Chapter 3
Autonomous Multi-rotor Unmanned Aerial Vehicles for Tactical Coverage

Julius A. Marshall, Paul Binder, and Andrea L'Afflitto

3.1 Introduction

In recent years, the use of small multi-rotor unmanned aerial vehicles (UAVs), such as quadcopters, has become particularly appealing to first responders, law enforcement agencies, and other emergency teams to collect information with high spatial and temporal resolutions about unknown and potentially dangerous environments. The complexity of employing UAVs in these tasks is substantially determined by the need for these vehicles to operate in complex environments where external sources of information, such as the Global Positioning System (GPS) or alternative global navigation satellite systems, either underperform or are unavailable.

This chapter presents an original guidance system for autonomous multi-rotor UAVs, such as quadcopters, equipped with forward-facing cameras to create maps of unknown, potentially hostile, GPS-denied environments, while flying at very low altitudes. Several unique features distinguish the proposed system. Firstly, to operate in dangerous environments, this guidance system allows the aircraft to implement several tactics to minimize its exposure to potential threats. In this chapter, these tactics include coasting obstacles so that potential threats may only detect and intercept the UAV from limited directions. Furthermore, the UAV coasts obstacles to reduce the ability of systems based on sound reflection to localize the UAV. If no obstacle can be used to shelter the UAV, then the vehicle accelerates toward its next goal point to minimize the time spent in unsheltered areas. The proposed guidance system also allows the implementation of alternative tactics, such as maximizing flight time

J. A. Marshall · P. Binder · A. L'Afflitto (✉)
Virginia Tech, Blacksburg, VA, USA
e-mail: a.lafflitto@vt.edu

J. A. Marshall
e-mail: mjulius@vt.edu

P. Binder
e-mail: paulbinder@vt.edu

A. L'Afflitto et al. (eds.), *Control of Autonomous Aerial Vehicles*, Advances in Industrial Control, https://doi.org/10.1007/978-3-031-39767-7_3

in already explored areas. An additional unique feature of the proposed guidance system is that it can operate with any vision-based navigation system, such as the one presented in [1, 2], tasked with localizing the UAV relative to the environment and reconstructing the environment in the form of voxel maps. In the voxel maps employed in this work, the size of all voxels is user-defined and each voxel captures whether the corresponding portion of the environment is still unexplored or, with some user-defined confidence level, explored and either occupied by some obstacle or unoccupied. Finally, the proposed guidance system implements an explore-then-exploit approach [3], since its path planner a suitable strategy to cover a user-defined set, and its trajectory planner enables the mapping process by exploiting the vehicle's dynamics. However, a key difference between the proposed guidance system for mapping and the majority of existing ones based on the explore-then-exploit technique [4] is that, in this work, the map is created by employing forward-facing, and not down-facing, cameras.

The proposed path planner includes both an original algorithm to select multiple goal points for the UAV to visit in order to cover the environment and an optimization-based algorithm to produce sequences of waypoints connecting consecutive goal points. The goal point selection algorithm partitions the set to be covered in parallelepipeds of user-defined size and containing user-defined numbers of occupied voxels, and then sets the next goal point as the barycenter of the nearest partition, the barycenter of the largest partition, or a convex combination thereof. Thus, the proposed goal point selection algorithm allows the user to choose a more systematic approach or greedier approach, according to needs. For instance, a greedier approach may imply a longer flight time, which, in the presence of potential threats to the UAV, may result in an unsafe approach. The lifelong planning A^* (LPA^*) algorithm underlying the proposed path planner generates a sequence of waypoints for the UAV to connect consecutive goal points; the LPA^* algorithm has been chosen for its ability to rapidly generate paths [5]. Furthermore, in the proposed framework, this search algorithm can be seamlessly replaced by alternative algorithms operating over graphs such as D^* or D^* lite [6, 7]. In order to induce tactical behaviors, the cost-to-come function characterizing the proposed path planner includes a weighing function that steers the UAV toward obstacles to seek shelter. This weighing function is tunable by means of user-defined parameters that define both the maximum distance and the strength of the attraction exerted by obstacles on the UAV. Remarkably, alternative or additional criteria to generate more cautious trajectories, such as maximizing the distance traveled in already covered areas, can be implemented in the proposed path planning system by modifying the weighing function.

