As shown by means of numerical analysis, the proposed algorithm to compute convex constraint sets for collision avoidance is faster than the SFC algorithm.

3 Notation, Definitions, and Mathematical Preliminaries

The *interior of the set* $\mathcal{E} \subset \mathbb{R}^n$ is denoted by $\mathring{\mathcal{E}}$, the *boundary of* $\mathcal{E} \subset \mathbb{R}^n$ is denoted by $\partial\mathcal{E}$, and the *closure of* $\mathcal{E}$ is denoted by $\overline{\mathcal{E}}$. The *ith element of the canonical basis of* $\mathbb{R}^n$ is denoted by $\mathbf{e}_{i,n} \triangleq [0, \ldots, 1, \ldots, 0]^{\mathrm{T}}$, the *zero vector* in $\mathbb{R}^n$ is denoted by 0_n, the *zero* $n \times m$ *matrix* in $\mathbb{R}^{n\times m}$ is denoted by $0_{n\times m}$, and the *identity matrix* in $\mathbb{R}^{n\times n}$ is denoted by $\mathbf{1}_n$. The diagonal matrix $D \in \mathbb{R}^{n\times n}$, whose diagonal entries are given by $\{k_1, \ldots, k_n\}$, is denoted by $D = \mathrm{diag}(k_1, \ldots, k_n)$. The block-diagonal matrix formed by $M_i \in \mathbb{R}^{n_i\times n_i}$, $i = 1, \ldots, p$, is denoted by $M = \mathrm{blockdiag}\left(M_1, \ldots, M_p\right)$. The Kronecker product of $A \in \mathbb{R}^{n\times m}$ and $B \in \mathbb{R}^{p\times q}$ is denoted by $A \otimes B$ [10, Def. 7.1.2].

Given $x, y \in \mathbb{R}^n$, if each component of x is larger than the corresponding component of y, then we write $x >> y$. If each component of x is not smaller than the corresponding component of y, then we write $x \geq\geq y$. If each component of x is smaller than the corresponding component of y, then we write $x << y$. If each component of x is not larger than the corresponding component of y, then we write $y \leq\leq x$. If the symmetric matrix $P \in \mathbb{R}^{n\times n}$ is positive-definite, positive-semidefinite, negative-semidefinite, or negative-definite, then we write $P > 0_{n\times n}$, $P \geq 0_{n\times n}$, $P \leq 0_{n\times n}$, and $P < 0_{n\times n}$, respectively. Furthermore, if both $P \in \mathbb{R}^{n\times n}$ and $Q \in \mathbb{R}^{n\times n}$ are symmetric and $(P - Q) > 0_{n\times n}$, $(P - Q) \geq 0_{n\times n}$, $(P - Q) \leq 0_{n\times n}$, and $(P - Q) < 0_{n\times n}$, then we write $P > Q$, $P \geq Q$, $P \leq Q$, and $P < Q$, respectively.

The *natural logarithm of* $x > 0$ is denoted by $\log(x)$, and the *saturation function* $\mathrm{sat} : \mathbb{R} \to [-1, 1]$ is defined so that if $x \in [-1, 1]$, then $\mathrm{sat}(x) = x$, if $x > 1$, then $\mathrm{sat}(x) = 1$, and if $x < -1$, then $\mathrm{sat}(x) = -1$.

The proposed guidance algorithm allows to generate reference trajectories for a UAV, whose degree of cautiousness can be imposed by tuning the user-defined parameters $\mu_q \in \mathbb{R}$, $q \in \{1, \ldots, 9\}$. Additional user-defined parameters, which do not affect directly the UAV's degree of cautiousness, are denoted by $\nu_s \in \mathbb{R}$, $s \in \{1, \ldots, 4\}$.

4 Guidance System's Architecture

In the following, we describe in detail both the path planning and the trajectory planning subsystems underlying the proposed tactical guidance system.

4.1 Tactical Path Planning Subsystem

Consider the orthonormal inertial reference frame $\mathbb{I} \triangleq \{O; X, Y, Z\}$ centered at O and with axes $X, Y, Z \in \mathbb{R}^3$. The path planning algorithm underlying the proposed guidance system generates the *UAV's reference path*, a sequence of waypoints $\{\hat{r}_k\}_{k=0}^{n_\mathrm{p}} \subset \mathbb{R}^3 \setminus \mathcal{O}$ for the UAV expressed in the reference frame $\mathbb{I}$, where $\hat{r}_0$ denotes the UAV's position at the beginning of each iteration of the path planning algorithm, $\hat{r}_{n_\mathrm{p}}$ denotes a point of the goal set $\mathcal{G}$, and $\mathcal{O} \subset \mathbb{R}^3$ denotes the *obstacles' set*. The obstacles' set $\mathcal{O}$ is given by the union of those voxels in the occupancy map produced by the navigation system, whose probability of being occupied is larger than a user-defined threshold value. The integer $k \in \{0, \ldots, n_\mathrm{p}\}$ is solely employed as an index to denote a generic waypoint and to express functional dependencies on waypoints.

The UAV's reference path is generated as the solution of an optimization problem by applying the A^* algorithm [49, App. C]. Specifically, assuming that the occupancy map is partitioned into cubic voxels, and assuming that the UAV is able to move to any unoccupied voxel that surrounds the

one currently occupied by the aircraft, the UAV's reference path is given by a sequence of adjacent voxels that connects the UAV's initial position to the goal set $\mathcal{G}$ and minimizes a user-defined cost function.

The cost function underlying the path planning algorithm is given by

$$f_k \triangleq g_k + h_k, \tag{1}$$

where

$$g_k \triangleq \sum_{q=1}^{k} \left[\kappa(\mathrm{d}_2(\hat{r}_q, \mathcal{O}))\mathrm{d}_2(\hat{r}_q, \hat{r}_{q-1})\right] \tag{2}$$

denotes the *cost-to-come function*,

$$h_k \triangleq (1-\mu_2)\mathrm{d}_2(\hat{r}_k, \mathcal{G}) \tag{3}$$

denotes the *heuristic function*,

$$\kappa(\alpha) \triangleq 1 - \mu_2 e^{4\mu_1\mu_3 - \left[\mu_3\alpha + \mu_1\alpha^{-1}\right]^2}, \qquad \alpha > 0, \tag{4}$$

denotes the *weighing function*, and $\mu_1, \mu_3 > 0$ and $\mu_2 \in [0, 1)$ are user-defined parameters. In practice, Eq. 1 is the sum of the weighted distance traveled by the UAV, namely Eq. 2, and an under-estimate of the Euclidean distance between the voxel occupied by the UAV and the goal set, namely Eq. 3. The role of the weighing function $\kappa(\cdot)$ in Eq. 2 is to encourage tactical behaviors by rewarding paths that are closer to $\mathcal{O}$. If a tactical behavior is desired, then the A^* algorithm is encouraged to search reference paths $\{\hat{r}_k\}_{k=0}^{n_\mathrm{p}}$ that reach the goal set $\mathcal{G}$ among unoccupied voxels that are closer to the obstacles' set $\mathcal{O}$.

As illustrated by Fig. 1, $\kappa(\cdot)$ is such that $\min_{\alpha\in\mathbb{R}} \kappa(\alpha) = 1 - \mu_2$. Thus, it follows from Eq. 2 that for smaller values of μ_2, the attractive effect of the obstacles' set is less enhanced, and the UAV exhibits a more reckless behavior. Since $1-\mu_2$ scales the distance between the UAV's position and the goal set $\mathcal{G}$ in Eq. 3, and $\kappa(\alpha) \geq 1 - \mu_2$ for all $\alpha > 0$, it follows from the triangle inequality that the heuristic function h_k is both admissible and consistent and hence, the proposed path planning subsystem guarantees optimality of the reference path and does not search voxels that were already visited in previous iterations of the A^* algorithm. Moreover, for smaller values of μ_2, the proposed search algorithm investigates a smaller number of voxels and hence, is faster.

The weighing function $\kappa(\cdot)$ is such that $\kappa(\alpha) \in (0, 1]$, $\alpha > 0$, $\lim_{\alpha\to 0^+} \kappa(\alpha) = 1$, and $\lim_{\alpha\to\infty} \kappa(\alpha) = 1$. Thus, if the UAV occupies a cell that is either arbitrarily close or arbitrarily far from the obstacles' set, then the weighing function tends to unity, and the cost-to-come function reduces to the cost-to-come function employed in classical A^*-based path planning algorithms. This property of $\kappa(\cdot)$ allows the user to impose that the UAV's reference path does not coast the obstacles' set too closely, and reduces the attraction exerted by those subsets of $\mathcal{O}$ that are arbitrarily far from the goal set $\mathcal{G}$.

We also note that $\arg\min_{\alpha>0} \kappa(\alpha) = \sqrt{\mu_1\mu_3^{-1}}$, and $\kappa(\cdot)$ is twice continuously differentiable and strictly convex for all $\alpha \in \mathring{\mathcal{I}}$, where $\mathcal{I} \triangleq \{\alpha > 0 : \ddot{\kappa}(\alpha) \geq 0\}$; an expression of $\ddot{\kappa}(\cdot)$ is omitted for brevity. Thus, the boundaries of the compact set $\mathcal{I}$ are given by the two inflection points of $\kappa(\cdot)$ on $(0, \infty)$, namely $\alpha_{\min}, \alpha_{\max} \in \partial\mathcal{I}$. Furthermore, the diameter of $\mathcal{I}$, that is, $|\alpha_{\max} - \alpha_{\min}|$, is finite and increases by decreasing μ_3. Therefore, for smaller values of $\sqrt{\mu_1\mu_3^{-1}}$ and for larger values of μ_3, the proposed search algorithm produces reference paths that are closer to the obstacles' set $\mathcal{O}$ and hence, more cautious. In practice, the diameter of $\mathcal{I}$ and the minimizer of $\kappa(\cdot)$ on $(0, \infty)$ are measures of the extent of the region of influence of the obstacles' set $\mathcal{O}$ on the UAV's reference path. The role of μ_1, μ_2, μ_3

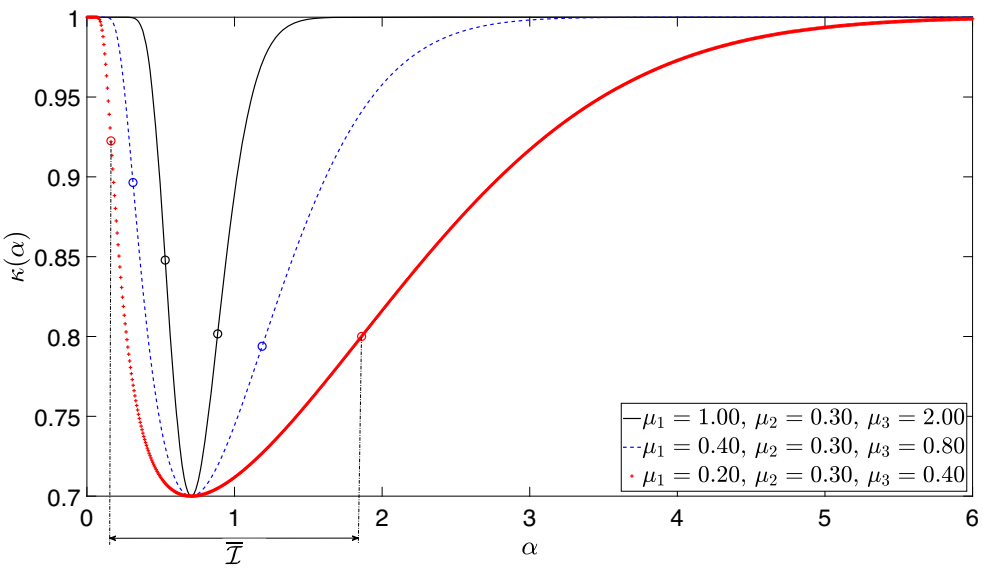

Fig. 1 Plot of the weighing function $\kappa(\cdot)$ for $\mu_2 = 0.3$ and multiple values of μ_1, and μ_3. Since both μ_2 and μ_1/μ_3 are constant, all curves share both the same minimum and the same minimizer. The inflection points of each curve are marked by a circle. The distance between the two inflection points of the scaling function is finite and increases by decreasing the user-defined parameter μ_3

and recommendations for their tuning to obtain reckless or tactical behaviors are summarized in Table 1.

Collision avoidance is enforced by the underlying A^* algorithm. Pathological cases, wherein both the goal set and the obstacles' set attract the UAV in equal manner, and the aircraft is unable to reach $\mathcal{G}$ can not occur. Indeed, the cost function (1) is positive, and the weighing function (4) does not depend on the UAV's distance from the goal set.

To illustrate the applicability of the proposed path planner for tactical UAVs, two sets of numerical simulations have been performed considering multiple values of the user-defined parameter μ_2, while setting $\mu_1 = \mu_3 = 0.20$. The outcomes of these simulations are shown in Fig. 2. The voxel map represented in Fig. 2 captures both a high-bay area, where the quadcopter takes off, and some office space, where the quadcopter is tasked to land; this voxel map has been obtained by employing the same vision-based navigation system as for the flight tests presented in Section 8 below. Table 2 shows both the lengths and average distances of the reference paths from the obstacles' set. As it appears from Fig. 2 and Table 2, and as anticipated in Table 1, larger values of the user-defined parameter μ_2 produce reference trajectories that are longer and closer to the obstacles' set, and hence, more tactical.

4.2 Tactical Trajectory Planning Subsystem

4.2.1 Overview

The path planning algorithm outlined in Section 4.1 does not account for the UAV's dynamics, does not parameterize the reference path as a function of time, and can not enable tactical strategies that require regulating the UAV's velocity. To overcome these limitations, reference trajectories are computed as solutions of an optimal control problem solved numerically by means of a fast model predictive control algorithm, for which the reference path serves as a sequence of waypoints to interpolate.

This section is organized as follows. Section 4.2.2 presents the notation used to describe the proposed trajectory planning system. Section 4.2.3 describes in detail the cost function proposed to instill a tactical behavior in the UAV by coasting the obstacles' set. Section 4.2.4 presents the UAV's dynamical model and discusses how the UAV's velocity at each waypoint can be chosen to enable tactical behaviors. Section 4.2.5 presents how constraints on the UAV's yaw angle, constraints on the controllers' saturation, and collision avoidance constraints can be captured by convex constraint sets. Section 4.2.6 shows how the structure of the given optimal control problem can be exploited to compute fast numerical solutions according to the model predictive control methodology. Finally, Section 4.2.7 shows how soft constraints can be introduced to anticipate hard constraints without disrupting the block-tridiagonal structure of the matrices underlying the proposed trajectory planning system.

4.2.2 Notation

Time is denoted by $t \in \mathbb{R}$, and we assume that the UAV is able to fly from $\hat{r}_k$, $k \in \{0, \ldots, n_\mathrm{p} - 1\}$, to $\hat{r}_{k+1}$ in $n_\mathrm{t}\Delta T$ time units, where both $n_\mathrm{t} \in \mathbb{N}$ and $\Delta T > 0$ are user-defined. In general, both n_t and ΔT are different for each pair of consecutive waypoints. However, the functional dependency of these quantities on k is omitted for simplicity of exposition.

The *UAV's position* is captured by $r_k : [0, n_\mathrm{t}\Delta T] \to \mathbb{R}^3 \setminus \mathcal{O}$ in the inertial reference frame $\mathbb{I}$, where $r_k(0) = \hat{r}_k$ and $r_k(n_\mathrm{t}\Delta T) = \hat{r}_{k+1}$. Employing a 3-2-1 rotation sequence of intrinsic Tait-Bryan angles [46, Ch. 1], the *UAV's roll angle* is denoted by $\phi_k : [0, n_\mathrm{t}\Delta T] \to [0, 2\pi)$, the *UAV's pitch angle* is denoted by $\theta_k : [0, n_\mathrm{t}\Delta T] \to \left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$, the *UAV's yaw angle* is denoted by $\psi_k : [0, n_\mathrm{t}\Delta T] \to [0, 2\pi)$, the *UAV's velocity* with respect to $\mathbb{I}$ is denoted by $v_k : [0, n_\mathrm{t}\Delta T] \to \mathbb{R}^3$, and the UAV's *angular velocity* with respect to $\mathbb{I}$ is denoted by $\omega_k : [0, n_\mathrm{t}\Delta T] \to \mathbb{R}^3$. The *total thrust force* produced by the UAV's propellers is denoted by $u_{1,k}(\cdot)$, the *roll moment* produced by the UAV's propellers is denoted by $u_{2,k}(\cdot)$, the *pitch moment* produced by the UAV's propellers is denoted by $u_{3,k}(\cdot)$, and the *yaw moment* produced by the UAV's propellers is denoted by $u_{4,k}(\cdot)$.

The UAV's trajectory planning algorithm generates the *UAV's reference state vector* $x_k(j\Delta T) \triangleq \big[r_k^\mathrm{T}(j\Delta T), \phi(j\Delta T), \theta(j\Delta T), \psi(j\Delta T), v_k^\mathrm{T}(j\Delta T),$

Table 1 Summary of the user-defined parameters of the proposed path planning system, and recommended values to instill reckless or tactical behaviors

Parameter	Description	Reckless domain	Tactical domain	Equation number
μ_1	Varies the position of both the left inflection point and the minimum of $\kappa(\cdot)$	$(1, \infty)$	$(0, 1]$	Eq. 4
μ_2	Varies both the weight of the heuristic function h_k and the minimum of $\kappa(\cdot)$	$[0, 0.5]$	$(0.5, 1)$	Eq. 3, Eq. 4
μ_3	Varies the position of both the right inflection point and the minimum of of $\kappa(\cdot)$	$(0.7, \infty)$	$(0, 0.7]$	Eq. 4

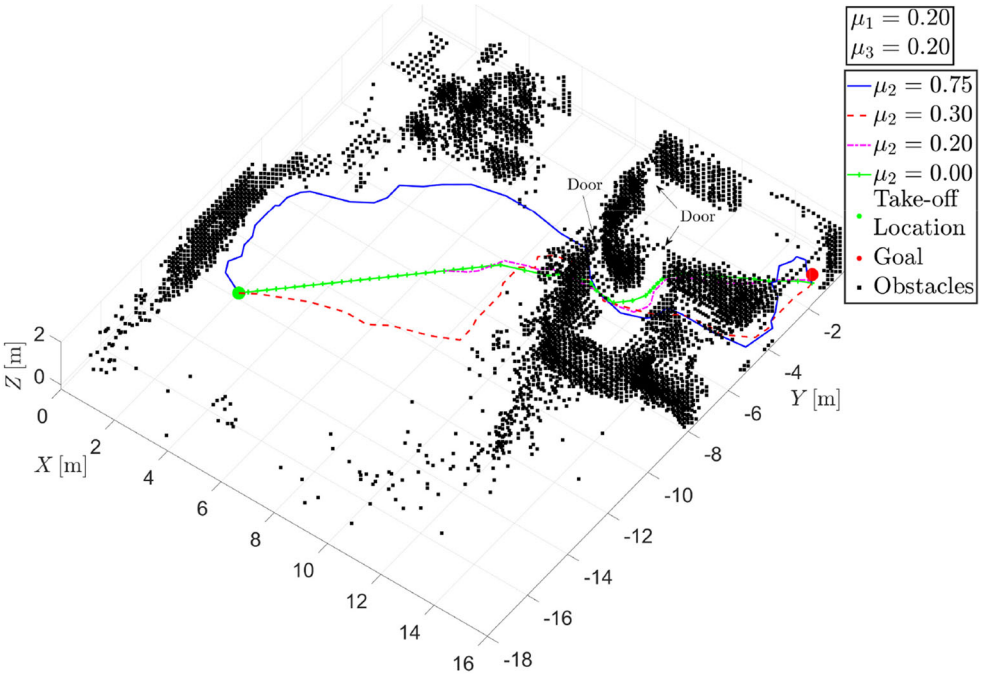

Fig. 2 Reference paths obtained by finding minimizers of the cost function (1) through a numerical simulation. These paths have been generated by setting $\mu_1 = \mu_3 = 0.20$ in Eq. 4 and varying μ_2. The voxel map captures a high-bay area, where the quadcopter takes off, and some office space, where the quadcopter is tasked to land. Larger values of $\mu_2 \in [0, 1)$ induce a more cautious behavior by coasting the obstacles' set more closely. If $\mu_2 = 0.00$, then the UAV follows a reckless path by traversing an open space in the high-bay area

$\omega_k^{\mathrm{T}}(j\Delta T)\big]^{\mathrm{T}}$, $j \in \{i, \ldots, n_{\mathrm{t}}\}$, $i \in \{0, \ldots, n_{\mathrm{t}}\}$, and the corresponding *control input* $u_k(j\Delta T) \triangleq [u_{1,k}(j\Delta T), u_{2,k}(j\Delta T), u_{3,k}(j\Delta T), u_{4,k}(j\Delta T)] \in \mathbb{R}^4$. In particular the pair $(x_k(\cdot), u_k(\cdot))$, is calculated applying the model predictive control algorithm and hence, is recomputed at each time step $j\Delta T$, starting from current time step $i\Delta T$. The integer $i \in \{0, \ldots, n_{\mathrm{t}}\}$ is solely employed to count iterations of the model predictive control algorithm for a given pair of waypoints, and $j \in \{i, \ldots, n_{\mathrm{t}}\}$ is solely employed to indicate the time step $j\Delta T$ within the interval $[i\Delta T, n_{\mathrm{t}}\Delta T]$.

4.2.3 Cost Function Definition

In this paper, the need to outline tactical reference trajectories for UAVs by coasting the obstacles' set is captured by the cost function

$$\tilde{J}[\hat{r}_k, u_k(\cdot)] \triangleq \ell_{\mathrm{f}}(\tilde{r}_k(n_{\mathrm{t}}\Delta T)) + \sum_{i=0}^{n_{\mathrm{t}}-1} \tilde{\ell}(r_k(i\Delta T), u_k(i\Delta T)), \quad (5)$$

Table 2 Length of the reference paths shown in Fig. 2 and their average distance from the obstacles' set $\mathcal{O}$

μ_2	Path length	Average distance from $\mathcal{O}$
0.00	18.54m	2.83m
0.20	19.08m	2.50m
0.30	23.46m	2.29m
0.75	27.26m	1.61m

Smaller values of μ_2 produce shorter reference paths that, on average, are further from the obstacles' set and hence, less cautious

where

$$\tilde{\ell}(\tilde{r}_k, u_k) \triangleq \begin{bmatrix} \tilde{r}_k \\ u_k \end{bmatrix}^{\mathrm{T}} \tilde{R} \begin{bmatrix} \tilde{r}_k \\ u_k \end{bmatrix} + \tilde{q}_r^{\mathrm{T}} \tilde{r}_k + \tilde{q}_u^{\mathrm{T}} u_k, \quad (6)$$

$$\ell_{\mathrm{f}}(r_k) \triangleq \left(r_k - \hat{r}_{k+1}\right)^{\mathrm{T}} R_{r,\mathrm{f}} \left(r_k - \hat{r}_{k+1}\right) + q_{r,\mathrm{f}}^{\mathrm{T}} \left(r_k - \hat{r}_{k+1}\right), \quad (7)$$

$\tilde{R} \triangleq \begin{bmatrix} \tilde{R}_r & \tilde{R}_{r,u} \\ \tilde{R}_{r,u}^{\mathrm{T}} & R_u \end{bmatrix}$, $\tilde{R}_r \in \mathbb{R}^{3\times 3}$ is symmetric, $\tilde{R}_{r,u} \in \mathbb{R}^{3\times 4}$, and $R_u \in \mathbb{R}^{4\times 4}$ are user-defined and such that R_u is positive-definite and

$$\tilde{R}_r - 2\tilde{R}_{r,u}^{\mathrm{T}} R_u^{-1} \tilde{R}_{r,u} > 0, \quad (8)$$

$R_{r,\mathrm{f}} \in \mathbb{R}^{3\times 3}$ is symmetric and positive-semidefinite, $\tilde{q}_r \in \mathbb{R}^3$, $q_{r,\mathrm{f}} \in \mathbb{R}^3$, and $\tilde{q}_u \in \mathbb{R}^4$ are user-defined,

$$\tilde{r}_k(i\Delta T) \triangleq \mu_4 \left[r_k(i\Delta T) - \hat{r}_{k+1}\right] + (1 - \mu_4) \times f_{\mathrm{sat}}\left(\mu_5(\hat{r}_k - r_{\mathcal{O}})\right) \left[r_k(i\Delta T) - r_{\mathcal{O}}\right], \quad (9)$$

$\mu_4 \in (0, 1]$ and $\mu_5 > 0$ are user-defined, $f_{\mathrm{sat}}(w) \triangleq \dfrac{\mathrm{sat}\,(\|w\|)}{\|w\|}$, $w \in \mathbb{R}^n$, and $r_{\mathcal{O}} \triangleq \mathrm{d}_2(\hat{r}_k, \mathcal{O})$. It is worthwhile to note that $f_{\mathrm{sat}}(w)$ is such that $f_{\mathrm{sat}}(w) \to 1$ as $\|w\| \to 0$. Furthermore, it follows from Eq. 8 and the Schur complement condition on the positive-definiteness of block-matrices [13, pp. 7-8] that $\tilde{R}$ is symmetric and positive-definite.

The cost function (5) captures the objectives of reaching the next waypoint, namely $\hat{r}_{k+1}$, and, if a tactical behavior is desired, coasting the obstacles' set $\mathcal{O}$. Specifically, the

Mayer's term (7) captures the UAV's need to reach $\hat{r}_{k+1}$ from the current position r_k; the weighting terms in Eq. 7 are denoted by the subscript f. The first term on the right-hand side of Eq. 9 captures the UAV's distance from the next waypoint, and the second term on the right-hand side of Eq. 9 captures the UAV's distance from the obstacles' set $\mathcal{O}$. Therefore, the *Lagrangian function* (6) captures the UAV's competing needs of reaching the next waypoint and coasting the obstacles' set.

By setting $\mu_4 = 1$, it follows from Eq. 9 that minimizing (5) induces a reckless behavior, since the UAV's sole goal is to reach the next waypoint, whereas decreasing μ_4 induces a more tactical behavior since coasting the obstacles' set becomes a higher priority. Since $f_{\text{sat}}(\mu_5 w) = 1$ for all $w \in \mathbb{R}^n$ such that $\|w\| \leq \mu_5^{-1}$, $f_{\text{sat}}(\mu_5 w) < 1$ for all w such that $\|w\| > \mu_5^{-1}$, and $\lim_{\|w\|\to\infty} f_{\text{sat}}(\mu_5 w) = 0$, the function $f_{\text{sat}}(\cdot)$ in Eq. 9 reduces the attractive effect of obstacles at a distance from the waypoint $\hat{r}_k$ that is larger than μ_5^{-1}. Therefore, smaller values of μ_5 induce a more tactical behavior.

The user-defined matrix $R_{r,\text{f}}$ in Eq. 7 weighs the relative importance of minimizing some components of $\left(r_k - \hat{r}_{k+1}\right)$ over others. For instance, if $R_{r,\text{f}}$ is diagonal and the first two diagonal elements are larger than the third one, then reaching the same altitude as the waypoint $\hat{r}_{k+1}$ is less relevant than reaching the same location as $\hat{r}_{k+1}$ in the horizontal plane. The user-defined vector $q_{r,\text{f}}$ in Eq. 7 can be designed to instill a desired behavior. For instance, by setting $q_{r,\text{f}} = \frac{\hat{r}_k - \hat{r}_{k+1}}{\|\hat{r}_k - \hat{r}_{k+1}\|^2}$ the UAV will be less attracted by $\hat{r}_{k+1}$ as this waypoint is being approached, and this behavior will allow the onboard navigation system more time to detect new features of the environment. The user-defined matrix $\tilde{R}$ in Eq. 6 weighs the relative importance of minimizing some components of the state vector $\tilde{r}_k$ over some components of the control input u_k. The user-defined vectors q_r and q_u in Eq. 6 can be designed to instill a desired behavior. For instance, by setting $\tilde{q}_r = [0, 0, 1]^{\text{T}}$, the UAV will be further drawn toward the ground or the ceiling of an indoor space, which can be considered as tactical behaviors since detection devices are usually employed to point toward open passages, and not impenetrable features such as walls or floors. Similarly, by setting $\tilde{q}_u = [1, 0, 0, 0]^{\text{T}}$, variations in the first element of u_k, namely the thrust force produced by all propellers, are penalized.

Next, we recast (5) as an explicit function of both the UAV's position $r_k(\cdot)$ and the control input $u_k(\cdot)$. To this goal, substituting (9) in Eq. 6, we note that minimizing (5) is equivalent to minimizing

$$\ell_{\text{f}}(r_k(n_{\text{t}}\Delta T)) + \sum_{i=0}^{n_{\text{t}}-1} \ell(r_k(i\Delta T), u_k(i\Delta T)), \tag{10}$$

where

$$\ell(r_k, u_k) \triangleq \begin{bmatrix} r_k \\ u_k \end{bmatrix}^{\text{T}} R_k \begin{bmatrix} r_k \\ u_k \end{bmatrix} + q_{r,k}^{\text{T}} r_k + q_{u,k}^{\text{T}} u_k, \tag{11}$$

$$R_k \triangleq \begin{bmatrix} R_{r,k} & R_{r,u,k} \\ R_{r,u,k}^{\text{T}} & R_u \end{bmatrix}, \tag{12}$$

$$R_{r,k} \triangleq \left[1 + \mu_4(1 - f_{\text{sat}}\left(\mu_5(\hat{r}_k - r_{\mathcal{O}})\right))\right] \tilde{R}_r, \tag{13}$$

$$R_{r,u,k} \triangleq \left[1 + \mu_4(1 - f_{\text{sat}}\left(\mu_5(\hat{r}_k - r_{\mathcal{O}})\right))\right] \tilde{R}_{r,u}, \tag{14}$$

$$\begin{aligned} q_{r,k} \triangleq\ & \left[1 + \mu_4(1 - f_{\text{sat}}\left(\mu_5(\hat{r}_k - r_{\mathcal{O}})\right))\right] \\ & \cdot \Big[\tilde{q}_r - 2\tilde{R}_r \left(\mu_4 \hat{r}_{k+1} + (1 - \mu_4)\right. \\ & \left. f_{\text{sat}} \times \left(\mu_5(\hat{r}_k - r_{\mathcal{O}})\right) r_{\mathcal{O}}\right) \Big], \end{aligned} \tag{15}$$

$$\begin{aligned} q_{u,k} \triangleq\ & \tilde{q}_u - 2\tilde{R}_{r,u}^{\text{T}} \left[\mu_4 \hat{r}_{k+1} + (1 - \mu_4)\right. \\ & \left. f_{\text{sat}} \times \left(\mu_5(\hat{r}_k - r_{\mathcal{O}})\right) r_{\mathcal{O}}\right]. \end{aligned} \tag{16}$$

Note that since $f_{\text{sat}}\left(\mu_5(\hat{r}_k - r_{\mathcal{O}})\right) \in (0, 1]$, it follows from Eq. 8 and the Schur complement condition on the positive-definiteness of block-matrices [13, pp. 7-8] that R_k is positive-definite. Since the proposed trajectory planning system employs a model predictive control framework to minimize (10), the UAV's reference trajectories and the corresponding control inputs are computed iteratively at each time step over the discrete time horizon $\{i\Delta T, \ldots, n_{\text{t}}\Delta T\}$, $i \in \{0, \ldots, n_{\text{t}} - 1\}$, as minimizers of the cost function

$$\begin{aligned} J[\hat{r}_k, u_k(\cdot)] \triangleq\ & \ell_{\text{f}}(r_k(n_{\text{t}}\Delta T)) + \sum_{j=i}^{n_{\text{t}}-1} \ell(r_k(j\Delta T), \\ & u_k(j\Delta T)). \end{aligned} \tag{17}$$

4.2.4 Dynamic Constraints

Let $g > 0$ denote the gravitational acceleration, $m > 0$ the UAV's mass, and $I_x, I_y, I_z > 0$ the UAV's moments of inertia with respect to the roll, pitch, and yaw axes, respectively. The discrete-time, linearized, zero-order hold [93] equations of motion of a quadcopter UAV are given by

$$\begin{aligned} x_k((j+1)\Delta T) &= Ax_k(j\Delta T) + Bu_k(j\Delta T), \\ \begin{bmatrix} r_k(i\Delta T) \\ v_k(i\Delta T) \end{bmatrix} &= \begin{bmatrix} r_{\text{init}} - r_{\text{e}} \\ v_{\text{init}} \end{bmatrix}, \\ \begin{bmatrix} r_k(n_{\text{t}}\Delta T) \\ v_k(n_{\text{t}}\Delta T) \end{bmatrix} &= \begin{bmatrix} \hat{r}_{k+1} - r_{\text{e}} \\ v_{\text{end}} \end{bmatrix}, \\ & j \in \{i, \ldots, n_{\text{t}} - 1\}, \\ & i \in \{0, \ldots, n_{\text{t}} - 1\}, \end{aligned} \tag{18}$$

where $r_{\text{e}} \triangleq [0, 0, h_{\text{e}}]^{\text{T}} \in \mathbb{R}^3$, $h_{\text{e}} \geq 0$ denotes the hover altitude for the UAV at equilibrium, $A = e^{\tilde{A}\Delta T}$, $B =$

$\int_0^{\Delta T} e^{\tilde{A}\sigma} \mathrm{d}\sigma \, \tilde{B}$, and $\tilde{A} \triangleq \begin{bmatrix} 0_{6\times3} & 0_{6\times2} & 0_{6\times1} & \mathbf{1}_6 \\ 0_{2\times3} & \begin{bmatrix} 0 & g \\ -g & 0 \end{bmatrix} & 0_{2\times1} & 0_{2\times6} \\ 0_{4\times3} & 0_{4\times2} & 0_{4\times1} & 0_{4\times6} \end{bmatrix} \in \mathbb{R}^{12\times12}$ and $\tilde{B} \triangleq \begin{bmatrix} 0_{8\times4} \\ \mathrm{diag}(m^{-1}, I_x^{-1}, I_y^{-1}, I_z^{-1}) \end{bmatrix} \in \mathbb{R}^{12\times4}$ capture the continuous-time linearized dynamics of the UAV in a neighborhood of the hover condition.

The algebraic (18) serve as equality constraints for the model predictive control algorithm underlying the proposed trajectory planner. The boundary conditions for Eq. 18 are defined in Table 3; note that since the UAV's state vector comprises 12 elements and Eq. 18 involves 12 boundary conditions on the UAV's position and velocity, the boundary conditions on the UAV's pitch, roll, and yaw angles and their angular velocity are unspecified. Equation 18 can be replaced by the equations of motion of the aircraft output-feedback linearized employing the position vector and yaw angle as measured output. However, since the UAVs considered in this paper do not perform aggressive maneuvers, linear models suffice to describe their dynamics [47].

To enable a tactical behavior, μ_6, μ_7, μ_8, which capture the UAV's velocity at the waypoints $\hat{r}_k$ can be chosen as functions of the UAV's distance from the obstacles' set. For example, μ_6 and μ_7 can be set as

$$|\mu_6 \cos\hat{\psi}_k + \mu_7 \sin\hat{\psi}_k| = \hat{\mu}\,\mathrm{sat}\left(\frac{\overline{\mu}}{f_{\mathrm{sat}}\left(\mu_5(\hat{r}_k - r_{\mathcal{O}})\right)}\right),$$

where $\overline{\mu} \in (0,1)$ and $\hat{\mu} > 0$ are user-defined, and $\mu_6 \cos\hat{\psi}_k + \mu_7 \sin\hat{\psi}_k$ captures the UAV's forward velocity at the waypoint $\hat{r}_k$. According to this strategy, if the UAV is at a waypoint $\hat{r}_k$ that is sufficiently close to the obstacles' set, that is, if $\|\hat{r}_k - r_{\mathcal{O}}\| \in [0, \mu_5^{-1})$, then its forward velocity at the waypoint $\hat{r}_k$ is equal to $\overline{\mu}\hat{\mu}$. If $\|\hat{r}_k - r_{\mathcal{O}}\| \in [\mu_5^{-1}, (\mu_5\overline{\mu})^{-1})$, then the UAV's forward velocity is equal to $\overline{\mu}\hat{\mu}\mu_5\|\hat{r}_k - r_{\mathcal{O}}\|$. Finally, if the UAV is at a waypoint $\hat{r}_k$ that is sufficiently far from any obstacles, that is, if $\|\hat{r}_k - r_{\mathcal{O}}\| \in [(\mu_5\overline{\mu})^{-1}, \infty)$, then its forward velocity is equal to $\hat{\mu}$. Thus, the user-defined parameters μ_5^{-1}, $\overline{\mu}$, and $\hat{\mu}$ can be chosen as follows. By setting μ_5^{-1}, the user defines the maximum distance for the UAV from the obstacles' set to be considered as sheltered, and by setting $(\mu_5\overline{\mu})^{-1}$, the user defines the minimum distance for the UAV from the obstacles' set to be considered as not sheltered. Finally, by setting $\hat{\mu}$, the user defines the UAV's forward velocity at those waypoints that are sufficiently far from any obstacles or, equivalently, by setting $\overline{\mu}\hat{\mu}$, the user defines the UAV's forward velocity at those waypoints that are sufficiently close to the obstacles' set. Strategies for the choice of μ_6, μ_7, μ_8 as functions of the UAV's position with respect to the obstacles' set are numerous, and will be designed in future works by leveraging bio-inspired tactics [35, 113].

4.2.5 Yaw Angle, Saturation, and Collision Avoidance Constraints

To find fast solutions of the proposed trajectory planning problem, constraints on the UAV's state vector and control input are captured by

$$F_k(i\Delta T)\begin{bmatrix} x_k(j\Delta T) \\ u(j\Delta T) \end{bmatrix} \leq\leq f_k(i\Delta T), \tag{19}$$

where $F_k(i\Delta T) \triangleq \begin{bmatrix} F_{r,k}(i\Delta T) & 0_{l\times2} & 0_{l\times1} & 0_{l\times6} & 0_{l\times4} \\ 0_{2\times3} & 0_{2\times2} & F_\psi & 0_{2\times6} & 0_{2\times4} \\ 0_{8\times3} & 0_{8\times2} & 0_{8\times1} & 0_{8\times6} & F_u \end{bmatrix} \in \mathbb{R}^{(l+10)\times16}$, $f_k(i\Delta T) \triangleq \begin{bmatrix} f_{r,k}(i\Delta T) \\ f_{\psi,k}(i\Delta T) \\ f_u \end{bmatrix}$, $F_{r,k}(i\Delta T) \in \mathbb{R}^{l\times3}$, $F_\psi \in \mathbb{R}^2$, $F_u \in \mathbb{R}^{8\times4}$, $f_{r,k}(i\Delta T) \in \mathbb{R}^l$, $f_{\psi,k}(i\Delta T) \in \mathbb{R}^2$, and $f_u \in \mathbb{R}^8$. In the following, we discuss in detail how Eq. 19 is determined.

The optical axes of the UAV's cameras are aligned to the aircraft's roll axis and, consistently with several other results on the guidance of multi-rotor UAVs [63, 110], the reference yaw angle $\psi_k(\cdot)$ is constrained so that the endpoint $\hat{r}_{k+1}$ is

Table 3 Parameters needed to define the boundary conditions for Eq. 18

	$k=0$	$k\in\{1,\ldots,n_\mathrm{p}-2\}$	$k=n_\mathrm{p}-1$
$i=0$	$r_{\mathrm{init}}=\hat{r}_0$,	$r_{\mathrm{init}}=\hat{r}_k$,	$r_{\mathrm{init}}=\hat{r}_{n_\mathrm{p}-1}$,
	$v_{\mathrm{init}}=0$,	$v_{\mathrm{init}}=[\mu_6,\mu_7,\mu_8]^\mathrm{T}$,	$v_{\mathrm{init}}=[\mu_6,\mu_7,\mu_8]^\mathrm{T}$,
	$v_{\mathrm{end}}=[\mu_6,\mu_7,\mu_8]^\mathrm{T}$.	$v_{\mathrm{end}}=[\mu_6,\mu_7,\mu_8]^\mathrm{T}$.	$v_{\mathrm{end}}=0$.
$i\in\{1,\ldots,n_\mathrm{t}-1\}$	$r_{\mathrm{init}}=r_0(i\Delta T)$,	$r_{\mathrm{init}}=r_k(i\Delta T)$,	$r_{\mathrm{init}}=r_{n_\mathrm{p}-1}(i\Delta T)$,
	$v_{\mathrm{init}}=v_0(i\Delta T)$,	$v_{\mathrm{init}}=v_k(i\Delta T)$,	$v_{\mathrm{init}}=v_{n_\mathrm{p}-1}(i\Delta T)$,
	$v_{\mathrm{end}}=[\mu_6,\mu_7,\mu_8]^\mathrm{T}$.	$v_{\mathrm{end}}=[\mu_6,\mu_7,\mu_8]^\mathrm{T}$.	$v_{\mathrm{end}}=0$.

The columns capture the UAV's boundary conditions after departing from the first waypoint, over the course of the mission, and before reaching the last waypoint, respectively. The rows capture the UAV's boundary conditions at the first and the successive iterations of the model predictive control algorithm, respectively

always in the cameras' field of view. This requirement is captured by

$$-\psi_k(j\Delta T) \leq -\hat{\psi}_k(i\Delta T) + \psi_{\max}, \tag{20}$$

$$\psi_k(j\Delta T) \leq \hat{\psi}_k(i\Delta T) + \psi_{\max}, \tag{21}$$

where $\hat{\psi}_k(i\Delta T) \triangleq \tan^{-1}\left(\frac{\mathbf{e}_{2,3}^{\mathrm{T}}\left(\hat{r}_{k+1} - r_k(i\Delta T)\right)}{\mathbf{e}_{1,3}^{\mathrm{T}}\left(\hat{r}_{k+1} - r_k(i\Delta T)\right)}\right)$, $\psi_{\max} > 0$ denotes the *cameras' half field of view*, and $\tan^{-1}\left(\frac{\cdot}{\cdot}\right)$ denotes the signed inverse tangent function. Therefore,

$$F_{\psi} \triangleq [-1, 1]^{\mathrm{T}}, \tag{22}$$

and

$$f_{\psi,k}(i\Delta T) \triangleq \left[\psi_{\max} - \hat{\psi}_k(i\Delta T), \psi_{\max} + \hat{\psi}_k(i\Delta T)\right]^{\mathrm{T}}. \tag{23}$$

The saturation constraints on the control input are captured by

$$-u_k(j\Delta T) \leq\leq u_{\min}, \tag{24}$$

$$u_k(j\Delta T) \leq\leq u_{\max}, \tag{25}$$

where $u_{\min}, u_{\max} \in \mathbb{R}^4$ are user-defined and such that $u_{\max} \geq\geq 0$ and $\mathbf{e}_{1,4}^{\mathrm{T}} u_{\min} \geq 0$. Therefore,

$$F_u \triangleq [-\mathbf{1}_4, \mathbf{1}_4]^{\mathrm{T}}, \tag{26}$$

$$f_u \triangleq [u_{\min}^{\mathrm{T}}, u_{\max}^{\mathrm{T}}]^{\mathrm{T}}. \tag{27}$$

Finally, to enforce collision avoidance constraints on the UAV's reference trajectory, we propose a new algorithm to generate convex, collision-free sets containing the UAV and excluding any obstacle. According to this method, firstly we define the closed ellipsoid

$$\begin{aligned}&\overline{\mathcal{E}}_k(i\Delta T)\\ &\triangleq \Big\{w \in \mathbb{R}^3 : (w - r_k(i\Delta T))^{\mathrm{T}} P_k(i\Delta T)(w - r_k(i\Delta T))\\ &\quad + c_k(i\Delta T) \leq 0\Big\},\end{aligned} \tag{28}$$

where $P_k(i\Delta T) \in \mathbb{R}^{3\times 3}$ and $c_k(i\Delta T) \in \mathbb{R}$ define the shape of $\overline{\mathcal{E}}_k(i\Delta T)$ and a scaling factor for the shape of $\overline{\mathcal{E}}_k(i\Delta T)$, respectively. Both $P_k(\cdot)$ and $c_k(\cdot)$ are computed as solutions of a quadratic discrimination problem, whose cost function is given by

$$\min \mathbf{e}_{8,8}^{\mathrm{T}} b_k(i\Delta T), \tag{29}$$

and whose constraints are given by

$$\begin{aligned}(w_\alpha - r_k(i\Delta T))^{\mathrm{T}} P_k(i\Delta T)(w_\alpha - r_k(i\Delta T))\\ + c_k(i\Delta T) \leq -\gamma_k(i\Delta T),\\ \alpha \in \{1, \ldots, n_{\mathrm{UAV}}\},\end{aligned} \tag{30}$$

$$\begin{aligned}(w_\beta - r_k(i\Delta T))^{\mathrm{T}} P_k(i\Delta T)(w_\beta - r_k(i\Delta T))\\ + c_k(i\Delta T) \geq \gamma_k(i\Delta T),\\ \beta \in \{1, \ldots, n_{\mathcal{O}}\},\end{aligned} \tag{31}$$

$$\gamma_k(i\Delta T) \geq 1, \tag{32}$$

$$P_k(i\Delta T) \geq \mathbf{1}_3, \tag{33}$$

$b_k(i\Delta T) \triangleq [P_{11,k}(i\Delta T), P_{12,k}(i\Delta T), P_{13,k}(i\Delta T), P_{22,k}(i\Delta T), P_{23,k}(i\Delta T), P_{33,k}(i\Delta T), c_k(i\Delta T), \gamma_k(i\Delta T)]$ denotes the state vector of the discrimination problem, $P_{\Delta\epsilon,k}(\cdot)$, $\Delta, \epsilon \in \{1, 2, 3\}$, denotes the element on the Δth row and ϵth column of $P_k(\cdot)$, $w_\alpha \in \mathbb{R}^3$ denotes the αth point used to discretize the UAV, $w_\beta \in \mathbb{R}^3$ denotes the βth point in the obstacles' set $\mathcal{O}$, $\gamma_k(\cdot) \in \mathbb{R}$ denotes a scaling factor, $n_{\mathcal{O}} \in \mathbb{N}$ denotes the number of occupied voxels in the obstacles' set $\mathcal{O}$, and $n_{\mathrm{UAV}} \in \mathbb{N}$ denotes the number of points used to discretize the UAV; in this paper, the quadcopter is captured by a parallelepiped so that $n_{\mathrm{UAV}} = 8$. As a second step in the proposed algorithm to generate convex, collision-free sets, we consider the sampling points $s_1(i\Delta T), \ldots, s_l(i\Delta T) \in \partial\mathcal{E}_k(i\Delta T)$. Finally, we compute the hyperplanes tangent to $\partial\mathcal{E}_k(\cdot)$ at the l sampling points, and capture collision avoidance constraints by setting

$$\begin{aligned}F_{r,k}(i\Delta T) \triangleq &\left[\sum_{q=1}^{l} \mathbf{e}_{q,l} \otimes (s_q(i\Delta T) - r_k(i\Delta T))^{\mathrm{T}}\right]\\ &\cdot P_k(i\Delta T),\end{aligned} \tag{34}$$

$$\begin{aligned}f_{r,k}(i\Delta T) \triangleq &\sum_{q=1}^{l}\Big[\mathbf{e}_{q,l} \otimes (s_q(i\Delta T) - r_k(i\Delta T))^{\mathrm{T}}\\ &\cdot P_k(i\Delta T) s_q(i\Delta T)\Big].\end{aligned} \tag{35}$$

It follows from Eq. 33 that $\overline{\mathcal{E}}_k(i\Delta T)$ is a closed ellipsoid centered at $r_k(i\Delta T)$, it follows from Eqs. 29 and 30 that the UAV is contained in $\mathring{\mathcal{E}}_k(i\Delta T)$, and it follows from Eqs. 29 and 31 that $\partial\mathcal{E}_k(i\Delta T)$ intersects $\mathcal{O}$ in at least one point and that $\mathring{\mathcal{E}}_k(i\Delta T)$ does not intersect the obstacles' set. In practice, $\mathring{\mathcal{E}}_k(\cdot)$ contains the UAV and a sufficiently large number of unoccupied voxels and excludes any occupied voxel. A subset of the l sampling points $s_q(i\Delta T)$, $q \in \{1, \ldots, l\}$, is given by those points, where $\overline{\mathcal{E}}_k(\cdot)$ and $\mathcal{O}$ intersect. If the ellipsoid $\overline{\mathcal{E}}_k(i\Delta T)$, does not contain the waypoint $\hat{r}_{k+1}$, then the boundary conditions to the UAV's equations of motion Eq. 18 can not be met. In this case, to find a feasible reference trajectory, we introduce an additional waypoint, which is given by projecting $\hat{r}_{k+1}$ on $\partial\mathcal{E}_k(i\Delta T)$ along the line that joints $\hat{r}_{k+1}$ to the current UAV's position. Figure 3 illustrates how finding minimizers of Eq. 29 subject to Eqs. 30–33 produces ellipsoids that separate the UAV from the obstacles' set given by an L-shaped hallway.