A fast model predictive control framework allows the proposed guidance system to generate real-time reference trajectories that are compatible with the UAV's dynamics and interpolate the waypoints produced by the path planner with user-defined precision. The cost function underlying the proposed trajectory planner captures the effort needed to control the UAV, the cost to reach the next waypoint, and the tactical advantage for the UAV to coast obstacles to seek shelter; user-defined parameters allow to tune the attractive effect of the obstacles' set on the UAV. The proposed guidance system accounts for the UAV's nonlinear dynamics, collision avoidance

constraints, attitude constraints, and saturation constraints on each propeller's thrust force. In particular, the UAV's dynamics are captured by the UAV's discrete-time output-feedback linearized equations of motion; output-feedback linearization allows to exploit the UAV's dynamics while leveraging a quadratic programming framework to solve the model predictive control problem in real time [2]. Collision avoidance constraints generated by an original algorithm in the form of convex sets that include the UAV, exclude all detected obstacle points, and are sufficiently large to produce reference trajectories over long time horizons. Constraints on the UAV's yaw angle are imposed by means of inequalities, and constraints on the UAV's pitch and roll angles are imposed by means of barrier functions embedded in the cost function. Finally, the boundary conditions underlying the proposed trajectory planner allow the user to define how closely waypoints need to be followed and the UAV's acceleration, while traversing less safe areas.

The literature on the coverage problem for UAVs employed in agriculture, mining, and civil engineering, has grown rapidly in the past few years [4, 8–10]. However, the problem of covering unknown and potentially hostile environments by using UAVs is still under-explored. Among the few works in this area, it is worthwhile to recall [11], where a modified logistic map and a modulo tactic are used to generate unpredictable motion profiles [12], where a nonlinear programming-based approach is employed to design paths for teams of UAVs for continuous coverage and sustained situational awareness, and [13], where a path planner based on genetic algorithms was proposed for heterogeneous fleets of UAVs flying at low altitudes in dynamic and potentially dangerous environments. However, it is worthwhile to recall that both [12, 13] rely on a priori information about the threats to the UAV and, hence, are not recommended for completely unknown environments. Furthermore, [11–13] propose path planners only. However, not accounting for the UAV's dynamics reduces the ability to implement these systems on UAVs operating in cluttered environments, and, to succeed in these missions, fast trajectory planners and collision avoidance algorithms need to be included. For its ability to operate in environments, where the nature and the location of threats are unknown, and the integration of a fast path planner and a fast trajectory planner, the proposed guidance system advances the state of the art in the design of UAVs involved in mapping of unknown, contested areas.

This chapter is structured as follows. In Sect. 3.2, the notation is outlined. Successively, in Sect. 3.3, we present the path planner underlying the proposed guidance system for tactical coverage. In Sect. 3.4, we present the trajectory planning algorithm underlying the proposed guidance system. Section 3.5 presents the results of two sets of numerical simulations aimed at showing the applicability of the proposed approach and its ability to efficiently cover large areas both in a tactical and in a reckless manner. Finally, in Sect. 3.6, we draw conclusions and recommend future work directions.