The efficacy of the proposed algorithm to find convex collision avoidance constraint sets has been measured against the performance of two state-of-the-art algorithms designed for the same purpose, namely the IRIS algorithm [24] and the SFC algorithm [53]. In this comparative analysis, whose results are summarized by Table 4, the proposed algorithm to find convex collision avoidance constraint sets, the IRIS

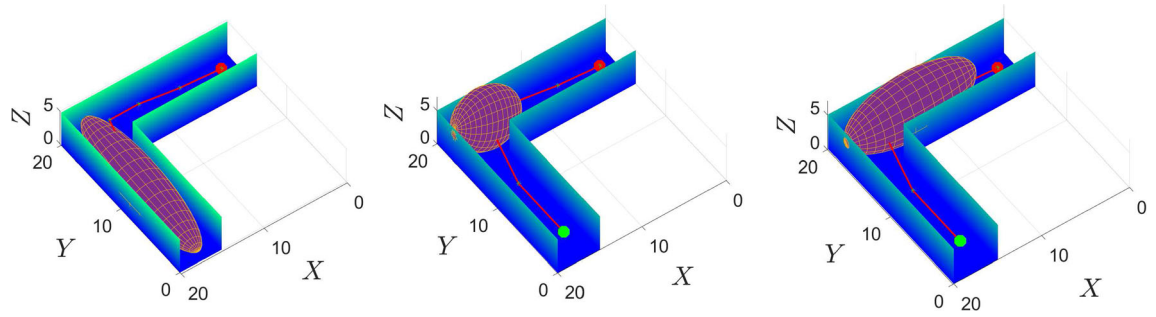

Fig. 3 Graphical representation of the ellipsoid $\overline{\mathcal{E}}_k(\cdot)$ generated as a solution of the quadratic discrimination problem given by Eq. 29 subject to Eqs. 30–33. At each time step, the proposed collision avoidance algorithm produces an ellipsoid that contains the quadcopter in its interior and is tangent to the obstacles' set. The first image shows how the next waypoint, which is depicted as a red dot, may not lie in $\overline{\mathcal{E}}_k(\cdot)$. In this case, a new waypoint is introduced as the projection of the original waypoint on the boundary of $\overline{\mathcal{E}}_k(\cdot)$

algorithm, and the SFC algorithm were executed fifty times on an Intel NUC 7i7DNBE single board computer at 24 waypoints along a predefined path through a winding corridor; for additional details on the single-board computer employed in this paper, see Section 6 below. This corridor, which is which represented in Fig. 4 by 11,430 homogeneously distributed obstacle points, is 5m wide and comprises seven 25m long rectilinear segments. To perform this comparative analysis, the proposed algorithm was coded in C++, and the computer codes for the IRIS algorithm and the SFC algorithm were retrieved from the GitHub repositories [26] and [52], respectively. The SFC algorithm employs a bounding box to reduce the computational time required to search through the obstacles' set $\mathcal{O}$. Therefore, to perform these tests, we assumed that the obstacles within a $5 \times 5 \times 3$m parallelepiped centered in the UAV's position were visible. It appears from Table 4 and Fig. 5 that the proposed approach to generate online convex constraint sets for collision avoidance outperforms both IRIS and the SFC approach in computational time.

4.2.6 Numerical Solution of the Trajectory Planning Problem

The optimal control framework employed in this paper allows to compute the reference state vector $x_k(j\Delta T)$ for each $j \in \{i, \ldots, n_t - 1\}$, $i \in \{0, \ldots, n_t - 1\}$, and for all $k \in \{0, \ldots, n_p - 1\}$, and the corresponding control input $u_k(j\Delta T)$ as minimizers of the cost function Eq. 17 subject to Eqs. 18 and 19. The complexity of the applications considered in this work require the use of a numerical solver, and hence, we employ a model predictive control approach, whereby minimizers of Eq. 17 subject to Eqs. 18 and 19 are computed as solutions of a quadratic programming problem. The cost function Eq. 17 is equivalent to

$$I_{i,k}(z_{i,k}) \triangleq z_{i,k}^{\mathrm{T}} H_{i,k} z_{i,k} + g_{i,k}^{\mathrm{T}} z_{i,k}, \quad z_{i,k} \in \mathbb{R}^{16(n_t - i)}, \tag{36}$$

and the constraints (18) and (19) are equivalent to

$$C_i z_{i,k} = b_{i,k}, \tag{37}$$

$$P_{i,k} z_{i,k} \leq \leq h_{i,k}, \tag{38}$$

respectively, where

$$z_{i,k} \triangleq \left[u_k^{\mathrm{T}}(i\Delta T), \left(\sum_{j=i+1}^{n_t - 1} \mathbf{e}_{j-i,n_t-i} \otimes \left[x_k^{\mathrm{T}}(j\Delta T), u_k^{\mathrm{T}}(j\Delta T) \right]^{\mathrm{T}} \right)^{\mathrm{T}}, x_k^{\mathrm{T}}(n_t \Delta T) \right]^{\mathrm{T}} \in \mathbb{R}^{16(n_t - i)}, \tag{39}$$

$$H_{i,k} \triangleq \mathrm{blockdiag}\left(R_u, \mathbf{1}_{n_t - i - 1} \otimes \hat{R}_k, \mathrm{blockdiag}\left(R_{r,\mathrm{f}}, 0_{9\times 9} \right) \right) \in \mathbb{R}^{16(n_t - i) \times 16(n_t - i)}, \tag{40}$$

$$\hat{R}_k \triangleq \begin{bmatrix} R_{r,k} & 0_{3\times 9} & R_{r,u,k} \\ 0_{9\times 3} & 0_{9\times 9} & 0_{9\times 4} \\ R_{r,u,k}^{\mathrm{T}} & 0_{4\times 9} & R_u \end{bmatrix} \in \mathbb{R}^{16\times 16}, \tag{41}$$

$$g_{i,k} \triangleq \left[q_{u,k}^{\mathrm{T}} + 2 r_k^{\mathrm{T}}(i\Delta T) R_{r,u,k}, \left(\sum_{j=i+1}^{n_t - 1} \mathbf{e}_{j-i,n_t-i} \otimes \left[q_{r,k}^{\mathrm{T}}, 0_{1\times 9}, q_{u,k}^{\mathrm{T}} \right]^{\mathrm{T}} \right)^{\mathrm{T}} \right.,$$

Table 4 Comparative analysis of the computational costs for the proposed algorithm to generate convex collision avoidance constraint sets, IRIS, and SFC

	Proposed approach	IRIS	SFC
Average time [s]	0.2023	0.9694	0.3747
Standard deviation [s]	0.0990	0.1660	0.0320
Median time [s]	0.2008	0.9603	0.3543
Minimum time [s]	0.1975	0.9524	0.3524
Maximum time [s]	0.2703	1.0469	0.3753

Executing each algorithm 50 times along 24 waypoints designed to traverse the winding hallway shown in Fig. 4, the proposed algorithm outperforms both IRIS and SFC

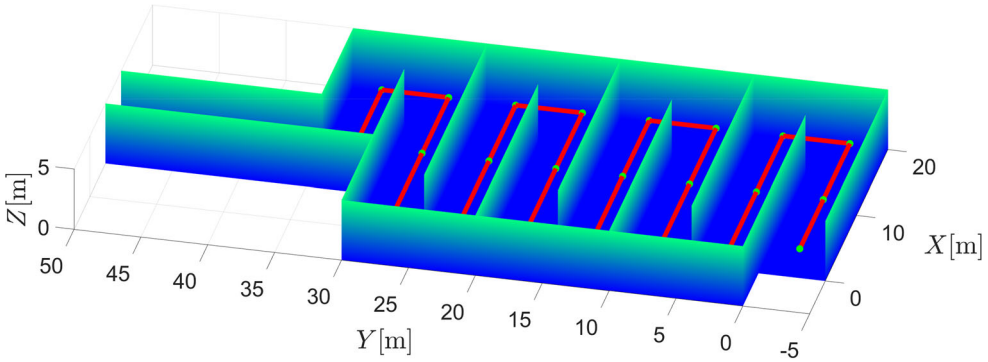

Fig. 4 Winding corridor, waypoints, and reference path used in tests of the proposed algorithm to compute convex, collision-free sets, the SFC algorithm, and the IRIS algorithm

$$q_{r,\mathrm{f}}^{\mathrm{T}} - 2\hat{r}_{k+1}^{\mathrm{T}} R_{r,\mathrm{f}}, 0_{1\times 9}\Big]^{\mathrm{T}} \in \mathbb{R}^{16(n_{\mathrm{t}}-i)}, \tag{42}$$

$$C_i \triangleq \begin{bmatrix} -B & \mathbf{1}_{12} & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & -A & -B & \mathbf{1}_{12} & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & -A & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & \mathbf{1}_{12} & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & -A & -B & \mathbf{1}_{12} \end{bmatrix} \in \mathbb{R}^{12(n_{\mathrm{t}}-i)\times 16(n_{\mathrm{t}}-i)}, \tag{43}$$

$$b_{i,k} \triangleq [x_k^{\mathrm{T}}(i\Delta T)A^{\mathrm{T}}, 0_{1\times 12(n_{\mathrm{t}}-i-1)}]^{\mathrm{T}} \in \mathbb{R}^{12(n_{\mathrm{t}}-i)}, \tag{44}$$

$$P_{i,k} \triangleq \mathrm{blockdiag}\left(F_u, \mathbf{1}_{n_{\mathrm{t}}-i-1} \otimes F_k(i\Delta T), F_{\mathrm{f},k}\right) \in \mathbb{R}^{(l+10)(n_{\mathrm{t}}-i)\times 16(n_{\mathrm{t}}-i)}, \tag{45}$$

$$F_{\mathrm{f},k} \triangleq \begin{bmatrix} F_{r,k}(n_{\mathrm{t}}\Delta T) & 0_{l\times 2} & 0_{l\times 1} & 0_{l\times 6} \\ 0_{2\times 3} & 0_{2\times 2} & F_\psi & 0_{2\times 6} \end{bmatrix} \in \mathbb{R}^{(l+2)\times 12}, \tag{46}$$

$$h_{i,k} \triangleq \left[f_u^{\mathrm{T}}, \left(\sum_{j=i+1}^{n_{\mathrm{t}}-1} \mathbf{e}_{j-i,n_{\mathrm{t}}-i} \otimes f_k(i\Delta T) \right)^{\mathrm{T}}, f_{\mathrm{f},k}^{\mathrm{T}} \right]^{\mathrm{T}} \in \mathbb{R}^{(l+10)(n_{\mathrm{t}}-i)}, \tag{47}$$

$$f_{\mathrm{f},k} \triangleq \left[f_{r,k}^{\mathrm{T}}(n_{\mathrm{t}}\Delta T), f_{\psi,k}^{\mathrm{T}}(n_{\mathrm{t}}\Delta T) \right]^{\mathrm{T}} \in \mathbb{R}^{l+2}; \tag{48}$$

as shown in [100] the equivalence of Eqs. 17–19 and 36–38 can be verified by direct substitution of Eqs. 39–48 in Eqs. 36–38.

It is important to note that the matrices $H_{i,k}$, C_i, and $P_{i,k}$ are block-tridiagonal, and this structure can be exploited by fast tridiagonal solvers such as [4, p. 534], [31], and [109] to find minimizers of Eq. 36 subject to Eqs. 37 and 38. Therefore, the proposed approach to the optimal trajectory planning problem is particularly suitable to compute in real-time reference trajectories for UAVs operating in hostile, unknown environments.

The minimizer $z_{i,k}^* \in \mathbb{R}^{16(n_{\mathrm{t}}-i)}$ of Eq. 36 subject to Eqs. 37 and 38 can be found numerically by applying several techniques such as infeasible interior point method [72], the active set method [61, Ch. 10], and the augmented Lagrangian method [65, Ch. 17]. In this paper, we embed inequality constraints by means of logarithmic barrier functions and implement the infeasible start Newton method to solve the resulting optimization problem [14, Ch. 11]. Specifically, we approximate the problem of minimizing (36) subject to Eqs. 37 and 38 with the problem of minimizing

$$I_{i,\mathrm{lb}}(z_{i,k}) \triangleq z_{i,k}^{\mathrm{T}} H_{i,k} z_{i,k} + g_{i,k}^{\mathrm{T}} z_{i,k} + \nu_1 f_{\mathrm{lb}}(z_{i,k}), \quad z_{i,k} \in \mathbb{R}^{16(n_{\mathrm{t}}-i)}, \tag{49}$$

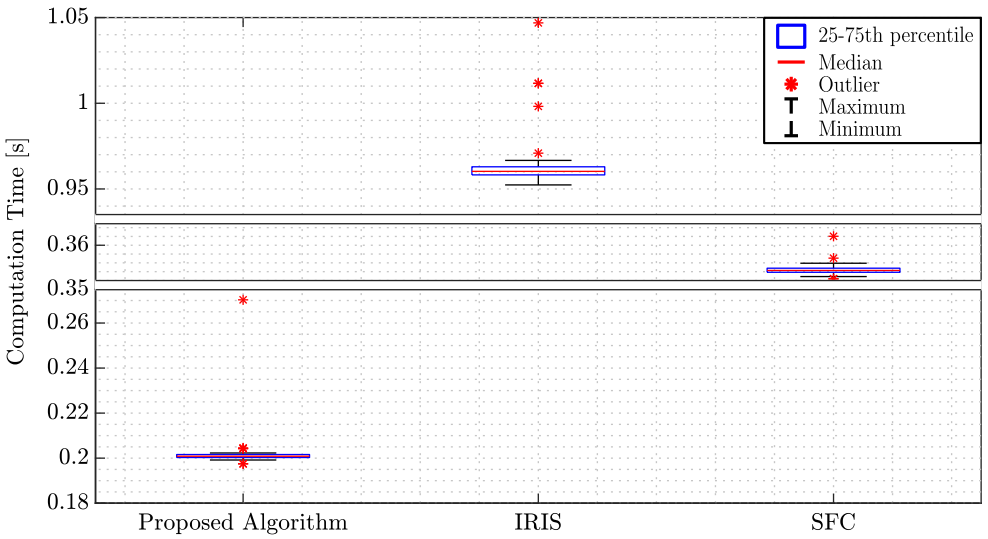

Fig. 5 Boxplot of computational costs for the proposed algorithm to generate convex collision avoidance constraint sets, IRIS, and SFC. The underlying numerical values are presented in Table 4

subject to Eq. 37, where

$$f_{\text{lb}}(z_{i,k}) \triangleq -\sum_{q=1}^{(l+10)(n_{\text{t}}-i)} \log\left(h_{i,k,q} - p_{i,k,q} z_{i,k}\right) \tag{50}$$

denotes the *logarithmic barrier function* associated to Eq. 38, $h_{i,k,q}$ denotes the qth element of $h_{i,k}$, $q \in \{1, \ldots, (l+10)(n_{\text{t}}-i)\}$, $p_{i,k,q}$ denotes the qth row of $P_{i,k}$, and $\nu_1 > 0$ is user-defined and captures the accuracy in approximating the optimal control problem given by Eqs. 36–38 with the optimal control problem given by Eqs. 49 and 37. Successively, we apply the infeasible start Newton method to compute the minimizer of Eq. 49 subject to Eq. 37.

Smaller values of ν_1 guarantee that the minimizer of Eq. 49 subject to Eq. 37 is closer to the minimizer of Eq. 36 subject to Eqs. 37 and 38. However, arbitrarily small values of ν_1 may induce computational errors while evaluating the term $\nu_1 f_{\text{lb}}(z_{i,k})$ in Eq. 49 [14, p. 564]. To overcome this issue, we set $\nu_1 = 2.00$ at the first step of Newton's iterative method and decrease the value of ν_1 at each successive iteration by a factor of 0.9, until $z^*_{i,k}$ meets the desired tolerance $\varepsilon > 0$ [14, p. 569]. Based on an analysis of the duality gap associated with $z^*_{i,k}$ as a function of ν_1, this user-defined parameter can be alternatively set equal to $\varepsilon[(l+10)(n_{\text{t}}-i)]^{-1}$[14, p. 569].

The infeasible start Newton method is initialized by means of a warm start method [14, pp. 531-540, 571], [100]: to compute $z^*_{i,k}$, $i \geq 1$, the first iteration of the Newton method is initiated by $z^*_{i-1,k}$, and subsequent iterations are initiated by the approximation of $z^*_{i,k}$ obtained at the previous iteration. To compute $z^*_{0,k}$, the first iteration of the Newton method is initiated by setting $z_{0,k} = [0, x_k(0), 0, \ldots, x_k(0)]^{\text{T}}$.

4.2.7 Introduction of Soft Constraints

The inequality constraints given by Eq. 38 are hard constraints. If any of the constraints given by Eq. 38 becomes active, then the control input is expected to experience a sudden increase [9, Ch. 4]. In this section, we introduce soft constraints: we modify the cost function Eq. 49 to discourage, but not prevent, the UAV from approaching any of the constraints given by Eq. 38 within user-defined safety margins.

Let $\mu_9 \geq 0$ denote some user-defined safety margin on the UAV's distance from the obstacles' set. It follows from Eqs. 34 and 35 that the collision avoidance constraints are captured by

$$F_{r,k}(i\Delta T) r_k(j\Delta T) \leq\leq \hat{f}_{r,k}(i\Delta T), \tag{51}$$

where

$$\hat{f}_{r,k}(i\Delta T) \triangleq f_{r,k}(i\Delta T) - \mu_9 \textstyle\sum_{q=1}^{l} \mathbf{e}_{q,l}; \tag{52}$$

smaller values of μ_9 will induce a more tactical behavior by inducing the UAV to coast obstacles more closely. Furthermore, let $\nu_2 \geq 0$ denote some user-defined safety margin on the UAV's maximum yaw angle. In this case, it follows from Eqs. 22 and 23 that the camera's pointing requirements are captured by

$$F_\psi \psi(j\Delta T) \leq\leq \hat{f}_{\psi,k}(i\Delta T), \tag{53}$$

where $\hat{f}_{\psi,k}(i\Delta T) \triangleq f_{\psi,k}(i\Delta T) - \nu_2 \sum_{q=1}^{2} \mathbf{e}_{q,2}$. Lastly, let $\nu_3 \in \mathbb{R}^4$, where $\nu_3 \geq\geq 0_4$, denote user-defined margins on the saturation constraints for each component of the control input $u_k(\cdot)$. In this case, it follows from Eqs. 26 and 27 that saturation constraints are captured by

$$F_u u_k(j\Delta T) \leq\leq \hat{f}_u, \tag{54}$$

where $\hat{f}_u \triangleq f_u - \nu_3$. Conditions (51), (53), and (54) can be equivalently expressed as

$$F_k(i\Delta T) \begin{bmatrix} x_k(j\Delta T) \\ u_k(j\Delta T) \end{bmatrix} \leq\leq \hat{f}_k(i\Delta T), \tag{55}$$

where $\hat{f}_k(i\Delta T) \triangleq \left[\hat{f}^{\text{T}}_{r,k}(i\Delta T), \hat{f}^{\text{T}}_{\psi,k}(i\Delta T), \hat{f}^{\text{T}}_u\right]^{\text{T}} \in \mathbb{R}^{l+10}$.

The inequality constraints captured by Eq. 55 can be embedded as soft constraints in the cost function (49) by adding a logarithmic penalty function, namely the Kreisselmeier-Steinhauser penalty function [43]. Specifically, the reference state vector $x_k(j\Delta T)$ for each $j \in \{i, \ldots, n_{\text{t}}\}$, $i \in \{0, \ldots, n_{\text{t}}\}$, and for all $k \in \{0, \ldots, n_{\text{p}} - 1\}$, and the corresponding control input $u_k(j\Delta T)$ are computed as solutions of the optimization problem with cost function

$$\hat{I}_{i,k,\text{lb}}(z_{i,k}) \triangleq I_{i,\text{lb}}(z_{i,k}) + \sum_{q=1}^{(l+10)(n_{\text{t}}-i)} \frac{1}{\nu_{4,i,k,q}} \cdot \log\left(1 + e^{\nu_{4,i,k,q}\left[p_{i,k,q} z_i - \hat{h}_{i,k,q}\right]}\right), \tag{56}$$

subject to Eq. 37, where $\nu_{4,i,k,q}$, $i \in \{0, \ldots, n_{\text{t}} - 1\}$, $k \in \{0, \ldots, n_{\text{p}} - 1\}$, $q \in \{1, \ldots, (l+10)(n_{\text{t}}-i)\}$, is user-defined, and is chosen according to the following procedure outlined in [77]. If there exists $d_{i,k} \in \mathbb{R}^{16(n_{\text{t}}-i)}$ such that $P_{i,k} d_{i,k} = 0$, $\|d_{i,k}\|_\infty < 1$, and $d_{i,k} >> 0$, then

$$\nu_{4,i,k,q} \triangleq \frac{1}{\hat{h}_{i,k,q}} \log\left[\frac{1}{d_{i,k,q}} - 1\right], \tag{57}$$

where $P_{i,k}$ is given by Eq. 45, $p_{i,k,q}$ denotes the qth row of $P_{i,k}$, $d_{i,k,q}$ denotes the qth element of $d_{i,k}$, $\hat{h}_{i,k,q}$ denotes the qth element of

$$\hat{h}_{i,k} \triangleq \left[\hat{f}^{\text{T}}_u, \left(\sum_{j=i+1}^{n_{\text{t}}-1} \mathbf{e}_{j-i,n_{\text{t}}-i} \otimes \hat{f}_k(i\Delta T)\right)^{\text{T}}, \hat{f}^{\text{T}}_{\text{f},k}\right]^{\text{T}} \in \mathbb{R}^{(l+10)(n_{\text{t}}-i)} \tag{58}$$

and

$$\hat{f}_{\mathrm{f},k} \triangleq \left[\hat{f}_{r,k}^{\mathrm{T}}(n_{\mathrm{t}}\Delta T),\, \hat{f}_{\psi,k}^{\mathrm{T}}(n_{\mathrm{t}}\Delta T)\right]^{\mathrm{T}} \in \mathbb{R}^{l+2}. \tag{59}$$

Alternatively, if there is no $d_{i,k}$ such that $\|d_{i,k}\|_\infty < 1$ and $d_{i,k} >> 0$, then we set $\nu_{4,i,k,q}$ arbitrarily large for all $i \in \{0,\ldots,n_{\mathrm{t}}-1\}$, $k \in \{0,\ldots,n_{\mathrm{p}}-1\}$, and $q \in \{1,\ldots,(l+10)(n_{\mathrm{t}}-i)\}$.

The Kreisselmeier-Steinhauser penalty function is given by the second term in Eq. 56. The use of this penalty function to add soft constraints to the cost function (49) has three main advantages. Firstly, this function does not disrupt the block-tridiagonal structure of the original optimization problem captured by Eq. 49 subject to Eq. 37 [77]. Therefore, proceeding as in Section 4.2.6, minimizers of Eq. 56 subject to Eq. 37 can be found by applying the infeasible start Newton method with warm start. Secondly, the parameter $\nu_{4,i,k,q}$ given by Eq. 57 guarantees that the Kreisselmeier-Steinhauser function attains its minimum for $z_i = 0$ and hence, control inputs that minimize (56) subject to Eq. 37 are not offset compared to the control inputs minimizing the original optimization given by the cost function Eq. 36 subject to Eq. 37 [77, 104]. Lastly, larger values of $\nu_{4,i,k,q}$ guarantee that the minimizer of Eq. 56 subject to Eq. 37 is closer to the minimizer of Eq. 49 subject to Eq. 37. For this reason, by proceeding as in [77], if $\|d_{i,k}\|_\infty < 1$ and $d_{i,k} >> 0$, then at each iteration of Newton's method $d_{i,k}$ in Eq. 57 is rescaled by a factor of 0.4; this scaling factor has been chosen for providing a satisfactory trade-off between accuracy and convergence rate. Alternatively, if there is no $d_{i,k}$ such that $\|d_{i,k}\|_\infty < 1$ and $d_{i,k} >> 0$, then $\nu_{4,i,k,q}$ is given the arbitrary large value of 200. Both Eqs. 58 and 59 directly follow from Eqs. 47 and 48, respectively, by replacing f_u, $f_k(\cdot)$, $f_{\mathrm{f},k}(\cdot)$, $f_{r,k}(\cdot)$, and $f_{\psi,k}(\cdot)$ with $\hat{f}_u$, $\hat{f}_k(\cdot)$, $\hat{f}_{\mathrm{f},k}(\cdot)$, $\hat{f}_{r,k}(\cdot)$, and $\hat{f}_{\psi,k}(\cdot)$, respectively.

The role of the user-defined parameters $\mu_4,\ldots,\mu_9$ and recommendations for their tuning are summarized in Table 5. Together with Table 1, this table summarizes the role of the user-defined parameters needed to induce reckless or tactical behaviors in a UAV equipped with the proposed guidance system.

5 Navigation System's Architecture

The tactical guidance system described in Section 4 is supported by a custom-made vision-based navigation system, which detects obstacles, creates an occupancy map, deduces the obstacles' set from the occupancy map, and provides estimated information on the UAV's state vector. In particular, a stereo depth camera and a tracking camera detect the obstacles surrounding the UAV. To prevent high computational loads due to high data volumes produced by the depth camera, the resolution of the images is down-sampled so that point clouds that are further from the UAV are removed, and point clouds that are closer to the UAV are retained [19, pp. 17-18]. The tracking camera also estimates the UAV's attitude, translational velocity, and angular velocity with respect to an inertial reference frame $\mathbb{I}$. Successively, Bresenham's ray tracing algorithm [15] calculates the UAV's direction and distance with respect to obstacles and, together with an inverse sensor model [5], determines the occupancy along the ray from the sensor to the point cloud. The occupancy map is then updated by calculating the posterior probability distribution, conditioned on past measurements [91, Ch. 9]. Lastly, a binarization algorithm marks all voxels of the occupancy map, whose probability of being occupied is higher than a user-defined threshold, as occupied and all other voxels as empty. The resulting binarized map forms an occupancy map, whose voxels' locations are expressed in the reference frame $\mathbb{I}$. A schematic representation of this navigation system is shown in Fig. 6, and additional details can be found in [19].

In this paper, the voxels which have not been seen by the vision-based navigation system are classified as unoccupied, since the proposed guidance system enables tactical behaviors by coasting obstacles and unexplored areas may be completely unsheltered. This feature ensures that the path planning subsystem computes a path from the take-off position $\hat{r}_0$ to the goal set $\mathcal{G}$ that is complete, also in case $\mathcal{G}$ is not visible from $\hat{r}_0$. Moreover, a voxel retains its last known probability of being occupied once the voxel has left the line of sight of the navigation system. This feature allows faster re-planning whenever the UAV must revisit known areas, whose occupancy map has not changed.

6 Implementation of the Proposed Guidance and Navigation System

The guidance system presented in Section 4 and the navigation systems presented in Section 5 have been implemented on a custom-built quadcopter. This UAV is 0.4m long, 0.4m wide, and 0.3m high, its mass is 2.0kg, and, according to a high-fidelity computer aided design (CAD) model, its principal moments of inertia are $I_x = 0.0205\mathrm{kg{\cdot}m^2}$, $I_y = 0.0143\mathrm{kg{\cdot}m^2}$, and $I_z = 0.0281\mathrm{kg{\cdot}m^2}$. This UAV is equipped with an Intel NUC 7i7DNBE single-board computer, where software implementing both the proposed guidance system and the navigation system is executed; this computer is characterized by a 4.20GHz Intel i7-8650 Processor and a 4GB memory, and executes the Ubuntu 18.04 operating system. The UAV is also equipped with a Pixhawk autopilot that serves as an inertial measurement unit and controls the UAV's propellers, an Intel RealSense D435i that serves as

Table 5 Summary of the user-defined parameters of the proposed trajectory planning system, and recommended values to instill reckless or tactical behaviors

Parameter	Description	Reckless Domain	Tactical Domain	Equation number
μ_4	Relative important of of reaching a waypoint $\hat{r}_k$ over coasting the obstacles' set $\mathcal{O}$	(0.7, 1]	(0, 0.7]	Eq. 9
μ_5	Varies the attractive effect of obstacles	(0, 0.5)	[0.5, ∞)	Eq. 9
μ_6	Velocity boundary condition in the x-direction at the waypoint $\hat{r}_k$			Eq. 18
μ_7	Velocity boundary condition in the y-direction at the waypoint $\hat{r}_k$	$\mathbb{R}$; depends on user's strategy		Eq. 18
μ_8	Velocity boundary condition in the z-direction at the waypoint $\hat{r}_k$			Eq. 18
μ_9	Safety margin on the UAV's collision avoidance induced by soft constraints.	Arbitrarily large & positive	Arbitrarily small & positive	Eq. 52

a stereo depth camera, and an Intel RealSense T265 that serves as a tracking camera. The D435i's horizontal field-of-view angle is 86.00°, its vertical field-of-view angle is 57.00°, its depth is of 9m, and it communicates with the onboard computer via USB-C cable. The T265's horizontal field-of-view angle is 69.40°, its vertical field-of-view angle is 42.50°, and it communicates with the onboard computer via USB Micro B cable. The Pixhawk autopilot communicates with the onboard computer over a dedicated USB FTDI serial line.

Figure 7 provides a schematic representation of the software implementing the proposed guidance and navigation systems. This software revolves around the Flight Stack, which receives information on the UAV's estimated state and the environment's map from the navigation system and the inertial measurement unit, communicates with each of the sub-modules of the guidance system, and passes the UAV's reference trajectory to the Pixhawk autopilot. The estimates on the UAV's position and yaw angle produced by the navigation system are directly passed to the Flight Stack; the navigation system communicates with the Flight Stack at a frequency of approximately 20-60Hz, depending on the objects in view of the cameras. The estimates on the UAV's attitude, translational velocity, and angular velocity produced by the navigation system are passed to the Pixhawk autopilot and integrated with its estimates on the UAV's state by means of a Kalman filter. Finally, estimates on the UAV's pitch angle, roll angle, translational velocity, and angular velocity are passed from the autopilot to the Flight Stack. The estimates of the UAV's attitude, angular velocity, and angular acceleration have an accuracy of approximately $\pm 0.05°$, $\pm 0.004°/s$, and $\pm 7°/s^2$, respectively.

The guidance system's sub-modules of the Flight Stack are the path planner, which implements the search algorithm presented in Section 4.1, the collision avoidance algorithm, which solves the discrimination problem captured by Eq. 29 subject to Eqs. 30–33, and the trajectory planner, which solves the optimization problem captured by the cost function Eq. 56 subject to Eq. 37 according to the fast model predictive control approach presented in Section 4.2.6. The path planner leverages a custom-made implementation of the A^* optimization algorithm in C++, the collision avoidance algorithm exploits a C++ implementation of the semi-definite programming algorithm (SDPA) [27], and the trajectory planner exploits the C++ code presented in [100, 101], customized to implement the results presented in Section 4.2.7.

Alternative approaches to the codes presented in [101] have been considered to solve in real-time the quadratic programming problem presented in Section 4.2.7 at each time step $i\Delta T$. Among these, it is worthwhile to mention CVXGEN [60], ACADO [32], ACADOS [96], and NLopt [36].

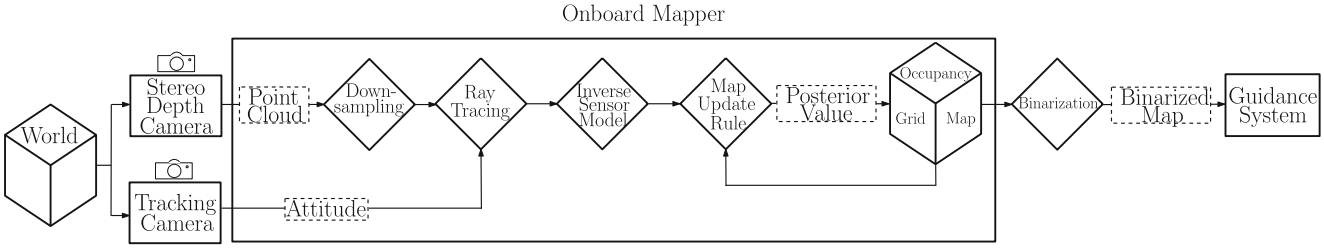

Fig. 6 Schematic representation of the vision-based navigation system employed to detect obstacles, create a voxel map of the environment, and eventually produce the obstacles' set employed by the proposed guidance system

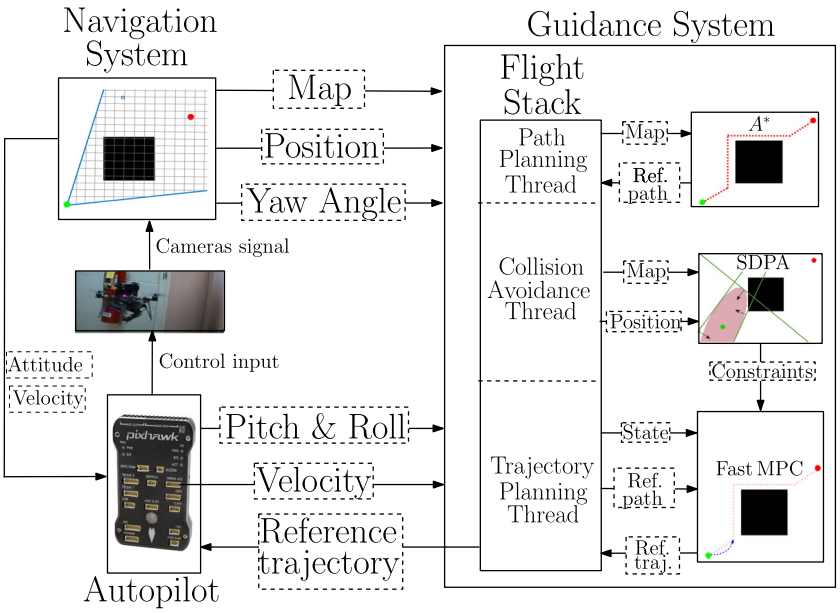

Fig. 7 Schematic representation of the software architecture implementing the proposed guidance system on a quadcopter UAV equipped with a stereo depth camera, a tracking camera, and an autopilot. The vision-based navigation system produces a voxel map of the environment and, together with the inertial measurement unit embedded in the autopilot, estimates the UAV's state. These information are used by the guidance system to produce reference trajectories. The software implementing the proposed guidance system is structured in a path planning thread, a collision avoidance thread, and a trajectory planning thread, which are coordinated by the Flight Stack program. The control law embedded in the autopilot actuates the UAV's propellers so that the UAV follows its reference trajectory

However, unlike the proposed guidance system, CVXGEN, ACADO, and ACADOS do not employ a warm start method. Our trajectory planner, CVXGEN, ACADO, and ACADOS solve optimization problems by means of iterative processes, and a suitable choice of the initial guess for each of these iterations allows to reduce the convergence time. According to the warm start method, fixed the time step $i\Delta T$, the initial guess for each iteration is given by the outcome of the previous iteration at $i\Delta T$. However, if utilized to implement the model predictive control approach, CVXGEN, ACADO, and ACADOS can be initiated at the time step $(i+1)\Delta T$ by employing the solution obtained at the time step $i\Delta T$ [6, pp. 26-31]. Additional reasons for not employing ACADO and ACADOS in this work is that these solvers under-perform with respect to specific solvers for optimization problems with large state vectors, such as those considered in this paper at earlier iterations of the model predictive control algorithm [97]. Finally, the NLopt manual discourages its use for solving quadratic optimization problems.

The path planner produces collision-free reference paths at a rate of approximately 0.5-20Hz, depending on the values of μ_1, μ_2, and μ_3 in Eqs. 3 and 4, and communicates with the Flight Stack at approximately 100Hz. The path planner is re-executed at least every 2s or after the UAV traveled 1.5m, which guarantees that a new path is produced after the aircraft has traveled approximately 3.75 times its size. Feasible reference trajectories are generated at approximately 50Hz. Once a feasible reference trajectory is outlined, it is sent from the Flight Stack to the Pixhawk autopilot at a rate of $(n_t\Delta T)^{-1}$. Similarly to the approach presented in [90], the trajectory planner is executed before the UAV reaches the next waypoint; producing a new reference trajectory only after the aircraft has reached a waypoint may cause unacceptable delays. The SDPA algorithm produces constraint sets to enforce collision avoidance at a frequency of 25Hz circa. A critical aspect for the integration of the path planner, the trajectory planner, and the SDPA algorithm is the synchronization of these modules and their frequency of execution relative to UAV's velocity. In the proposed architecture, setting arbitrarily large values of the user-defined parameters μ_6, μ_7, and μ_8, which characterize the UAV's velocity at the waypoints, without accounting for the speed of the path planner, trajectory planner, and SDPA algorithm may lead to failures in completely unknown environments. Future work directions involve computing μ_6, μ_7, and μ_8 as solutions of a constrained optimization problem that accounts for the user's requirements and the computational constraints given by the sub-modules of the Flight Stack.

The autopilot executes a proportional-integral-derivative controller [47] to actuate the 7×4.5 dual-blade propellers mounted on AirA 1,200kV motors. Since the model predictive control algorithm produces open-loop control inputs,

which are not robust to modeling uncertainties, the linear closed-loop control law provided by the UAV's autopilot has been employed to steer the quadcopter.

7 Numerical Analysis of the Proposed Guidance System

In this section, we present the results of 438 software-in-the-loop simulations to validate the hardware and software architecture presented in Section 6 and provide a taxonomy of flight behaviors deduced by varying some of the user-defined parameters that determine the UAV's cautiousness level, the UAV's initial conditions, and the occupancy map.

7.1 Validation of the Proposed Guidance System through Software-in-the-Loop Simulations

In the following, we present the results of two software-in-the-loop simulations aimed at showing the ability of the proposed guidance system to generate both a tactical trajectory and a reckless trajectory in an unknown environment, while traveling from a given initial position to a goal point, whose position is specified relative to the initial position. Figure 8 shows the results of these simulations performed using the same voxel map as in Fig. 2. In the first simulation, a reckless behavior was produced by setting $\mu_2 = 0.40$, $\mu_4 = 1.00$, and $\mu_5 = 0.01$; this reference trajectory is denoted by a red solid line. In the second simulation, a cautious behavior was produced by setting $\mu_2 = 0.75$, $\mu_4 = 0.80$, and $\mu_5 = 0.50$; this reference trajectory is denoted by a blue dashed line. For both simulations, we set $\mu_1 = 0.20$, $\mu_3 = 0.20$, $\mu_6 = 0.50$, $\mu_7 = 0.50$, $\mu_8 = 0.00$, $\mu_9 = 0.25$, $\Delta T = 0.01$s, $n_t = 80$, $\psi_{\max} = 43.00°$, $\tilde{R}_r = R_{r,\mathrm{f}} = 700 \cdot \mathbf{1}_3$, $\tilde{R}_{r,u} = 0_{3\times4}$, $R_u = \mathrm{blockdiag}(1, 300 \cdot \mathbf{1}_3)$, $\tilde{q}_r = q_{r,\mathrm{f}} = 0_3$ $\tilde{q}_u = 0_4$, $u_{\max} = [1.50, 0.5, 0.5, 0.5]^{\mathrm{T}}$, and $u_{\min} = -[0.0, 0.5, 0.5, 0.5]^{\mathrm{T}}$.

Figure 8 shows the flight times at multiple sample positions along the reckless and tactical reference trajectories; it is apparent that following the more reckless trajectory, the UAV reaches its goal earlier than following the more cautious trajectory. Figure 8 also shows the separating ellipsoid $\overline{\mathcal{E}}_k(i\Delta T)$ centered at $[15.41, -4.72, 1.20]^{\mathrm{T}}$m. Before these numerical simulations were performed, the voxel map has been produced and then memorized on the UAV's single-board computer. Thus, the voxel map was disclosed to the guidance system as if it were produced in the real-time by the onboard Intel RealSense D435i; this camera's field of view is represented in Fig. 8 by a cone, whose apex is in the UAV's current position and whose height is 10m. If a point in the obstacles' set $\mathcal{O}$ laid in the simulated camera's field of view, then it was marked as detected and disclosed to the guidance system for all future time.

Table 6 presents statistical data of both reference trajectories. As predicted in Sections 4.1 and 4.2, larger values of μ_2 and μ_5 and smaller values of μ_4 produce more tactical reference trajectories, whereas smaller values of μ_2 and μ_5 and larger values of μ_4 produce more reckless trajectories.

7.2 Taxonomy of Flight Behaviors Varying Individual Parameters

In this section, we present the results of two sets of software-in-the-loop simulations performed to establish a taxonomy of flight behaviors as a function of each of the user-defined parameters $\mu_1, \ldots, \mu_5$. These five parameters were chosen among the user-defined parameters $\mu_1, \ldots, \mu_9$ for their ability, according to our experience of the proposed guidance system, to affect the UAV's cautiousness level more markedly.

For the first set of simulations, we chose $\mu_1 = 1.00$, $\mu_2 = 0.10$, $\mu_3 = 0.50$, $\mu_4 = 1.00$, and $\mu_5 = 0.10$ to generate a reckless reference trajectory. Then, while keeping four of these five parameters fixed, we varied one parameter at the time both within intervals producing reckless behaviors and within intervals producing tactical behaviors; for details, see Tables 7 and 8. In the second set of simulations, we chose $\mu_1 = 0.10$, $\mu_2 = 0.95$, $\mu_3 = 0.10$, $\mu_4 = 0.80$, and $\mu_5 = 0.50$ to generate a tactical reference trajectory. Then, we varied each of these parameters within the same intervals as for the first set of simulations; for details, see Tables 9 and 10. For both sets of simulations, the UAV was tasked to fly from $r_0 = [1.0, 1.0, 0.6]^{\mathrm{T}}$m to the goal set $\mathcal{G} = \{[1.0, 19.0, 0.6]^{\mathrm{T}}\}$m; the remaining user-defined parameters were the same as in Section 7.1.

Tables 7–10 show the average length of the UAV's reference trajectories, the standard deviation of the length of the UAV's reference trajectories, the average distance from the obstacles' set, and the standard deviation of the distance from the obstacles' set for the 200 numerical simulations performed to support the proposed variational analyses. By comparing the third and the fifth columns of Table 7 with the third and the fifth columns of Table 9, respectively, and the third and fifth columns of Table 8 with the third and fifth columns of Table 10, respectively, it appears that variations of the user-defined parameters $\mu_1, \ldots, \mu_5$ about a reckless parameter set produce smaller variations in the UAV's behavior than variations of the same parameters about a tactical parameter set.