3.2 Definitions and Notation

Let $\mathbb{N}$ denote the *set of positive integers*, $\mathbb{R}$ the *set of real numbers*, $\mathbb{R}^n$ the *set of* $n \times 1$ *real column vectors*, $\mathbb{R}^{n\times m}$ the *set of* $n \times m$ *real matrices*, $\mathbb{R}^n_+$ the *positive orthant of* $\mathbb{R}^n$, and $\overline{\mathbb{R}}^n_+$ the *nonnegative orthant of* $\mathbb{R}^n$. The *interior of the set* $\mathcal{S} \subset \mathbb{R}^n$ is denoted by $\mathring{\mathcal{S}}$, the *boundary of* $\mathcal{S} \subset \mathbb{R}^n$ is denoted by $\partial\mathcal{E}$, and the *closure of* $\mathcal{S}$ is denoted by $\overline{\mathcal{S}}$. Given $x, y \in \mathbb{R}^n$, if each component of x is larger, not smaller, not larger, or smaller than the corresponding component of y, then we write $x >> y$, $x \geq\geq y$, $x \leq\leq y$, or $x << y$, respectively. The *open ball of radius* $\rho > 0$ *centered at* $x \in \mathbb{R}^n$ is denoted by $\mathcal{B}_\rho(x)$. Given the symmetric, negative-definite matrix $P \in \mathbb{R}^{3\times3}$, $r \in \mathbb{R}^3$, and $c < 0$, $\overline{\mathcal{E}}(P, r, c) \triangleq \{w \in \mathbb{R}^3 : (w - r)^T P(w - r) - c \geq 0\}$ captures a *closed ellipsoid*.

The *transpose of* $B \in \mathbb{R}^{n\times m}$ is denoted by B^{T}. 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 formed by $a_i \in \mathbb{R}$, $i = 1, \ldots, p$, is denoted by $A = \mathrm{diag}(a_1, \ldots, a_p)$, and the block-diagonal matrix formed by $M_i \in \mathbb{R}^{n_i\times n_i}$ is denoted by $M = \mathrm{blockdiag}\left(M_1, \ldots, M_p\right)$. The *distance between the point* $x \in \mathbb{R}^n$ *and the set* $\mathcal{S}$ is denoted by $\mathrm{d}_2(x, \mathcal{S})$ [14, p. 16]. We write $\|\cdot\|$ for the *Euclidean vector norm* and the corresponding *equi-induced matrix norm* [15, Definition 9.4.1].

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 first- and second-time derivatives of the smooth function $y : [0, \infty) \to \mathbb{R}^n$ are denoted by $\dot{y}(t)$, $t \geq 0$, and $\ddot{y}(t)$, and the kth-order derivative of $y(\cdot)$, $k \geq 3$, is denoted by $y^{(k)}(\cdot)$.

User-defined parameters are denoted by $\mu_a \in \mathbb{R}$, $a \in \{1, \ldots, 24\}$. All vectors are expressed with respect to the orthonormal inertial reference frame $\mathbb{I} \triangleq \{O; X, Y, Z\}$ centered at O and with axes $X, Y, Z \in \mathbb{R}^3$ defined so that the gravitational force acts along the $-Z$ axis.

3.3 Path Planning System for Tactical Coverage

3.3.1 Overview

The proposed guidance system computes both a reference path and a reference trajectory for a UAV quadcopter so that its camera-based navigation system maps a user-defined connected set, that is, classifies the voxels covering this set as occupied or unoccupied, and reconstructs a map of the obstacles in this set. To meet this goal, the UAV's reference path is computed iteratively by identifying a sequence of goal points, and then by finding a sequence of waypoints joining consecutive goal points. In particular, at the beginning of the mission, all voxels covering the set to be

mapped are considered as unexplored, and as the set is being mapped, unexplored voxels are grouped into partitions, that is, parallelepipeds whose minimum edge is user-defined. At each iteration, the UAV's goal point is chosen as the barycenter of the partition containing the largest number of unexplored voxels, or as the barycenter of the nearest partition containing a user-defined minimum number of unexplored voxels, or as a convex combination of these two points. Setting the barycenter of the partition containing the largest number of unexplored voxels as the goal point for a given iteration not only allows to classify a large number of voxels but also enforces a reckless strategy since the UAV may be drawn toward unsafe environments. Conversely, setting the barycenter of the nearest partition not only allows the classification of a small number of voxels but also enables a more cautious tactic by preventing the UAV from quickly reaching far, unexplored areas. Having set the goal point, the reference path is computed as the solution of an optimization problem that captures the UAV's need to reach the goal point while enabling additional tactical strategies to exhibit a tactical behavior; examples of such strategies include coasting the obstacles' set or flying faster when sufficiently far from the obstacles' set. The proposed path planner does not account for the UAV's dynamics, but models the vehicle as a point mass able to move across adjacent unoccupied voxels. The UAV's dynamics and, hence, the dynamics of the onboard cameras' focal axis, is accounted for by the trajectory planner presented in Sect. 3.4 below.