Figure 9 shows the results of 80 numerical simulations obtained by performing the proposed variational analyses as functions of μ_2 and μ_1. From the two images on the left of this figure, we deduce that if four of the five user-defined parameters are designed to impose a reckless behavior, then the UAV's reference trajectories do not deviate ostensibly

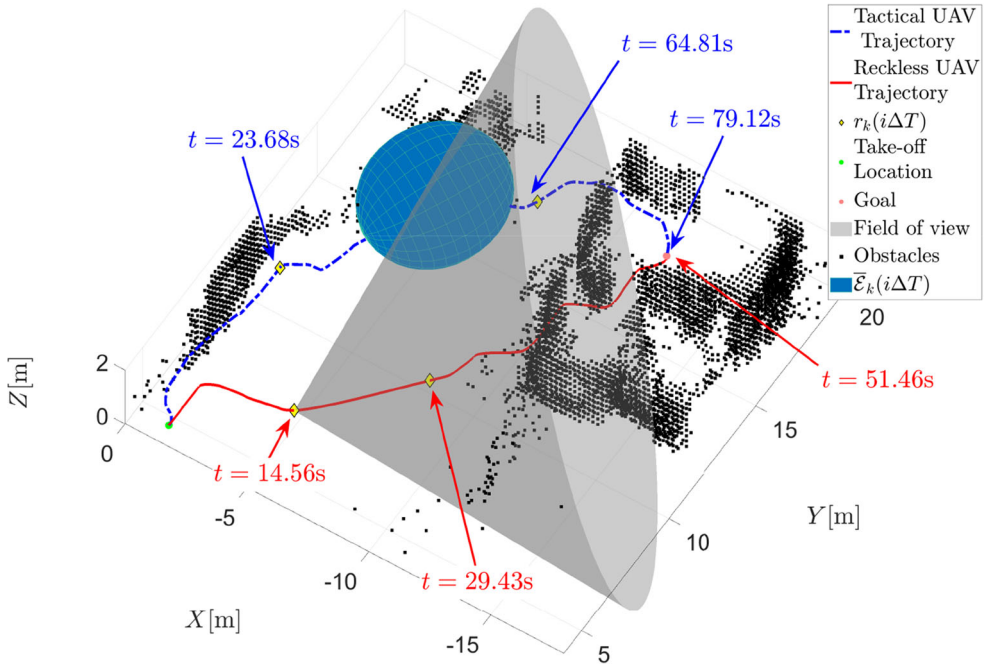

Fig. 8 UAV reference trajectories obtained through software-in-the-loop simulations. Both reference trajectories start at $\hat{r}_0 = [-1.80, 4.00, 0.80]^{\mathrm{T}}$m and end at the goal set $\mathcal{G} = \{[-12.60, 17.40, 1.00]^{\mathrm{T}}\}$m. The reference trajectory obtained by setting $\mu_2 = 0.75$, $\mu_4 = 0.80$, and $\mu_5 = 0.50$ is represented by a blue dashed line and shows a more cautious behavior since it closely coasts the obstacles' set. The reference trajectory obtained by setting $\mu_2 = 0.40$, $\mu_4 = 1.00$, and $\mu_5 = 0.01$ is represented by a red solid line and shows a more reckless behavior since it aims at the end-goal directly. The flight time at multiple locations along the UAV's reference trajectories, the separating ellipsoid $\overline{\mathcal{E}}_k(i\Delta T)$ centered at $[-4.72, 15.41, 1.20]^{\mathrm{T}}$m, and the simulated camera's field-of-view as shown as well

Table 6 Statistical data for the simulations shown in Fig. 8

	Reckless Trajectory	Tactical Trajectory
Trajectory length	19.53m	22.92m
Average distance from $\mathcal{O}$	1.84m	0.57m
Maximum distance from $\mathcal{O}$	3.43m	1.67m
Flight time	51.46s	79.12s

Table 7 Average length of the UAV's reference trajectories, standard deviation of the length of the UAV's reference trajectories, average distance from the obstacles' set, and standard deviation of the distance from the obstacles' set $\mathcal{O}$ when varying $\mu_1, \ldots, \mu_5$ one at the time within ranges that produce reckless behaviors, while the remaining four parameters are fixed and take values that induce reckless behaviors

Varied Parameters	Avg. Traj. Length	Std. Dev. Traj. Length	Avg. Dist. from $\mathcal{O}$	Std. Dev. Dist. from $\mathcal{O}$
$\mu_1 \in [1.00, 2.00]$	29.16m	1.23m	1.56m	0.05m
$\mu_2 \in [0.05, 0.50)$	28.73m	0.645m	3.65m	0.14m
$\mu_3 \in (0.70, 0.95]$	29.92m	1.81m	3.51m	0.10m
$\mu_4 \in [0.70, 1.00]$	29.38m	1.85m	1.70m	0.09m
$\mu_5 \in [0.01, 0.50)$	29.36m	1.53m	3.36m	0.09m

This data set was produced by evenly sampling each parameter's interval 10 times for a total of 50 simulations

Table 8 Average length of the UAV's reference trajectories, standard deviation of the length of the UAV's reference trajectories, average distance from the obstacles' set, and standard deviation of the distance from the obstacles' set when varying $\mu_1, \ldots, \mu_5$ one at the time within ranges that produce tactical behaviors, while the remaining four parameters are fixed and take values that induce reckless behaviors

Varied Parameters	Avg. Traj. Length	Std. Dev. Traj. Length	Avg. Dist. from $\mathcal{O}$	Std. Dev. Dist. from $\mathcal{O}$
$\mu_1 \in (0.00, 1.00]$	29.57m	1.78m	1.71m	0.10m
$\mu_2 \in [0.50, 0.95]$	31.67m	3.30m	3.74m	0.10m
$\mu_3 \in [0.01, 0.70]$	29.06m	0.74m	3.52m	0.06m
$\mu_4 \in [0.05, 0.70)$	28.42m	0.72m	1.64m	0.06m
$\mu_5 \in [0.50, 0.95]$	28.97m	0.68m	3.21m	0.12m

This data set was produced by evenly sampling each parameter's interval 10 times for a total of 50 simulations

from the average reckless trajectory, despite the fact that the fifth parameter takes values that induce a tactical behavior. This analysis is supported by the fact that the values in the second column of Table 7 differ between 1.33% and 9.28% from the corresponding values in Table 8. From the two images on the right of Fig. 9, we deduce that if four of the five user-defined parameters are designed to impose a tactical behavior, then the UAV's reference trajectories are tactical whenever the fifth parameter takes values that would induce a tactical behavior, and the UAV's reference trajectories are reckless whenever the fifth parameter takes values that would induce a reckless behavior. This analysis is supported by the fact that the values in the second column of Table 9 differ between 1.73% and 27.32% from the corresponding values in Table 10.

In conclusion, from Fig. 9 and Tables 7–10, we deduce that if a reckless reference trajectory is to be produced, then the UAV's reference trajectories are clustered around the average reckless trajectory. Thus, the UAV exhibits a more predictable behavior. Conversely, if a tactical reference trajectory is to be produced, then the UAV's reference trajectories are more scattered, and different obstacles are coasted. Thus, the UAV exhibits a less predictable behavior whenever it does not need to traverse narrow passages, such as doors and windows, to reach the goal set: the values in the fourth column of Table 7 are between 0.28% and 8.77% from the corresponding values in Table 8, and the values in the fourth column of Table 9 are between 0.74% and 65.71% from the corresponding values in Table 10.

7.3 Taxonomy of Flight Behaviors Varying Multiple Parameters

In this section, we present the results of 36 software-in-the-loop simulations to establish a taxonomy of flight behaviors as a function of the triplet (μ_1, μ_2, μ_3), which characterizes the path planner, and the pair (μ_4, μ_5), which more markedly characterize the level of cautiousness in the UAV's trajectory planner. The variational analysis presented in Section 7.2 concerned the effect of each of these parameters individually on the UAV's behavior, whereas the analysis presented in the following accounts for the coupling effect of these parameters.

For these simulations, we analyzed the trajectories obtained from by setting $(\mu_1, \mu_2, \mu_3) \times (\mu_4, \mu_5) \in \mathcal{S}_1 \times \mathcal{S}_2$, where $\mathcal{S}_1 \triangleq \{(2.00, 0.10, 1.50), (1.60, 0.20, 1.25), (1.20, 0.30,$

Table 9 Average length of the UAV's reference trajectories, standard deviation of the length of the UAV's reference trajectories, average distance from the obstacles' set, and standard deviation of the distance from the obstacles' set when varying $\mu_1, \ldots, \mu_5$ one at the time within ranges that produce reckless behaviors, while the remaining four parameters are fixed and take values that induce tactical behaviors

Varied Parameters	Avg. Traj. Length	Std. Dev. Traj. Length	Avg. Dist. from $\mathcal{O}$	Std. Dev. Dist. from $\mathcal{O}$
$\mu_1 \in [1.00, 2.00]$	32.02m	0.29m	3.72m	0.02m
$\mu_2 \in [0.05, 0.50)$	41.96m	4.02m	2.64m	0.15m
$\mu_3 \in (0.70, 0.95]$	38.10m	8.85m	1.60m	0.26m
$\mu_4 \in [0.70, 1.00]$	30.58m	0.31m	1.34m	0.02m
$\mu_5 \in [0.01, 0.50)$	31.29m	0.75m	1.35m	0.04m

This data set was produced by evenly sampling each parameter's interval 10 times for a total of 50 simulations

Table 10 Average length of the UAV's reference trajectories, standard deviation of the length of the UAV's reference trajectories, average distance from the obstacles' set, and standard deviation of the distance from the obstacles' set when varying $\mu_1, \ldots, \mu_5$ one at the time within ranges that produce tactical behaviors, while the remaining four parameters are fixed and take values that induce tactical behaviors

Varied Parameters	Avg. Traj. Length	Std. Dev. Traj. Length	Avg. Dist. from $\mathcal{O}$	Std. Dev. Dist. from $\mathcal{O}$
$\mu_1 \in (0.00, 1.00]$	30.23m	1.29m	1.31m	0.22m
$\mu_2 \in [0.50, 0.95]$	40.87m	6.41m	2.57m	0.44m
$\mu_3 \in [0.01, 0.70]$	52.42m	11.43m	1.37m	0.29m
$\mu_4 \in [0.05, 0.70)$	32.61m	4.03m	1.40m	0.18m
$\mu_5 \in [0.50, 0.95]$	30.75m	0.71m	1.34m	0.03m

This data set was produced by evenly sampling each parameter's interval 10 times for a total of 50 simulations

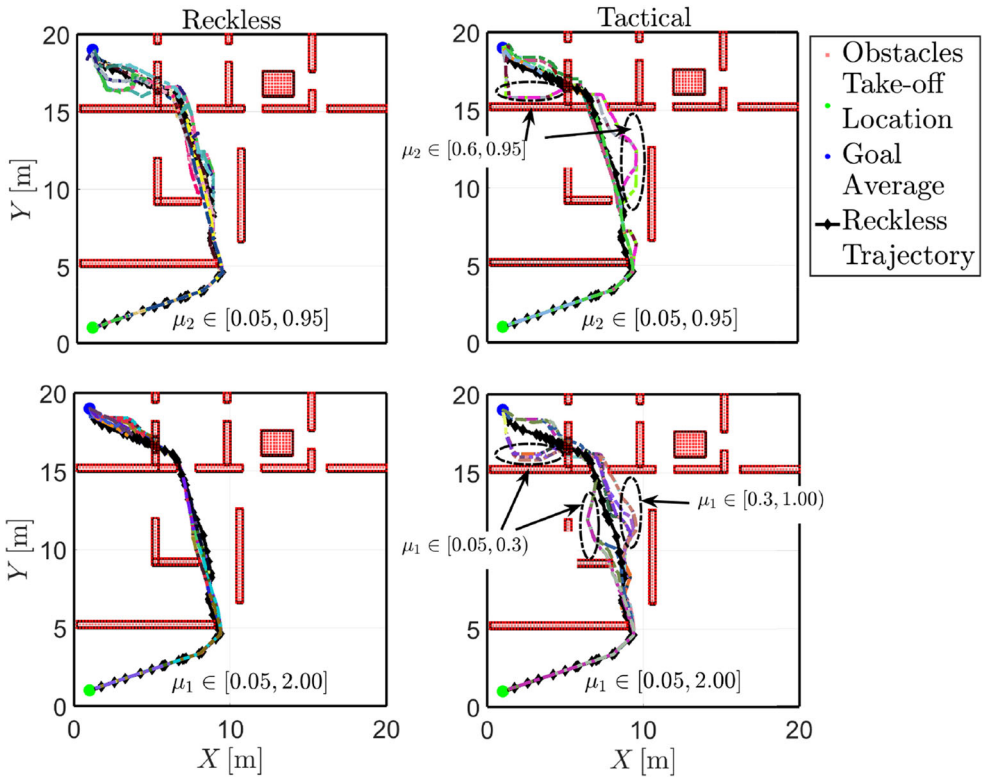

Fig. 9 Software-in-the-loop simulation results to establish a taxonomy of flight behaviors as a function of the user-defined parameters μ_2 and μ_1. The top left image shows the results of 20 simulations obtained by varying μ_2 in [0.05, 0.95], while μ_1, μ_3, μ_4, and μ_5 take values that induce a reckless trajectory. The top right image shows the results of 20 simulations obtained by varying μ_2, while μ_1, μ_3, μ_4, and μ_5 take values that induce a tactical trajectory. The bottom left image shows the results of 20 simulations obtained by varying μ_1, while $\mu_2, \ldots, \mu_5$ take values that induce a reckless trajectory. The bottom right image shows the results of 20 simulations obtained by varying μ_1 in [0.05, 2.00], while $\mu_2, \ldots, \mu_5$ take values that induce a tactical trajectory. The black line in the top images show the trajectory obtained by averaging over all trajectories obtained by spanning μ_2 in [0.05, 0.95], while μ_1, μ_3, μ_4, and μ_5 take values that induce a reckless trajectory. The bottom images show the trajectory obtained by averaging over all trajectories obtained by spanning μ_1 in [0.05, 2.00], while $\mu_2, \ldots, \mu_5$ take values that induce a reckless trajectory. It appears from the images on the left that if four of the five user-defined parameters are designed to impose a reckless behavior, then the UAV's reference trajectories do not deviate ostensibly from the average reckless trajectory, despite the fact that the fifth parameter takes also values that would induce a tactical behavior. It appears from the images on the right that if four of the five user-defined parameters are designed to impose a tactical behavior, then the UAV's reference trajectories are tactical whenever the fifth parameter takes values that would induce a tactical behavior, and the UAV's reference trajectories are reckless whenever the fifth parameter takes values that would induce a reckless behavior

1.00), (0.30, 0.75, 0.10), (0.20, 0.85, 0.08), (0.10, 0.95, 0.05)} and $\mathcal{S}_2 \triangleq \{(1.00, 0.001), (0.90, 0.01), (0.80, 0.01), (0.50, 0.01), (0.40, 0.90), (0.30, 1.00)\}$; the triplets in $\mathcal{S}_1$ and the pairs in $\mathcal{S}_2$ are sorted according to their ability to induce an increasingly cautious behavior in the UAV. In all 36 simulations, we set $r_0 = [1.00, 1.00, 1.00]^{\mathrm{T}}$m and $\mathcal{G} \triangleq \{[19.00, 17.00, 1.00]^{\mathrm{T}}\}$m.

Figures 10 and 11 show the number of times unoccupied voxels in a given voxel map are traversed by the UAV. In particular, Fig. 10 shows the simulation results assuming that the triplet (μ_1, μ_2, μ_3) is fixed and the pair (μ_4, μ_5) takes values in $\mathcal{S}_2$, and Fig. 11 shows the simulation results assuming that the pair (μ_4, μ_5) is fixed and the triplet (μ_1, μ_2, μ_3) takes values in $\mathcal{S}_1$. Both figures show that immediately after take-off and short before traversing the door to the goal point, all reference trajectories are substantially overlapping. To reach the door to the goal point, three clusters of reference trajectories are shown in both figures, namely *Cluster 1* of more tactical trajectories coasting the wall at $X = 20$m, and *Clusters 2* and *3* of more reckless trajectories. Figure 10 shows that, for any $(\mu_4, \mu_5) \in \mathcal{S}_2$, *Cluster 1* collects reference trajectories obtained with $(\mu_1, \mu_2, \mu_3) \in \{(0.30, 0.75, 0.10), (0.20, 0.85, 0.08), (0.10, 0.95, 0.05)\}$, *Cluster 2* collects reference trajectories obtained with $(\mu_1, \mu_2, \mu_3) = (2.00, 0.10, 1.50)$, and *Cluster 3* collects reference trajectories obtained with $(\mu_1, \mu_2, \mu_3) \in \{(1.20, 0.30, 1.00), (1.60, 0.20, 1.25)\}$. Figure 11 shows that for any $(\mu_1, \mu_2, \mu_3) \in \mathcal{S}_1$, *Cluster 1* collects tactical reference trajectories prevalently obtained with $(\mu_4, \mu_5) \in \{(0.50, 0.80), (0.40, 0.90), (0.30, 1.00)\}$, *Clusters 2* collects re-ference trajectories prevalently obtained with $(\mu_4, \mu_5) \in \{(1.00, 0.001), (0.90, 0.10)\}$, and *Clusters 3* collects reference trajectories prevalently obtained with $(\mu_4, \mu_5) \in \{(0.90, 0.10), (0.80, 0.70)\}$. However, each cluster collects reference trajectories produced by all $(\mu_4, \mu_5) \in \mathcal{S}_2$. Therefore, we deduce that the triplet (μ_1, μ_2, μ_3) affects the UAV's level of cautiousness more than the pair (μ_4, μ_5).

Finally, both in Fig. 10 and in Fig. 11, *Cluster 1* is wider than *Clusters 2* and *3*. Therefore, consistently with our analysis in Section 7.2, if a reckless reference trajectory is to be produced, then the proposed guidance system exhibits a more predictable behavior.

7.4 Taxonomy of Flight Behaviors Varying the Take-off Location

In this section, we present the results of 100 software-in-the-loop simulations aimed at establishing a taxonomy of flight behaviors as a function of the UAV's take-off position. For the first set of tests, we chose $\mu_1 = 1.00$, $\mu_2 = 0.50$, $\mu_3 = 0.10$, $\mu_4 = 0.80$, and $\mu_5 = 0.10$ to obtain reckless reference trajectories, and for the second set of test, we chose $\mu_1 = 0.10$, $\mu_2 = 0.75$, $\mu_3 = 0.05$, $\mu_4 = 0.70$, and $\mu_5 = 0.50$ to obtain tactical reference trajectories. In both sets of numerical simulations, the UAV's initial position was given by $r_0 = [r_{x,0}, r_{y,0}, 0.6]^{\mathrm{T}}$, where $(r_{x,0}, r_{y,0})$ are evenly spaced in $[2, 10] \times [1, 4]$m, the goal set was given by $\mathcal{G} = \{[1.00, 19.00, 0.60]^{\mathrm{T}}\}$m, the remaining simulation parameters were the same as in Section 7.1.

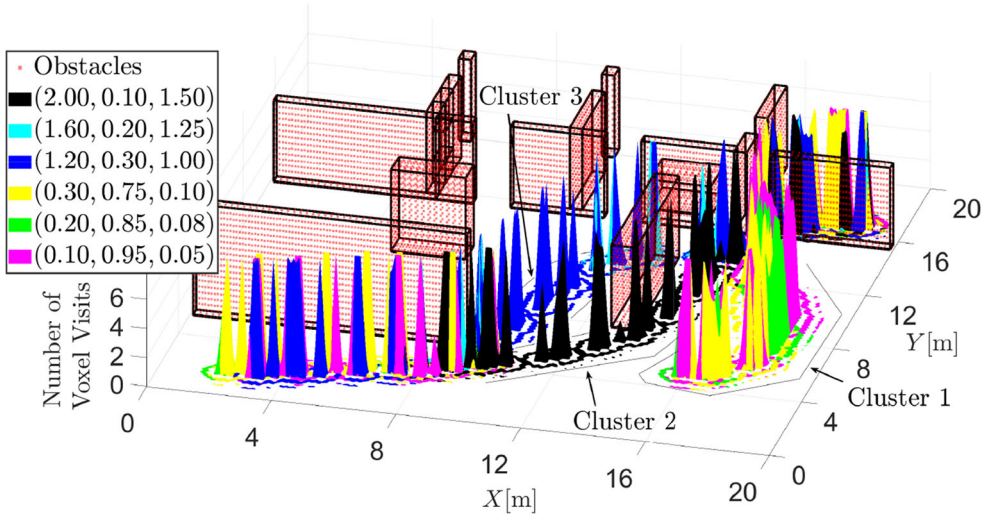

Fig. 10 Number of times unoccupied voxels in the given occupancy map are traversed by the UAV. These simulation results have been achieved by assuming that the triplet (μ_1, μ_2, μ_3) is fixed and the pair (μ_4, μ_5) takes values in $\mathcal{S}_2$. Immediately after take-off and short before traversing the door to the goal point, all reference trajectories are substantially overlapping. To reach the door to the goal point, all trajectories group in three main clusters. *Cluster 1* collects reference trajectories obtained with $(\mu_1, \mu_2, \mu_3) = \{(0.30, 0.75, 0.10), (0.20, 0.85, 0.08), (0.10, 0.95, 0.05)\}$, *Cluster 2* collects reference trajectories obtained with $(\mu_1, \mu_2, \mu_3) = (2.00, 0.10, 1.50)$, and *Cluster 3* collects reference trajectories obtained with $(\mu_1, \mu_2, \mu_3) = \{(1.20, 0.30, 1.00), (1.60, 0.20, 1.25)\}$

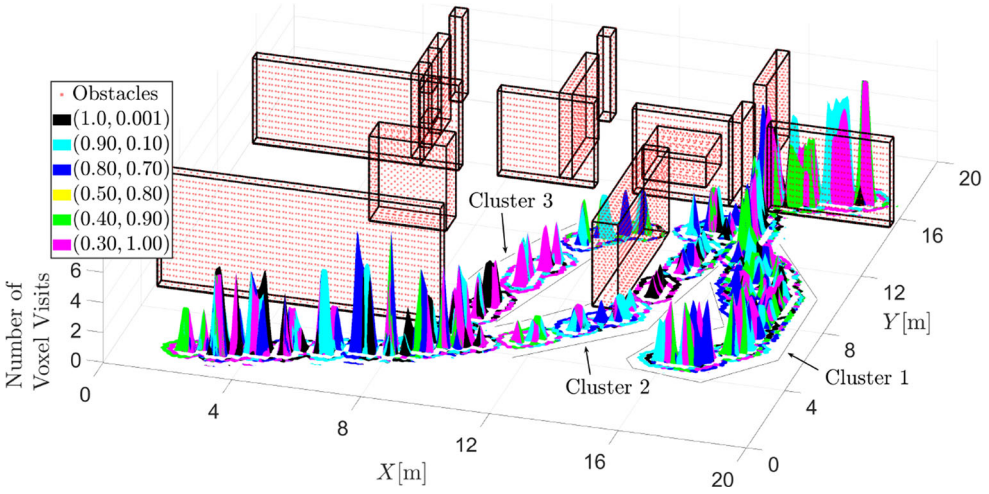

Fig. 11 Number of times unoccupied voxels in the given occupancy map are traversed by the UAV. These simulation results have been achieved by assuming that the pair (μ_4, μ_5) is fixed and the triplet (μ_1, μ_2, μ_3) takes values in $\mathcal{S}_1$. As in Fig. 10, immediately after take-off and short before traversing the door to the goal point set, all reference trajectories are substantially overlapping. Furthermore, all trajectories group in three main clusters. *Cluster 1* collects tactical reference trajectories prevalently obtained with $(\mu_4, \mu_5) = \{(0.50, 0.80), (0.40, 0.90), (0.30, 1.00)\}$, *Clusters 2* collects reference trajectories prevalently obtained with $(\mu_4, \mu_5) = \{(1.00, 0.001), (0.90, 0.10)\}$, and *Clusters 3* collects reference trajectories prevalently obtained with $(\mu_4, \mu_5) = \{(0.90, 0.10), (0.80, 0.70)\}$

Figure 12 shows the results of these flight tests. All reckless reference trajectories cluster upon passing the closest wall: the standard deviation around the average reckless trajectory is 0.42m. Conversely, tactical trajectories are considerably more spread: the standard deviation around the average tactical trajectory is 2.64m. Therefore, consistently with our analysis in Sections 7.2 and 7.3, this analysis suggests that if a reckless reference trajectory is to be produced, then the proposed guidance system exhibits a more predictable behavior. Conversely, with the tactical parameter set, all reference trajectories converge immediately before passing through the only door to the goal set. Thus, if a tactical reference trajectory is to be produced, then the proposed guidance system exhibits a less predictable behavior.

7.5 Taxonomy of Flight Behaviors Varying the Occupancy Map

In this section, we present the results of two sets of software-in-the-loop simulations aimed at establishing a taxonomy of flight behaviors as a function of the density of occupied voxels. For both sets of simulations, we considered a $20 \times 20 \times 6$m environment, and we let $r_0 = [1.00, 1.00, 1.00]^{\mathrm{T}}$m and $\mathcal{G} = \{[19.00, 17.00, 1.60]^{\mathrm{T}}\}$m. For a set of simulations, we let $\mu_2 = 0.40$, $\mu_4 = 1.00$, and $\mu_5 = 0.01$ to produce reckless trajectories, and for the other set of simulations, we let $\mu_2 = 0.75$, $\mu_4 = 0.80$, and $\mu_5 = 0.50$ to produce tactical trajectories; the remaining user-defined parameters were the same as in Section 7.1. In both sets of simulations, the first test was performed assuming that the environment

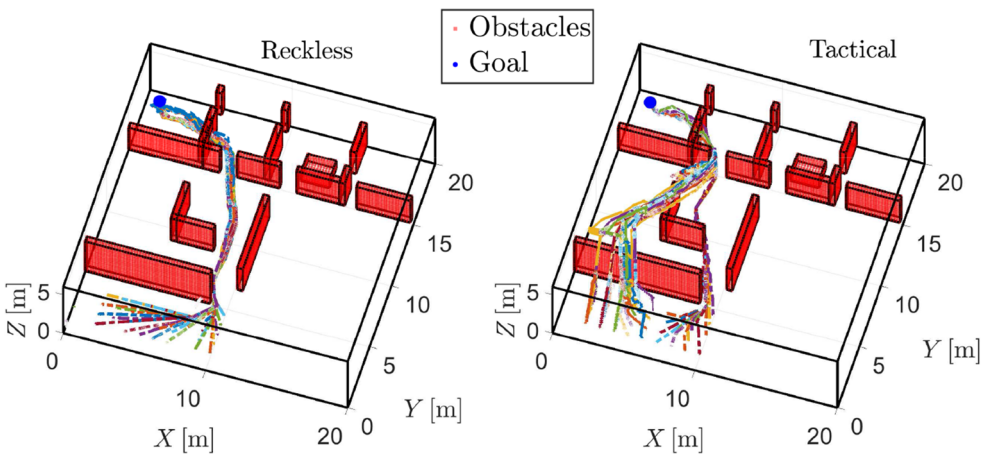

Fig. 12 Fifty reckless and fifty tactical reference trajectories obtained by iteratively varying the UAV's initial position $r_0 = [r_{x,0}, r_{y,0}, 0.6]^{\mathrm{T}}$, where $(r_{x,0}, r_{y,0})$ are evenly spaced in $[2.0, 10.0]$m $\times [1.0, 4.0]$m. In both sets of simulations, the goal set is given by $\mathcal{G} = \{[1.0, 19.0, 0.6]^{\mathrm{T}}\}$m. It is apparent that reckless trajectories are highly clustered around the shortest trajectory and hence, are more predictable, whereas tactical trajectories are more spread and hence, less predictable

was free of obstacles. Successively, at random locations, we iteratively introduced new obstacles, whose size randomly varied from $0.20 \times 0.20 \times 0.20$m to $4.00 \times 4.00 \times 6.00$m, and we let the proposed guidance system compute both a reckless and a tactical reference trajectory; the obstacles' maps were produced by a Matlab script employing a normal distribution, whose seed is Matlab's default global stream. For both sets of simulations, the maximum density of occupied voxels considered in both sets of simulations was 30% since for larger densities of randomly occupied voxels, it was impossible to find viable trajectories.

Figure 13 shows both a plot of the length of UAV's reckless reference trajectories, averaged over the set of reckless reference trajectories computed at previous iterations, and a plot of the length of UAV's tactical reference trajectories, averaged over the set of tactical reference trajectories computed at previous iterations. Averaging the trajectory length over the set of trajectories produced at previous iterations allows to capture the effect of the density of the voxel map on the UAV's behavior, irrespectively of the effect produced by individual obstacles introduced at each iteration. At the first iteration, in the absence of obstacles, both the reckless and the tactical trajectories were overlapping straight lines joining the initial condition r_0 to the goal set $\mathcal{G}$. For a density of occupied voxels in the interval $[0, 24.9]\%$, the average length of the reckless trajectories grew almost linearly at a rate of 1.40m per unit density. Over the density interval $[0, 1.3]\%$, the average length of the tactical trajectories grew almost linearly at a rate of 23.82m per unit density since the few obstacles introduced at the first iterations attracted the UAV more markedly. Over the density interval $[1.3, 10.7]\%$, the average length of the tactical trajectories decreased at a rate of 3.26m per unit density since the number of scattered occupied voxels increased and multiple obstacles nullified their attractive effect reciprocally. Over the density interval $[10.7, 24.9]\%$, the average length of the tactical trajectories grew almost linearly at a rate of 1.25m per unit density; remarkably, over the same interval, the average length of the tactical trajectories grew at a rate of 1.23m per unit density. From this analysis, we deduce that over an extended density interval, such as $[0, 24.9]\%$, reckless trajectories are less influenced by the occupancy map, except for the problem of avoiding obstacles, and hence, the average trajectory length grows linearly. Conversely, over the same interval, tactical trajectories are more influenced by the occupancy map, especially for low levels of obstacles' density. Finally, over the density interval $[24.9, 30.0]\%$, the average length of the reckless trajectories decreased almost linearly at a rate of 0.92m per unit density and the average length of the reckless trajectories decreased almost linearly at a rate of 0.89m per unit density; this synchronous decrease of the average length of the reckless and tactical trajectories was due to the large number of occupied voxels, which force the UAV to fly over several obstacles to reach the goal set.

Figure 14 shows the 448 reckless and tactical trajectories produced as part of proposed study both in the horizontal and vertical planes; obstacles are omitted for clarity of presentation. It is apparent that reckless trajectories cluster with one another both in the horizontal and the vertical planes, whereas tactical trajectories are spread both in the horizontal and vertical planes. Therefore, consistently with the analyses presented in Sections 7.2–7.4, reckless trajectories are more predictable than tactical ones.

8 Flight Tests Results

In this section, we present the results of three sets of flight tests. The first set of tests shows the applicability and some of the capabilities of the proposed guidance system. The second set of tests aims to show the repeatability of the

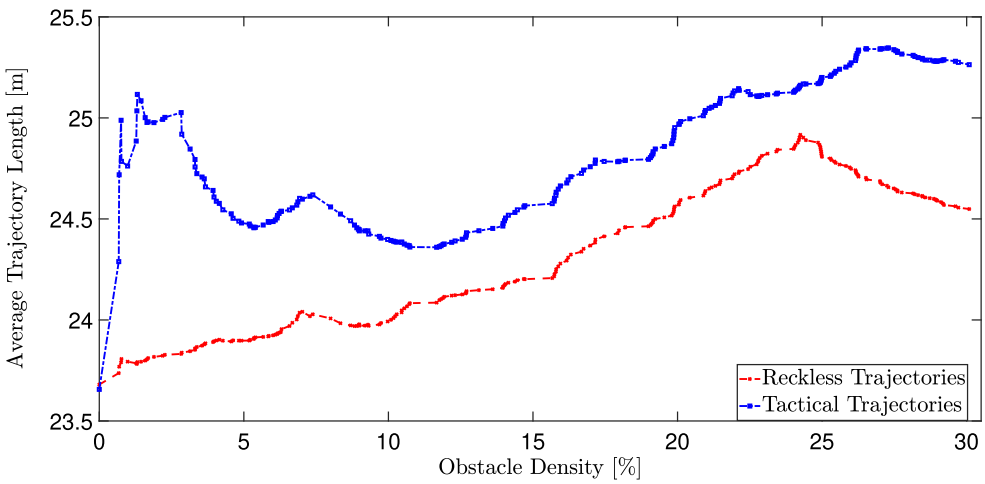

Fig. 13 Average length of the UAV's reckless and tactical reference trajectories as functions of the density of randomly generated obstacles in a given environment. Over 448 simulations, reckless trajectories are consistently shorter than tactical trajectories. For low values of the density of the obstacles' set, the average length of the reckless trajectories grows suddenly, since the effect of the few obstacles is stronger. Over the interval $[10.7, 30.0]\%$ the average length of the reckless and of the tactical trajectories follow similar trends

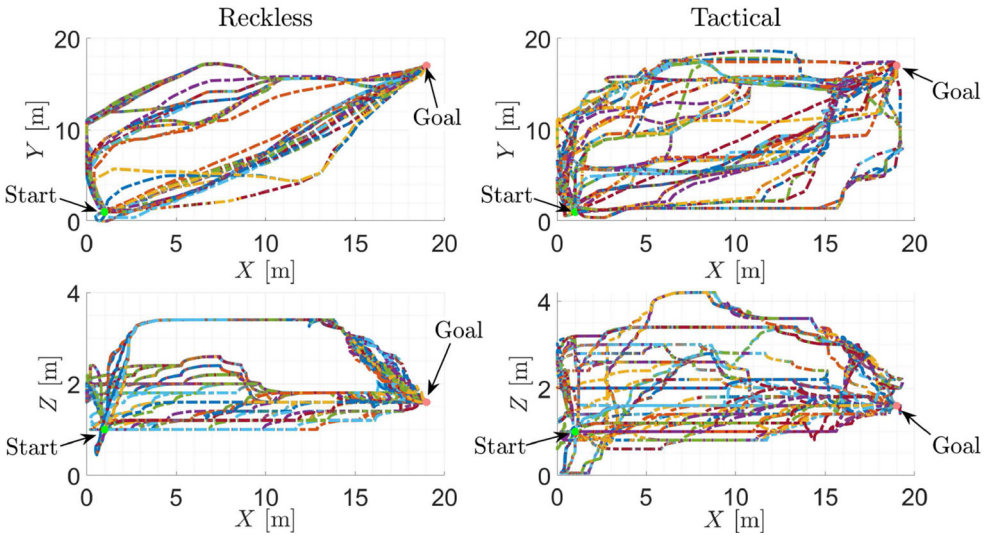

Fig. 14 Reckless and tactical reference trajectories obtained by iteratively introducing randomly generated obstacles in a given environment; obstacles are omitted for clarity of presentation. Over 224 simulations, reckless trajectories appear to cluster both in the horizontal and the vertical planes, whereas over 224 simulations, tactical trajectories are more spread than their reckless counterparts both in the horizontal and the vertical plane. Therefore, reckless trajectories and more predictable than tactical ones

flight test results. The third set of tests has two goals, namely examining the effect of varying initial conditions in a given environment and for a fixed goal point and showing the consistency between flight test and software-in-the-loop simulation results. For these three sets of flight tests, both a reckless and a tactical set of parameters are employed. To perform these tests, no map of the environment was pre-stored on the UAV's single-board computer.

8.1 Validation of the Proposed Guidance System through Flight Tests

In this section, we present the result of two flight test aimed to show how the proposed guidance system allows a quadcopter to reach the given goal set while exhibiting a reckless and a tactical behavior; videos of both flights can be found at [57]. In both flight tests, the UAV's mission is to take off from a given initial position, traverse a cluttered environment without prior knowledge of obstacles' set, avoid colliding with obstacles, and finally reach the goal set given by a disk of radius 1.75m, which is centered at a point specified with respect to the take-off position. The quadcopter takes off in a high-bay area and its goal set is located in an office space, which is accessible from the high-bay area through a 0.8m wide door. Multiple obstacles of different size and shape have been placed in the high-bay area. In order to demonstrate the results of a reckless flight test, the UAV's initial position and yaw angle have been chosen so that the goal set is visible from the take-off point through a cone of 3.25° aperture. This way, the most reckless trajectory consists of a straight line from the UAV's take-off position to the goal set. However, the proposed guidance system guarantees successful mission completion also in the case the goal set is not visible from the take-off position.

Figures 15 and 16 show the results of both flight tests in three dimensions and two dimensions, respectively. In both tests, the UAV takes off from the point $[3.00, -3.60, 1.00]^{\mathrm{T}}$ in the reference frame generated by the navigation system, which is denoted by a green hexagon, and its goal set is denoted by a salmon-colored disk centered at $[12.00, -7.60, 0.20]^{\mathrm{T}}$. Furthermore, in both flight tests we set $u_{\max} = [1.50, 0.25, 0.25, 0.25]^{\mathrm{T}}$, $u_{\min} = -[0, 0.25, 0.25, 0.25]^{\mathrm{T}}$, $\psi_{\max} = 43.00°$, $n_{\mathrm{t}} = 100$, $\Delta T = 0.01$s,

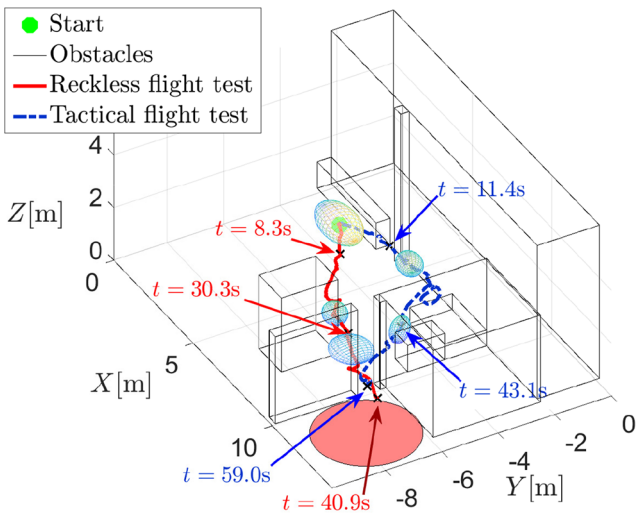

Fig. 15 Three-dimensional view of the experimental results. The take-off position is indicated by a green hexagon, obstacles are captured by means of a wire-frame representation, and the goal set is denoted by the salmon-colored disk. The UAV's reckless trajectory is represented by a solid red line, and its tactical trajectory is represented by a dashed blue line. The ellipsoids used to form the collision avoidance constraints are represented at $t = 0.0$s, $t = 27.3$s, and $t = 33.1$s for the reckless flight, and $t = 0.0$s, $t = 13.0$s, and $t = 43.9$s for the more tactical flight. Finally, timestamps are shown at multiple points along the UAV's trajectories. A video of these flight tests can be found at [57]

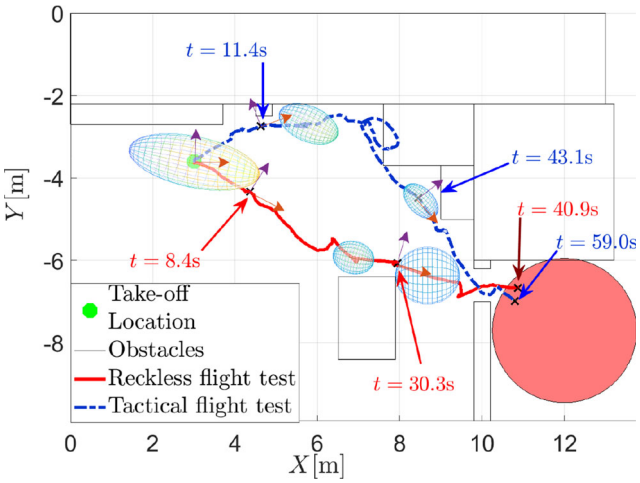

Fig. 16 View of the experimental results in the horizontal plane. The UAV's heading angle is captured by the UAV's roll axis, which is represented by orange arrows at multiple time instants. Purple arrows show the UAV's pitch axis

$\tilde{R}_r = R_{r,\mathrm{f}} = 700\,\mathbf{1}_3$, $\tilde{R}_{r,u} = 0_{3\times 4}$, $\tilde{R}_u = \mathrm{blockdiag}(10, 200, 200, 300)$, $\tilde{q}_r = q_{r,\mathrm{f}} = 0_3$, $\tilde{q}_u = 0_4$, $\mu_1 = 0.10$, $\mu_6 = 1.00$, $\mu_7 = 1.00$, $\mu_8 = 0.00$, $\nu_1 = 2.00$, $\nu_2 = 10.00$, $\nu_3 = 0.45$, and $\varepsilon = 10^{-3}$; the constraint $\psi_{\max}$ captures half of the field-of-view angle of the depth camera, which is responsible for mapping the environment. Furthermore, in both tests, the ellipsoid $\overline{\mathcal{E}}_k(\cdot)$ was sampled by setting $l = 24$. The parameters used to induce a reckless behavior were $\mu_2 = 0.20$, $\mu_3 = 0.20$, $\mu_4 = 1.00$, $\mu_5 = 0.01$, and $\mu_9 = 0.30$. The parameters used to induce a tactical behavior were $\mu_2 = 0.90$, $\mu_3 = 0.50$, $\mu_4 = 0.80$, $\mu_5 = 0.40$, and $\mu_9 = 0.40$.

While following the more reckless trajectory, the UAV flew toward a box placed in front of the door to the office space, where the goal set was located, and then proceeded along a substantially straight path to the goal set completing its mission in 40.9s. It is worthwhile to note that the UAV entered the room, where the goal set was located, by traversing the center of the doorway. While following a more tactical trajectory, the UAV immediately sought cover by flying towards one of the walls of the high-bay area. At $t = 27.3$s, the UAV flew over a box and the navigation system improved the map of the environment. Thus, the guidance system computed both a reference path and a reference trajectory that were considered as more cautious; this re-planning produced the loop shown in Figs. 15 and 16. Then, the UAV continued coasting the obstacles' set and proceeded through the doorway completing its mission in 59.0s. It is worthwhile to note that, by following this more tactical trajectory, the UAV coasted the door frame.

At each time step, the ellipsoids used to generate the collision avoidance constraints were tangent to some obstacles such as walls, boxes, or the floor. By coasting the obstacles' set closely, more tactical flight behaviors imply smaller ellipsoids, and hence, the reference trajectory must be computed more frequently than by flying along more reckless trajectories.

The UAV's average distance from the obstacles' set when behaving more recklessly was 1.34m, whereas the UAV's average distance from the obstacles' set when behaving more cautiously was 0.96m. Hence, when following the more tactical path, the UAV was 29% closer the obstacles' set than than when behaving more recklessly. The UAV's maximum distance from the obstacles' set when behaving recklessly was 2.55m, whereas the UAV's maximum distance from the obstacles' set when behaving tactically was 2.34m. The distance traveled by the UAV along the reckless and tactical trajectories were 12.82m and 17.08m, respectively. On average, when behaving more cautiously, the UAV moved approximately 10% slower than when behaving more recklessly. The average height of the UAV while behaving more cautiously is 0.30m, whereas the average height of the UAV while behaving more recklessly is 0.41m. Hence, as predicted in Section 7, if the UAV is tasked to behave more tactically, then, in average, it is closer both to the obstacles' set and the ground and travels a longer distance. Conversely, if the UAV is tasked to behave more recklessly, then, in average, it is further from the obstacles' set and the ground and travels a shorter distance.

8.2 Analysis of the Repeatability of Flight Tests

In this section, we present the results of flight tests, where a reckless parameter set and a tactical parameter set are tested 5 times each to determine the predictability of the UAV's trajectories and establish repeatability of results employing the proposed guidance system. The UAV's mission is to start from $r_0 = [3.6, 1.8, 1.0]^{\mathrm{T}}$m, traverse an unknown environment, and reach the goal set $\mathcal{G} = \{[11.8, 14.2, 1.6]^{\mathrm{T}}\}$m with a reckless or tactical parameter set. To induce a more reckless behavior, we set $\mu_1 = 1.00$, $\mu_2 = 0.40$, $\mu_3 = 0.20$, $\mu_4 = 1.00$, and $\mu_5 = 0.01$, and to induce a more tactical behavior, we set $\mu_1 = 0.20$, $\mu_2 = 0.75$, $\mu_3 = 0.10$, $\mu_4 = 0.80$, and $\mu_5 = 0.50$; the remaining user-defined parameters are the same as in Section 8.1.

Figure 17 shows both in the horizontal and the vertical planes the 10 reckless and tactical trajectories produced as part of the proposed study, and Table 11 presents some statistical data of these trajectories. From Fig. 17 we deduce that reckless reference trajectories tend to cluster around the shortest trajectory and approach the goal point through

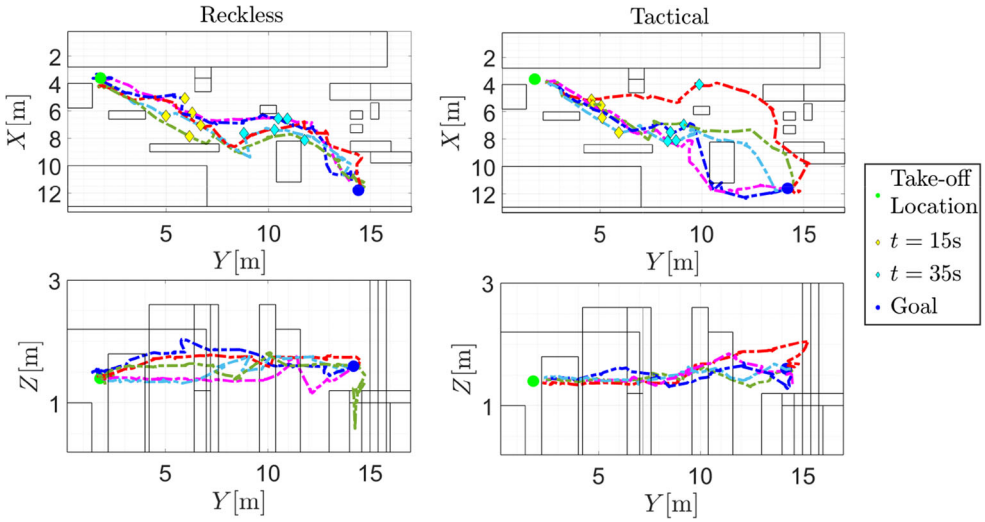

Fig. 17 Ten flight test results to determine the predictability of the UAV's trajectories and establish repeatability of results employing the proposed guidance system. The UAV trajectories from five flight tests performed employing a more reckless parameter set are shown in the left image, and the trajectories from five flight tests performed employing a more tactical parameter set are shown in the right image. In both cases, the take-off location is $r_0 = [3.6, 1.8, 1.0]^{\mathrm{T}}$m and the goal set is $\mathcal{G} = \{[11.8, 14.2, 1.6]^{\mathrm{T}}\}$m; approximately 11% of the voxels are occupied. Time stamps at $t = 15$s and $t = 35$s are provided to compare the UAV's velocity for each flight test and each parameter set. Reckless trajectories are consistently shorter, traversed in less time, and closer to the obstacles' set than tactical trajectories. Moreover, reckless trajectories tend to cluster around the shortest trajectory and approach the goal point through a narrow cone, whereas tactical reference trajectories are more scattered and approach the goal point through a wider cone

a narrow cone, whereas reckless reference trajectories are more scattered and approach the goal point through a wider cone. Therefore, consistently with the results presented in Sections 7.2–7.5, reckless trajectories are more predictable than tactical ones. Furthermore, from Table 11 we deduce that reckless trajectories are shorter, traversed in less time, and further from the obstacles' set $\mathcal{O}$ than tactical trajectories. Therefore, as anticipated in Section 7, reckless trajectories consistently aim to the goal set more directly than tactical trajectories. Finally, by the standard deviation about the average reckless trajectory and the average tactical trajectory, we deduce that reckless trajectories are more repeatable than tactical ones. This is due to the fact that, although although the parameters' set, the environment, the UAV's initial position, and the goal set are identical for each set of flight tests, small variations in the UAV's yaw angle lead the onboard navigation system to detect different environmental features at the boundaries of the depth camera's field-of-view and hence, produce slightly different occupancy maps. Since tactical trajectories are attracted by the obstacle's set more than reckless trajectories, tactical trajectories are less repeatable.