Although the proposed path planner is designed to lead the UAV through the voxels containing the goal points, the proposed coverage strategy does not require meeting this objective. The voxel containing a goal point merely needs to be detected by the onboard navigation system and classified either as occupied or as unoccupied. Attempting to classify all voxels in the user-defined connected set in a systematic manner, as it occurs for numerous coverage algorithms [4, 16–18], may induce a reckless behavior in the UAV.

3.3.2 Notation

A *voxel* is a rectangular parallelepiped of user-defined dimensions, and let the connected set $\mathcal{V} \subset \mathbb{R}^3$ denote the union of $n_{\mathcal{V}} \in \mathbb{N}$ congruent voxels that cover the set to be mapped. Each voxel in $\mathcal{V}$ is denoted by its center, whose position is captured by $\hat{r}_p \in \mathcal{V}$, $p \in \{1, \ldots, n_{\mathcal{V}}\}$. The *explored indicator function* $\chi_{\text{explored}} : \mathbb{R}^3 \to \{0, 1\}$ is defined so that, if the voxel centered in $\hat{r}_p \in \mathcal{V}$ is unexplored, then $\chi_{\text{explored}}(\hat{r}_p) = 0$, and if the voxel centered in $\hat{r}_p \in \mathcal{V}$ is explored, then $\chi_{\text{explored}}(\hat{r}_p) = 1$. Furthermore, let $\mathcal{V}_{\text{unexplored}} \triangleq \{\hat{r}_p \in \mathcal{V} : \chi_{\text{explored}}(\hat{r}_p) = 0,\ p = 1, \ldots, n_{\mathcal{V}}\}$ and $\mathcal{V}_{\text{explored}} \triangleq \mathcal{V} \setminus \mathcal{V}_{\text{unexplored}}$ capture the *unexplored subset* of $\mathcal{V}$ and the *explored subset* of $\mathcal{V}$, respectively. The *occupied indicator function* $\chi_{\text{occupied}} : \mathbb{R}^3 \to \{0, 1\}$ is defined so that, if the voxel centered in $\hat{r}_p \in \mathcal{V}$, $p \in \{1, \ldots, n_{\mathcal{V}}\}$, is unoccupied, then $\chi_{\text{occupied}}(\hat{r}_p) = 0$, and if the voxel centered in $\hat{r}_p \in \mathcal{V}$ is occupied, then $\chi_{\text{occupied}}(\hat{r}_p) = 1$. Furthermore, let $\mathcal{V}_{\text{free}} \triangleq \{\hat{r}_p \in \mathcal{V} : \chi_{\text{occupied}}(\hat{r}_p) = 0,\ p = 1, \ldots, n_{\mathcal{V}}\}$ and $\mathcal{V}_{\text{occupied}} \triangleq \mathcal{V} \setminus \mathcal{V}_{\text{free}}$

capture the *unoccupied subset* of $\mathcal{V}$ and the *occupied subset* of $\mathcal{V}$, respectively. The *obstacles' set* $\mathcal{O} \triangleq \mathcal{V}_{\text{occupied}} \cap \mathcal{V}_{\text{explored}}$ is defined as those voxels that are both occupied and explored; the cardinality of $\mathcal{O}$ is denoted by $n_{\mathcal{O}}$.

Remark 3.1 The notions of occupied and explored voxels are unrelated. Indeed, at a given time instant, an occupied voxel may have not been explored by the UAV's navigation system. In this chapter, all unexplored voxels are designated as unoccupied until they are explored by the UAV's navigation system. This choice is motivated by the need for the path planning algorithm to return complete paths between any two unoccupied voxels, regardless of whether these voxels have already been explored or not.