Table 11 Statistical data for the flight tests shown in Fig. 17

	Reckless Trajectory	Tactical Trajectory
Average trajectory length	17.20m	21.40m
Maximum trajectory length	18.51m	23.99m
Std. Dev. about average trajectory	0.97m	2.08m
Average distance from $\mathcal{O}$	0.91m	1.20m
Average flight time	53.69s	65.04s
Maximum flight time	72.54s	75.50s

8.3 Analysis of the Effect of Varying the Take-off Position

In this section, we analyze the results of 10 flight tests to assess both the effect of varying the UAV's take-off location on reckless and tactical trajectories and the consistency of the flight tests with the numerical simulations. Specifically, five flight tests have been performed by choosing the same set of reckless parameters as in Section 8.2 and the initial conditions $r_0^{\mathrm{T}} \in \mathcal{R}_0$, where $\mathcal{R}_0 \triangleq \{[4.0, 5.0, 1.4], [4.0, 3.5, 1.4], [4.0, 2.2, 1.4], [5.4, 2.2, 1.4], [7.8, 2.2, 1.4]\}$, and five flight tests have been performed by choosing the same set of tactical parameters as in Section 8.2 and $r_0^{\mathrm{T}} \in \mathcal{R}_0$. The environment and the goal point for these flight tests are the same as in Section 8.2.

Figure 18 shows both in the horizontal and the vertical planes the flight tests results produced as part of the proposed study, and Table 12 presents some statistical data of these trajectories. Consistently with the results anticipated in Section 7.4, fixed the initial condition, reckless trajectories are shorter and traversed in more time than tactical trajectories. Moreover, four of the five reckless trajectories are highly clustered and approach the goal

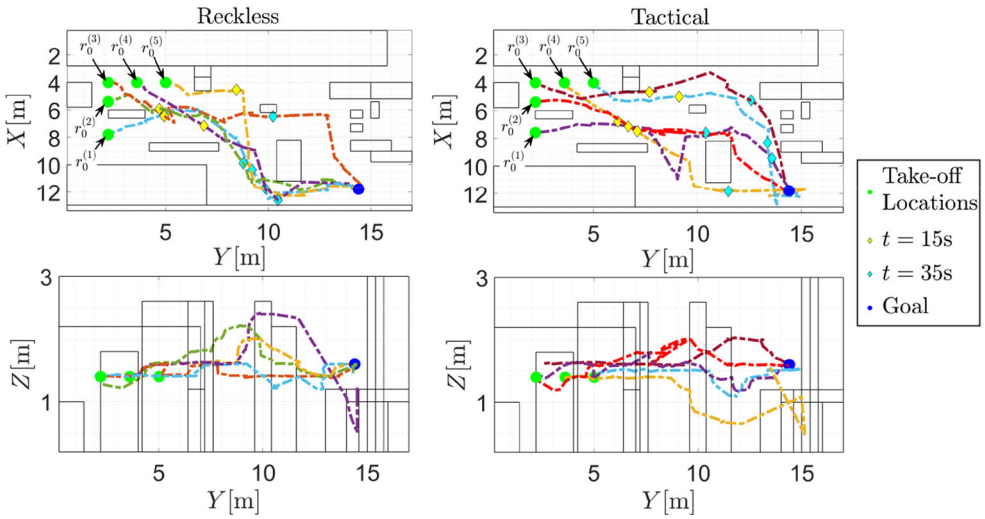

Fig. 18 Ten flight test results to analyze the effect of varying the UAV's take-off location on reckless and tactical trajectories; both the goal set and the obstacles' set are the same as in Fig. 17. Fixed the initial condition, reckless trajectories are consistently shorter and traversed in more time than tactical trajectories. Moreover, four of the five reckless trajectories are highly clustered and approach the goal point through a narrow cone and hence, are more predictable. All tactical reference trajectories are more scattered and approach the goal point through a wider cone and hence, are less predictable

point through a narrow cone, whereas all tactical reference trajectories are more scattered and approach the goal point through a wider cone. Therefore, consistently with the results presented in Section 7, reckless trajectories are more predictable than tactical ones.

Figure 19 shows both in the horizontal and the vertical planes the results of software-in-the-loop tests performed using the same voxel map and the same user-defined parameters as for the flight tests shown in Fig. 18. Similarly to the flight test results, four of the five simulated reckless trajectories overlap considerably and approach the goal point through a narrow cone, and all tactical trajectories are less clustered and approach the goal point through a wider cone. Table 13 presents some statistical data of these trajectories. For both reckless and tactical trajectories, the average trajectory length, the average flight time, and the maximum flight time for the software-in-the-loop tests are similar to the average trajectory length, the average flight time, and the maximum flight time for actual flight tests, respectively. For reckless trajectories, the maximum trajectory length in software-in-the-loop simulations is comparable to the maximum trajectory length in flight tests. However, for tactical trajectories, the maximum trajectory length in software-in-the-loop simulations is one-third larger than the maximum trajectory length in flight tests, which confirms the lower predictability of tactical flights. Therefore, software-in-the-loop simulation are useful to predict average behaviors of the proposed guidance system, although the variability of results is larger for tactical trajectories.

Table 12 Statistical data for the flight tests shown in Fig. 18

	Reckless Trajectory	Tactical Trajectory
Average trajectory length	145.81%	151.16%
Maximum trajectory length	156.02%	171.13%
Average distance from $\mathcal{O}$	2.54m	1.91m
Average flight time	49.39s	46.27s
Maximum flight time	63.38s	67.87s

Both the average trajectory length and the maximum trajectory length are normalized over the length of the segment joining the UAV's take-off position to the goal set

9 Comparison with the MAV Voxblox Planner

In this section, we present the outcomes of two sets of software-in-the-loop simulations performed using the same single board computer as in Section 7 and compare the performance of the proposed guidance system with those of the MAV Voxblox planner [68, 69], a state-of-the-art guidance system for multi-rotor UAVs. The scope of this comparative analysis is to showcase the ability of the proposed guidance system to effectively outline reference trajectories in unknown environments over an advanced algorithm employing a similar architecture. MAV Voxblox has been chosen as it comprises both an optimization-based path planner and a model predictive control law for trajectory planning.

In both sets of simulations, we compute a reference trajectory employing MAV Voxblox planner and a tactical and a reckless reference trajectory employing the proposed

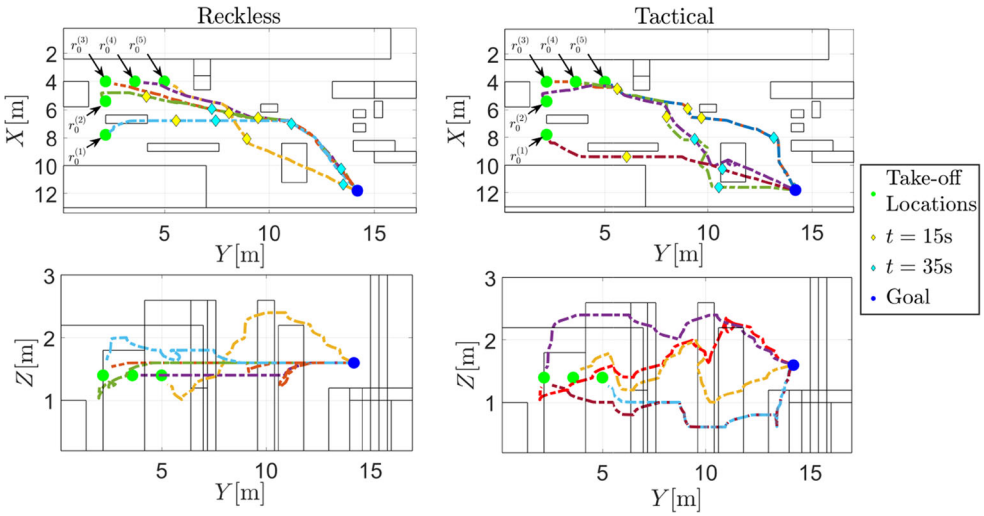

Fig. 19 Ten software-in-the-loop test results performed employing the same voxel map and the same user-defined parameters as for the flight tests shown in Fig. 18. Similarly to the flight test results, four of the five simulated reckless trajectories overlap considerably and approach the goal point through a narrow cone, and all tactical trajectories are less clustered and approach the goal point through a wider cone. For additional details, see Table 13

guidance system and the same user-defined parameters as in Section 7.1. Furthermore, both sets of simulations require the UAV to reach $\mathcal{G} = \{[19.0, 1.0, 1.0]^{\mathrm{T}}\mathrm{m}\}$, while traversing a maze provided in [68]. For the first set of simulations, the UAV takes off from $r_0^{(1)} = [1.0, 11.0, 1.0]^{\mathrm{T}}\mathrm{m}$, and for the second set of simulations, the UAV takes off from from $r_0^{(2)} = [5.0, 15.0, 1.0]^{\mathrm{T}}\mathrm{m}$. In both sets of simulations, it is assumed that the UAV's onboard cameras detect obstacles located at a distance that is less or equal to 5m from the UAV. However, consistently with the work presented in [68, 69], in simulations performed using the MAV Voxblox planner, it is assumed that the guidance system discovers occupied voxels within a ball centered in the UAV, whereas in simulations performed using the proposed guidance system, it is assumed that the guidance system discovers occupied voxels within a cone, whose apex is at the UAV's current position. The proposed guidance system prescribes that the UAV's velocity at the waypoints given by the path planner is given by $[0.5, 0.5, 0.0]^{\mathrm{T}}\mathrm{m/s}$, whereas the MAV Voxblox planner does not impose the UAV's reference velocity at any point and constrains the maximum velocity to 2m/s and the maximum acceleration to $2\mathrm{m/s}^2$. Compared to the proposed guidance system, this feature gives MAV Voxblox additional freedom to find viable reference trajectories.

Figure 20 shows the UAV's trajectories produced as part of both sets of numerical simulations; dashed lines with square markers show reckless trajectories produced employing the proposed guidance system, dashed lines with round markers show tactical trajectories produced employing the proposed guidance system, and solid lines show trajectories produced employing the MAV Voxblox planner. Table 14 presents statistical data of the reference trajectories shown in Fig. 20. Although all trajectories take similar routes to reach the goal set from a given take-off position, tactical reference trajectories produced by the proposed guidance system were closest to the obstacles' set, and reference trajectories produced by the MAV Voxblox planner were furthest from the obstacles' set. Indeed, the MAV Voxblox planner is not designed to produce tactical reference trajectories. Furthermore, the

Table 13 Statistical data for the flight tests shown in Fig. 19

	Simulated Reckless Trajectory	Difference with Flight Test	Simulated Tactical Trajectory	Difference with Flight Test
Average trajectory length	150.63%	4.82%	164.28%	13.12%
Maximum trajectory length	165.88%	9.85%	203.30%	32.17%
Average flight time	53.69s	4.30s	48.37s	2.27s
Maximum flight time	61.00s	−2.28s	56.41s	−11.46s

Both the average trajectory length and the maximum trajectory length are normalized over the length of the segment joining the UAV's take-off position to the goal set. The third and fifth columns present the difference between the trajectory produced through simulations and the trajectory from flight tests shown in Fig. 18. exhibiting how the trajectories produced by simulations emulates the trajectories produced by flight tests

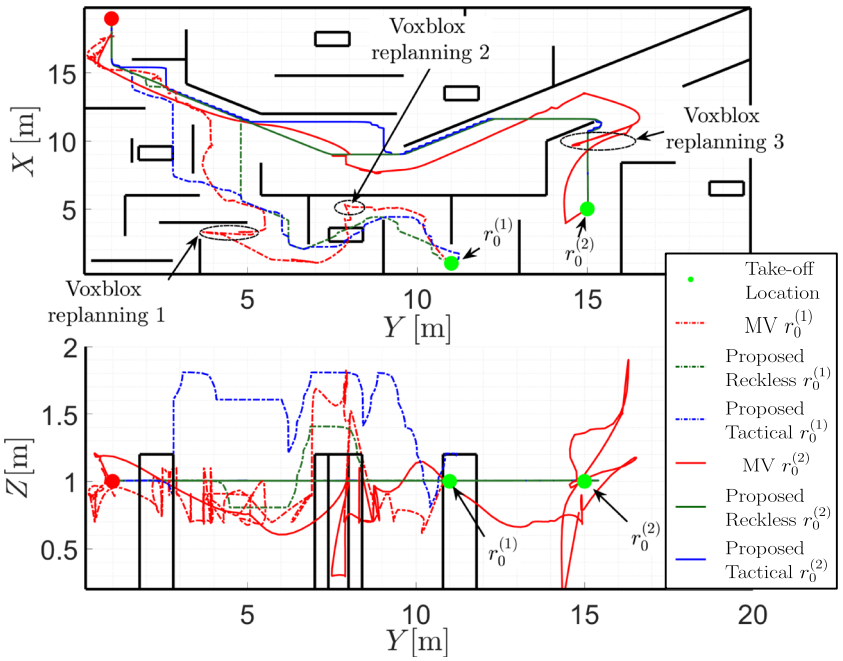

Fig. 20 Software-in-the-loop simulation results to compare the capabilities of the proposed guidance system (blue lines for reckless trajectories and green lines for tactical trajectories) with MAV Voxblox (MV) planner (red lines) [68, 69]. The take-off locations are given by $r_0^{(1)} = [1.0, 11.0, 1.0]^{\mathrm{T}}$ and $r_0^{(2)} = [5.0, 15.0, 1.0]^{\mathrm{T}}$. The goal set is given by $\mathcal{G} = \{[19.0, 1.0, 1.0]^{\mathrm{T}}\}$. The MAV Voxblox planner guidance system requires several attempts to escape from densely occupied areas and, on one occasion, lead the UAV to the ground. The proposed guidance system did not incur in the these problems

trajectory length was shortest for reference trajectories produced by the proposed guidance system with a reckless parameter set and longest for trajectories produced by the MAV Voxblox planner. Thus, the MAV Voxblox planner appears to produce longer reference trajectories without any specific reasons. The MAV Voxblox planner required several attempts to traverse the densely occupied areas marked by black ellipses in Fig. 20. The proposed guidance system did not require multiple attempts to find a solution to the trajectory planning problem both when tasked with finding reckless trajectories and when tasked with finding tactical trajectories. Finally, on several occasions, the MAV Voxblox planner lead the UAV to the ground, whereas the proposed guidance system never incurred in this problem.

10 Conclusion

This paper presented an innovative guidance system that allows autonomous vehicles to exhibit a tactical behavior, while operating in unknown, potentially hostile environments. This guidance system solely relies on information on the environment provided by a navigation system and deduced from a stereo depth camera, a tracking camera, and an inertial measurement unit. This guidance system has been customized for Class I quadcopters, and its efficiency has been tested by means of numerical simulations and flight tests in realistic scenarios.

One of the novelties of the proposed guidance system resides in the fact that it allows an autonomous vehicle

Table 14 Statistical data for the flight tests shown in Fig. 20

	$r_0^{(1)}$			$r_0^{(2)}$		
	Proposed Reckless	Proposed Tactical	Voxblox Planner	Proposed Reckless	Proposed Tactical	Voxblox Planner
Trajectory length	136.37%	149.21%	236.72%	140.46%	154.31%	233.87%
Avg. distance from $\mathcal{O}$	0.69m	0.69m	0.74m	0.93m	0.67m	1.33m
Max distance from $\mathcal{O}$	1.41m	1.39m	1.63m	2.29m	1.34m	2.37m

The trajectory length is normalized over the Euclidean distance from the respective take-off positions to the goal set. Tactical reference trajectories produced by the proposed guidance system are closest to the obstacles' set, and reference trajectories produced by the MAV Voxblox planner are furthest from the obstacles' set. The trajectory length was shortest for reference trajectories produced by the proposed guidance system with a reckless parameter set and longest for trajectories produced by the MAV Voxblox planner. Thus, the MAV Voxblox planner appears to produce reference trajectories that are longer without any specific reasons

to move in a tactical manner without any prior deterministic or stochastic knowledge of the obstacles' and the opponents' location. This tactical behavior is achieved by exploiting obstacles, seeking shelter from opponents that are able to acquire targets in direct line-of-sight, and regulating the UAV's velocity as a function of the distance from the obstacles' set. An additional novelty of the proposed guidance system lies in its architecture, which comprises an optimization-based path planner, an algorithm to produce collision avoidance constraint sets, whose boundaries are captured by affine functions, that are tangent to the obstacles' set, and an optimal control-based trajectory planning algorithm. Soft constraints allow to anticipate hard constraints and mitigate the sudden increases in control effort associated with the activation of hard constraints.

Numerical simulations illustrated key features of the proposed guidance system and allowed to produce a taxonomy of flight behaviors as a function of the most influencing user-defined parameters employed to induce a reckless or a cautious behavior, the take-off locations, and the occupancy map. Flight tests demonstrated the applicability of the proposed results and allowed to verify both the repeatability of flight tests and the sensitivity of the proposed guidance systems to variations in the UAV's take-off location. Both the numerical and the flight tests reveal that more reckless trajectories are also more predictable and hence, more vulnerable, than more tactical ones. Furthermore, when set to exhibit a more tactical behavior, the UAV is more sensitive to smaller variations in the environmental conditions.

Numerical simulations compared both the applicability and the performance of the proposed guidance system to those of the recently developed MAV Voxblox planner. Furthermore, numerical simulations demonstrate the higher computational efficiency of the proposed algorithm to determined convex, collision-free sets over similar state-of-the-art algorithms such as IRIS and SFC.

Funding This work was supported in part by DARPA under the Grant no. D18AP00069.

Code Availability The computer codes employed to perform the numerical simulations and flight tests presented in this paper will be disclosed at
http://lafflitto.com/guidance_code.html upon acceptance for publication of this paper.

Author Contributions Mr. J. A. Marshall primarily contributed to the literature review presented in Section 2, the coding of the trajectory planner presented in Section 4.2, the numerical simulations presented in Sections 7 and 9, and the flight tests presented in Section 8.

Dr. R. B. Anderson primarily contributed to the literature review presented in Section 2, the coding of the path planner presented in Section 4.1, some of the numerical simulations presented in Section 7, and some of the flight tests presented in Section 8.

Mr. W.-Y. Chien contributed to Section 5 and the coding of the navigation system employed in this work. Dr. E. N. Johnson supervised and contributed to the work performed by Mr. Chien.

Dr. A. L'Afflitto coordinated the work effort and primarily contributed to writing Section 1 and editing all other sections of this paper with the assistance of Mr. Marshall and Dr. Anderson.

Declarations

Ethics Approval This content of this paper has been prepared in compliance with the ethics rules of the authors' home institutions and Springer.

Consent to Participate and for Publication All authors actively contributed to the preparation of the content of this paper and consent to its publication in its present form.

References

1. Al Marzouqi, M., Jarvis, R.A.: Robotic covert path planning; A survey. In: IEEE Conference on Robotics, Automation and Mechatronics, Beijing, China, pp. 77–82 (2011)
2. Allgöwer, F., Zheng, A.: Nonlinear Model Predictive Control. Progress in Systems and Control Theory. Birkhäuser, Basel (2000)
3. Alonso-Mora, J., Baker, S., Rus, D.: Multi-robot formation control and object transport in dynamic environments via constrained optimization. Int. J. Robot. Res. **36**(9), 1000–1021 (2017)
4. Anderson, J.: Computational Fluid Dynamics. Computational Fluid Dynamics: The Basics with Applications. McGraw-Hill Education, Upper Saddle Hill (1995)
5. Andert, F.: Drawing stereo disparity images into occupancy grids; Measurement model and fast implementation. In: International Conference on Intelligent Robots and Systems, St. Louis, MO, pp. 5191–5197. IEEE (2009)
6. Ariens, D., Diehl, M., Ferreau, H.J., Houska, B., Logist, F., Quirynen, R., Vukov, M. ACADO Toolkit User's Manual, 1.2.1. KU Leuven, Leuven (2014)
7. Babel, L.: Coordinated target assignment and UAV path planning with timing constraints. J. Intell. Robot. Syst. **94**(3-4), 857–869 (2019)
8. Bemporad, A., Patrinos, P.: Simple and certifiable quadratic programming algorithms for embedded linear model predictive control. IFAC Proc. Vol. **45**(17), 14–20 (2012). https://doi.org/10.3182/20120823-5-NL-3013.00009. IFAC Conference on Nonlinear Model Predictive Control
9. Ben-Asher, J.: Optimal Control Theory with Aerospace Applications. AIAA education series American Institute of Aeronautics and Astronautics (2010)
10. Bernstein, D.S. Matrix Mathematics; Theory, Facts, and Formulas, 2nd edn. Princeton University Press, Princeton (2009)
11. Blackmore, L., Ono, M., Williams, B.C.: Chance-constrained optimal path planning with obstacles. IEEE Trans. Robot. **27**(6), 1080–1094 (2011)
12. Bohlin, R., Kavraki, L.E.: Path planning using lazy PRM. In: IEEE International Conference on Robotics and Automation, Paris, France, vol. 1, pp. 521–528 (2000)
13. Boyd, S., El Ghaoui, L., Feron, E., Balakrishnan, V.: Linear matrix inequalities in system and control theory. SIAM, Philadelphia (1994)
14. Boyd, S.P., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2004)

15. Bresenham, J.E.: Algorithm for computer control of a digital plotter. IBM Syst. J. **4**(1), 25–30 (1965)
16. Buijs, J., Ludlage, J., Brempt, W.V., Moor, B.D.: Quadratic programming in model predictive control for large scale systems. IFAC Proc. Vol. **35**(1), 301–306 (2002). https://doi.org/10.3182/20020721-6-ES-1901.00300. IFAC World Congress
17. Chaudhry, A., Misovec, K., D'Andrea, R.: Low observability path planning for an unmanned air vehicle using mixed integer linear programming. In: IEEE Conference on Decision and Control, vol. 4, pp. 3823–3829 (2004). https://doi.org/10.1109/CDC.2004.1429334
18. Chen, X., Chen, X.: The UAV dynamic path planning algorithm research based on voronoi diagram. In: Chinese Control and Decision Conference, Changsha, China, pp. 1069–1071. IEEE (2014)
19. Chien, W.Y.: Stereo-camera occupancy grid mapping. Master's thesis, Aerospace Engineering (2020)
20. Coutinho, W.P., Battarra, M., Fliege, J.: The unmanned aerial vehicle routing and trajectory optimisation problem, a taxonomic review. Comput. Ind. Eng. **120**, 116–128 (2018)
21. Cui, J.Q., Lai, S., Dong, X., Chen, B.M.: Autonomous navigation of uav in foliage environment. J. Intell. Robot. Syst. **84**(1), 259–276 (2016). https://doi.org/10.1007/s10846-015-0292-1
22. Davis, J., Perhinschi, M., Wilburn, B., Karas, O.: Development of a modified Voronoi algorithm for UAV path planning and obstacle avoidance. In: AIAA Guidance, Navigation, and Control Conference, Minneapolis, MN, pp. 1–11 (2012). https://doi.org/10.2514/6.2012-4904
23. De Filippis, L., Guglieri, G.: Advanced graph search algorithms for path planning of flight vehicles. pp. 157–192. Intech. https://doi.org/10.5772/37033 (2012)
24. Deits, R., Tedrake, R.: Computing large convex regions of obstacle-free space through semidefinite programming. In: Algorithmic Foundations of Robotics XI, pp. 109–124. Springer (2015)
25. Deits, R., Tedrake, R.: Efficient mixed-integer planning for UAVs in cluttered environments. In: International Conference on Robotics and Automation, pp. 42–49. IEEE (2015)
26. Deits, R.L.H., Tedrake, R.: IRIS-distro. https://github.com/rdeits/iris-distro.git. Last access; 01/20/2021 (2021)
27. Fujisawa, K., Kojima, M., Nakata, K., Yamashita, M.: SDPA SemiDefinite Programming Algorithm) user's manual - version 6.2.0. In: Research Reports on Mathematical and Computing Sciences Series B; Operations Research, pp. 1–32 (2002)
28. Geraerts, R., Schager, E.: Stealth-based path planning using corridor maps. In: Computer Animation and Social Agents (2010)
29. Han, L., Gao, F., Zhou, B., Shen, S.: FIESTA: Fast incremental Euclidean distance fields for online motion planning of aerial robots. In: International Conference on Intelligent Robots and Systems, pp. 4423–4430 (2019). https://doi.org/10.1109/IROS40897.2019.8968199
30. Harabor, D.D., Grastien, A.: Online graph pruning for pathfinding on grid maps. In: AAAI Conference on Artificial Intelligence, San Francisco, CA, pp. 1114–1119 (2011)
31. Heller, D.E.: Direct and iterative methods for block tridiagonal linear systems. Ph.D. thesis, Carnegie-Mellon University (1977)
32. Houska, B., Ferreau, H.J., Diehl, M.: ACADO toolkit. https://acado.github.io/ (2009)
33. Huang, H., Savkin, A.V., Ni, W.: A method for covert video surveillance of a car or a pedestrian by an autonomous aerial drone via trajectory planning. In: International Conference on Control, Automation and Robotics, pp. 446–449. IEEE, Singapore (2020)
34. Isaacs, R.: Differential Games; A Mathematical Theory with Applications to Warfare and Pursuit, Control and Optimization. Mineola, NY (1999)
35. Jensen, S.P., Gray, S.J., Hurst, J.L.: How does habitat structure affect activity and use of space among house mice? Anim. Behav. **66**(2), 239–250 (2003)
36. Johnson, S.G.: The NLopt nonlinear-optimization package. http://github.com/stevengj/nlopte (2020)
37. Kamel, M., Burri, M., Siegwart, R.: Linear vs nonlinear MPC for trajectory tracking applied to rotary wing micro aerial vehicles. IFAC-PapersOnLine **50**(1), 3463–3469 (2017). https://doi.org/10.1016/j.ifacol.2017.08.849. IFAC World Congress
38. Kelly, M.: An introduction to trajectory optimization; How to do your own direct collocation. SIAM Rev. **59**(4), 849–904 (2017). https://doi.org/10.1137/16M1062569
39. Koenig, S.: Likhachev, M.: D^* lite. In: National conference on Artificial intelligence, vol. 15, pp. 476-483. AAAI, Alberta (2002)
40. Koenig, S., Likhachev, M.: Fast replanning for navigation in unknown terrain. IEEE Trans. Robot. **21**(3), 354–363 (2005)
41. Kögel, M., Findeisen, R.: A fast gradient method for embedded linear predictive control. IFAC Proc. Vol. **44**(1), 1362–1367 (2011). https://doi.org/10.3182/20110828-6-IT-1002.03322. IFAC World Congress
42. Kothari, M., Postlethwaite, I.: A probabilistically robust path planning algorithm for UAVs using rapidly-exploring random trees. J. Intell. Robot. Syst. **71**(2), 231–253 (2013)
43. Kreisselmeier, G., Steinhauser, R.: Systematic control design by optimizing a vector performance index. IFAC Proc. Vol. **12**(7), 113–117 (1979). https://doi.org/10.1016/S1474-6670(17)65584-8. IFAC Symposium on computer Aided Design of Control Systems
44. Kwon, W., Han, S.: Receding Horizon Control; Model Predictive Control for State Models. Advanced Textbooks in Control and Signal Processing. Springer, London (2005)
45. L'Afflitto, A.: Differential games, continuous Lyapunov functions, and stabilisation of non-linear dynamical systems. IET Control Theory Appl. **11**, 2486–2496 (2017)
46. L'Afflitto, A.: A Mathematical Perspective on Flight Dynamics and Control. Springer, London (2017)
47. L'Afflitto, A., Anderson, R.B., Mohammadi, K.: An introduction to nonlinear robust control for unmanned quadrotor aircraft. IEEE Control. Syst. Mag. **38**(3), 102–121 (2018)
48. Landry, B., Deits, R., Florence, P.R., Tedrake, R.: Aggressive quadrotor flight through cluttered environments using mixed integer programming. In: IEEE International Conference on Robotics and Automation, pp. 1469–1475 (2016). https://doi.org/10.1109/ICRA.2016.7487282
49. Latombe, J.C.: Robot Motion Planning, vol. 124. Springer, Berlin (2012)
50. Li, K., Wang, K., Zhang, K., Chen, B.M.: Aggressive maneuvers of a quadrotor MAV based on composite nonlinear feedback control. In: IEEE International Conference on Advanced Intelligent Mechatronics, pp. 513–518 (2016). https://doi.org/10.1109/AIM.2016.7576819
51. Liu, S., Atanasov, N., Mohta, K., Kumar, V.: Search-based motion planning for quadrotors using linear quadratic minimum time control. In: International Conference on Intelligent Robots and Systems, pp. 2872–2879. IEEE, Vancouver (2017)

52. Liu, S., Watterson, M., Mohta, K.: Decomputil. https://github.com/sikang/DecompUtil. Last access; 01/20/2021 (2021)
53. Liu, S., Watterson, M., Mohta, K., Sun, K., Bhattacharya, S., Taylor, C.J., Kumar, V.: Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-D complex environments. IEEE Robot. Autom. Lett. **2**(3), 1688–1695 (2017)
54. Liu, S., Watterson, M., Tang, S., Kumar, V.: High speed navigation for quadrotors with limited onboard sensing. In: IEEE International Conference on Robotics and Automation, Stockholm, Sweden, pp. 1484–1491 (2016)
55. Maciejowski, J.: Predictive Control: With Constraints. Prentice Hall, Upper Saddle Hill (2002)
56. Madridano, A., Al-Kaff, A., Martin, D.: 3D trajectory planning method for UAVs swarm in building emergencies. Sensors **20**(3), 642 (2020)
57. Marshall, J.A., Anderson, R.B., L'Afflitto, A.: A guidance system for a tactical autonomous unmanned aerial vehicle. https://youtu.be/6F5_QYwNJrE. Last accessed 09/04/2020 (2020)
58. Marzouqi, M., Jarvis, R.A.: Covert path planning for autonomous robot navigation in known environments. In: Australasian Conference on Robotics and Automation. Citeseer, Brisbane, Australia (2003)
59. Masehian, E., Amin-Naseri, M.: A Voronoi diagram-visibility graph-potential field compound algorithm for robot path planning. J. Robot. Syst. **21**(6), 275–300 (2004)
60. Mattingley, J., Boyd, S.: CVXGen. https://cvxgen.com/docs/index.html. Last access; 04/19/2021 (2021)
61. Murty, K.G., Yu, F.T.: Linear complementarity linear and nonlinear programming, vol. 3. Heldermann, Ann Arbor (1988)
62. Nägeli, T., Meier, L., Domahidi, A., Alonso-Mora, J., Hilliges, O.: Real-time planning for automated multi-view drone cinematography. Trans. Graph. **36**(4). https://doi.org/10.1145/3072959.3073712 (2017)
63. Nieuwenhuisen, M., Behnke, S.: Search-based 3D planning and trajectory optimization for safe micro aerial vehicle flight under sensor visibility constraints. In: International Conference on Robotics and Automation, pp. 9123–9129 (2019). https://doi.org/10.1109/ICRA.2019.8794086
64. Niu, H., Lu, Y., Savvaris, A., Tsourdos, A.: An energy-efficient path planning algorithm for unmanned surface vehicles. Ocean Eng. **161**, 308–321 (2018)
65. Nocedal, J., Bonnans, J.F., Mikosch, T.V., Wright, S.: Numerical Optimization. Springer, Berlin (2006)
66. Noreen, I., Khan, A., Ryu, H., Doh, N.L., Habib, Z.: Optimal path planning in cluttered environment using RRT*-AB. Intell. Serv. Robot. **11**(1), 41–52 (2018)
67. Nuske, S., Choudhury, S., Jain, S., Chambers, A., Yoder, L., Scherer, S., Chamberlain, L., Cover, H., Singh, S.: Autonomous exploration and motion planning for an unmanned aerial vehicle navigating rivers. J. Field Robot. **32**(8), 1141–1162 (2015). https://doi.org/10.1002/rob.21596
68. Oleynikova, H., Baehnemann, R., Fehr, M., Millane, A., Lim, J., Ratnesh, M., Rosinol, T.: mav_voxblox_planning.https://github.com/ethz-asl/mav_voxblox_planning (2019)
69. Oleynikova, H., Lanegger, C., Taylor, Z., Pantic, M., Millane, A., Siegwart, R., Nieto, J.: An open-source system for vision-based micro-aerial vehicle mapping, planning, and flight in cluttered environments. J. Field Robot. **37**(4), 642–666 (2020). https://doi.org/10.1002/rob.21950
70. Pehlivanoglu, Y.V.: A new vibrational genetic algorithm enhanced with a Voronoi diagram for path planning of autonomous UAV. Aerosp. Sci. Technol. **16**(1), 47–55 (2012)
71. Pestana, J., Maurer, M., Muschick, D., Hofer, M., Fraundorfer, F.: Overview obstacle maps for obstacle-aware navigation of autonomous drones. J. Field Robot. **36**(4), 734–762 (2019). https://doi.org/10.1002/rob.21863
72. Potra, F.A., Wright, S.J.: Interior-point methods. J. Comput. Appl. Math. **124**(1), 281–302 (2000). https://doi.org/10.1016/S0377-0427(00)00433-7
73. Primatesta, S., Guglieri, G., Rizzo, A.: A risk-aware path planning strategy for UAVs in urban environments. J. Intell. Robot. Syst. **95**(2), 629–643 (2019)
74. Prodan, I., Olaru, S., Bencatel, R., ao Borges de Sousa, J., Stoica, C., Niculescu, S.I.: Receding horizon flight control for trajectory tracking of autonomous aerial vehicles. Control Eng. Prac **21**(10), 1334–1349 (2013). https://doi.org/10.1016/j.conengprac.2013.05.010
75. Radmanesh, M., Kumar, M., Guentert, P.H., Sarim, M.: Overview of path-planning and obstacle avoidance algorithms for UAVs; A comparative study. Unmanned Syst. **6**(2), 95–118 (2018)
76. Rao, A.V.: Trajectory optimization: A survey. In: Optimization and Optimal Control in Automotive Systems, pp. 3–21. Springer (2014)
77. Richards, A.: Fast model predictive control with soft constraints. Eur. J. Control. **25**, 51–59 (2015). https://doi.org/10.1016/j.ejcon.2015.05.003
78. Richter, C., Bry, A., Roy, N.: Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments. In: Robotics Research, pp. 649–666. Springer (2016)
79. Richter, S., Jones, C.N., Morari, M.: Real-time input-constrained MPC using fast gradient methods. In: IEEE Conference on Decision and Control, pp. 7387–7393 (2009). https://doi.org/10.1109/CDC.2009.5400619
80. Richter, S., Morari, M., Jones, C.N.: Towards computational complexity certification for constrained MPC based on lagrange relaxation and the fast gradient method. In: IEEE Conference on Decision and Control and European Control Conference, pp. 5223–5229 (2011). https://doi.org/10.1109/CDC.2011.6160931
81. Sahingoz, O.K.: Generation of Bezier curve-based flyable trajectories for multi-UAV systems with parallel genetic algorithm. J. Intell. Robot. Syst. **74**(1-2), 499–511 (2014)
82. Sanchez-Lopez, J.L., Wang, M., Olivares-Mendez, M.A., Molina, M., Voos, H.: A real-time 3d path planning solution for collision-free navigation of multirotor aerial robots in dynamic environments. J. Intell. Robot. Syst. **93**(1), 33–53 (2019). https://doi.org/10.1007/s10846-018-0809-5
83. Sethian, J.: Level Set Methods and Fast Marching Methods; Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science. Cambridge University Press, Cambridge (1999)
84. de Souza, J.P.C., Marcato, A.L.M., de Aguiar, E.P., Juca, M.A., Teixeira, A.M.: Autonomous landing of UAV based on artificial neural network supervised by fuzzy logic. J. Control Autom. Electric. Syst. **40**(4), 522–531 (2019). https://doi.org/10.1007/s40313-019-00465-y
85. Spedicato, S., Notarstefano, G., Bülthoff, H.H., Franchi, A.: Aggressive maneuver regulation of a quadrotor UAV. In: Inaba, M., Corke, P. (eds.) Robotics Research; The 16th International Symposium ISRR, pp. 95–112. Springer (2016). https://doi.org/10.1007/978-3-319-28872-7_6
86. Spitzer, A., Yang, X., Yao, J., Dhawale, A., Goel, K., Dabhi, M., Collins, M., Boirum, C., Michael, N.: Fast and agile vision-based flight with teleoperation and collision avoidance on a multirotor. In: International Symposium on Experimental Robotics, pp. 524–535. Springer (2018)
87. Sun, W., Theodorou, E.A., Tsiotras, P.: Game theoretic continuous time differential dynamic programming. In: American Control Conference, pp. 5593–5598 (2015). https://doi.org/10.1109/ACC.2015.7172215

88. Sun, W., Tsiotras, P.: Pursuit evasion game of two players under an external flow field. In: American Control Conference, pp. 5617–5622 (2015). https://doi.org/10.1109/ACC.2015.7172219
89. Tal, E., Karaman, S.: Accurate tracking of aggressive quadrotor trajectories using incremental nonlinear dynamic inversion and differential flatness.IEEE Trans. Control Syst. Technol., 1–16. https://doi.org/10.1109/TCST.2020.3001117 (2020)
90. Tang, L., Wang, H., Liu, Z., Wang, Y.: A real-time quadrotor trajectory planning framework based on B-spline and nonuniform kinodynamic search. J. Field Robot. **38**(3), 452–475 (2021). https://doi.org/10.1002/rob.21997
91. Thrun, S., Burgard, W., Fox, D., Arkin, R.: Probabilistic Robotic. MIT Press, Boston (2005)
92. Tordesillas, J., Lopez, B.T., How, J.P.: FASTER: Fast and safe trajectory planner for flights in unknown environments. In: International Conference on Intelligent Robots and Systems, pp. 1934–1940 (2019). https://doi.org/10.1109/IROS40897.2019.8968021
93. Tsai, J.S.H., Huang, C.C., Guo, S.M., Shieh, L.S.: Continuous to discrete model conversion for the system with a singular system matrix based on matrix sign function. Appl. Math. Model. **35**(8), 3893–3904 (2011). https://doi.org/10.1016/j.apm.2011.02.009
94. U.S. Army: An Infantyman's guide to combat in built-up areas. Technical report (2013)
95. Vasile, M., De Pascale, P., Casotto, S.: On the optimality of a shape-based approach based on pseudo-equinoctial elements. Acta Astronaut. **61**(1), 286–297 (2007). https://doi.org/10.1016/j.actaastro.2007.01.017
96. Verschueren, R., Frison, G., Kouzoupis, D., van Duijkeren, N., Zanelli, A., Quirynen, R., Diehl, M.: ACADOS toolkit. https://github.com/acados/acados (2018)
97. Verschueren, R., Frison, G., Kouzoupis, D., van Duijkeren, N., Zanelli, A., Quirynen, R., Diehl, M.: Towards a modular software package for embedded optimization. In: IFAC Conference on Nonlinear Model Predictive Control, vol. 51, pp. 374–380 (2018). https://doi.org/10.1016/j.ifacol.2018.11.062
98. Votion, J., Cao, Y.: Diversity-based cooperative multivehicle path planning for risk management in costmap environments. IEEE Trans. Ind. Electron. **66**(8), 6117–6127 (2018)
99. Wallace, R.J., Loffi, J.M.: How law enforcement unmanned aircraft systems (UAS) could improve tactical response to active shooter situations: The case of the 2017 Las Vegas shooting. Int. J. Aviat. Aeron. Aerosp. **4**(4), 7 (2017)
100. Wang, Y., Boyd, S.: Fast model predictive control using online optimization. IEEE Trans. Control Syst. Technol. **18**(2), 267–278 (2009)
101. Wang, Y., Boyd, S.: Fast MPC. https://web.stanford.edu/boyd/papers/fast_mpc.html (2009)
102. Wang, Z., Zhou, X., Xu, C., Chu, J., Gao, F.: Alternating minimization based trajectory generation for quadrotor aggressive flight. IEEE Robot. Autom. Lett. **5**(3), 4836–4843 (2020). https://doi.org/10.1109/LRA.2020.3003871
103. Watterson, M., Liu, S., Sun, K., Smith, T., Kumar, V.: Trajectory optimization on manifolds with applications to quadrotor systems. Int. J. Robot. Res. **39**(2-3), 303–320 (2020). https://doi.org/10.1177/0278364919891775
104. Wills, A.G., Heath, W.P.: Barrier function based model predictive control. Automatica **40**(8), 1415–1422 (2004). https://doi.org/10.1016/j.automatica.2004.03.002
105. Wright, S.J.: Efficient convex optimization for linear MPC. In: Raković, S.V., Levine, W.S. (eds.) Handbook of Model Predictive Control, pp. 287–303. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-319-77489-3_13
106. Yang, L., Qi, J., Song, D., Xiao, J., Han, J., Xia, Y.: Survey of robot 3D path planning algorithms. J. Control Sci. Eng. (2016)
107. Yang, Z., Fang, Z., Li, P.: Bio-inspired collision-free 4D trajectory generation for UAVs using tau strategy. J. Bionic Eng. **13**(1), 84–97 (2016)
108. Zhang, X., Chen, J., Xin, B., Fang, H.: Online path planning for UAV using an improved differential evolution algorithm. IFAC Proc. **44**(1), 6349–6354 (2011)
109. Zhang, Y., Cohen, J., Owens, J.D.: Fast tridiagonal solvers on the GPU. ACM Sigplan Notices **45**(5), 127–136 (2010)
110. Zhang, Z., Wu, J., Dai, J., He, C.: A novel real-time penetration path planning algorithm for stealth UAV in 3D complex dynamic environment. IEEE Access **8**, 122,757–122,771 (2020). https://doi.org/10.1109/ACCESS.2020.3007496
111. Zhou, B., Gao, F., Wang, L., Liu, C., Shen, S.: Robust and efficient quadrotor trajectory generation for fast autonomous flight. IEEE Robot. Autom. Lett. **4**(4), 3529–3536 (2019)
112. Zhou, B., Pan, J., Gao, F., Shen, S.: RAPTOR: Robust and perception-aware trajectory replanning for quadrotor fast flight. IEEE Trans. Robot., 1–18. https://doi.org/10.1109/TRO.2021.3071527 (2021)
113. Zylberberg, J., DeWeese, M.R.: How should prey animals respond to uncertain threats? Front. Comput. Neurosci. **5**, 20 (2011)

Julius A. Marshall received the B.S. degree in Aerospace Engineering from the University of Oklahoma in 2018. Since 2018, he has been a part of the Advanced Control Systems Lab under Dr. L'Afflitto's advisement. Mr. Marshall is pursuing a Ph.D. degree in Industrial and Systems Engineering at Virginia Tech, where he is specializing in robotics. His current research interests include robust control, optimal control, guidance and navigation, and adaptive control for autonomous shipboard landing. In 2019, he received the SMART Scholar award.

Robert B. Anderson received the B.S. (2017) in Engineering Physics and the M.S. (2019) in Aerospace Engineering from the University of Oklahoma. In 2021, he received the Ph.D. in Industrial Engineering from Virginia Tech. Currently, he works in the flight control and navigation group at Collins Aerospace researching high integrity vision-based autoland solutions for aircraft.

Wen-Yu Chien is a Robotics Engineer at Near Earth Autonomy and integrates hardware and software for Unmanned Aerial Systems. He graduated from Penn State with his M.S. degree in Aerospace Engineering in August 2020. He was in Dr. Eric Johnson's PSU UAS Research Laboratory(PURL) and working on voxel grid mapping by using a stereo camera and a VIO sensor that allows a quadcopter to map an unknown and GPS-denied environment.

Eric N. Johnson is a Professor of Aerospace Engineering at Pennsylvania State University. He received a B.S. degree from University of Washington, M.S. degrees from MIT and The George Washington University, and a Ph.D. from Georgia Tech. He also has five years of industry experience working at Lockheed Martin and Draper. As faculty since 2001, he has performed research in unmanned aircraft fault-tolerant control, aided inertial navigation, and autonomy at Georgia Tech and Penn State. This work has included the first air-launch of a hovering aircraft, automatic flight of a helicopter with simulated frozen actuators, and visionbased air-to-air tracking. His most recent work has included automatic low altitude high speed flight of helicopters, indoor and outdoor vision-aided inertial navigation, and methods for sensing and avoiding other aircraft. The mission of this work has been to enable unmanned aircraft systems to contribute to society.

Andrea L'Afflitto received the B.S. degree in Aerospace Engineering and the M.S. degree in Aerospace Engineering and Astronautics from the University of Napoli "Federico II," Italy, in 2004 and 2006, respectively, the M.S. degree in Mathematics from Virginia Tech in 2010, and the Ph.D. in Aerospace Engineering from Georgia Tech in 2015. In 2019, he joined the Grado Department of Industrial and Systems Engineering at Virginia Tech as an assistant professor, from 2015 to 2019, he worked at the School of Aerospace and Mechanical Engineering of the University of Oklahoma as an assistant professor, and from 2008 to 2009 he worked as System Engineer at the German Aerospace Agency (DRL) in Cologne, Germany. Dr. L'Afflitto's current research interests include nonlinear robust control, optimal control, and control of unmanned aerial systems. In 2018, he received the DARPA Young Faculty Award. Dr. L'Afflitto is a member of the IEEE Aerospace Controls Technical Committee.