The set $\mathcal{V}$ is partitioned in $n_{\mathcal{P}}$ rectangular parallelepipeds $\hat{\mathcal{P}}_a$, $a \in \{1, \ldots, n_{\mathcal{P}}\}$, such that $\hat{\mathcal{P}}_a$ is the union of voxels, $\bigcup_{a=1}^{n_{\mathcal{P}}} \hat{\mathcal{P}}_a = \mathcal{V}$, and $\hat{\mathcal{P}}_a \cap \hat{\mathcal{P}}_b = \{\emptyset\}$ for all $a, b \in \{1, \ldots, n_{\mathcal{P}}\}$ such that $a \neq b$. The *barycenter of* $\hat{\mathcal{P}}_a$, $a \in \{1, \ldots, n_{\mathcal{P}}\}$ is denoted by $\hat{r}_{\hat{\mathcal{P}}_a} \in \hat{\mathcal{P}}_a$. Lastly, the smallest parallelepiped containing $\mathcal{V}$ is denoted by $\hat{\mathcal{P}}_0$.

The subset of the voxel map contained in parallelipiped $\hat{\mathcal{P}}_i$, $i = 0, \ldots, n_{\mathcal{P}}$, is denoted by $\mathcal{V}_i \subseteq \hat{\mathcal{P}}_i$. The *set of explored voxels in* $\hat{\mathcal{P}}_i$, $i = 0, \ldots, n_{\mathcal{P}}$, is defined as $\mathcal{V}_{\text{explored},\hat{\mathcal{P}}_i} \triangleq \mathcal{V}_{\text{explored}} \cap \mathcal{V}_{\text{i}}$, the *set of unexplored voxels in* $\hat{\mathcal{P}}_i$ is defined as $\mathcal{V}_{\text{unexplored},\hat{\mathcal{P}}_i} \triangleq \mathcal{V}_{\text{unexplored}} \cap \mathcal{V}_i$, the *set of unoccupied voxels in* $\hat{\mathcal{P}}_i$ is defined as $\mathcal{V}_{\text{free},\hat{\mathcal{P}}_i} \triangleq \mathcal{V}_{\text{free}} \cap \mathcal{V}_i$, and the *set of occupied voxels in* $\hat{\mathcal{P}}_i$ is defined as $\mathcal{V}_{\text{occupied},\hat{\mathcal{P}}_i} \triangleq \mathcal{O} \cap \mathcal{V}_i$; for details, see Fig. 3.1. The number of explored voxels in $\hat{\mathcal{P}}_i$, $i = 0 \ldots, n_{\mathcal{P}}$, is denoted by $n_{\text{explored},\hat{\mathcal{P}}_i}$, the number of occupied voxels in $\hat{\mathcal{P}}_i$ is denoted by $n_{\text{occupied},\hat{\mathcal{P}}_i}$, the number of unexplored voxels in $\hat{\mathcal{P}}_i$ is denoted by $n_{\text{unexplored},\hat{\mathcal{P}}_i}$, and the length of the smallest edge of $\mathcal{P}_i$ $\ell_{\hat{\mathcal{P}}_i,\text{min}}$.

3.3.3 Goal Points Selection Algorithm

As discussed in Sect. 3.3.1, the UAV's reference path is computed iteratively by identifying a sequence of goal points determined by executing Algorithm 3.1. The first step of this algorithm is to partition $\hat{\mathcal{P}}_0 \subseteq \mathcal{V}$ in $n_{\mathcal{P}}$ rectangular parallelepipeds by employing Algorithm 3.2, which is discussed in detail in the following. Next, at each iteration of Algorithm 3.1, we find the partitions $\left\{\widetilde{\mathcal{P}}_1, \ldots, \widetilde{\mathcal{P}}_{\tilde{n}_{\mathcal{P}}}\right\} \subseteq \left\{\hat{\mathcal{P}}_1, \ldots, \hat{\mathcal{P}}_{n_{\mathcal{P}}}\right\}$ such that $\frac{n_{\text{unexplored},\widetilde{\mathcal{P}}_i}}{n_{\widetilde{\mathcal{P}}_i}} \geq 1 - \mu_1$, $i = 1, \ldots, \tilde{n}_{\mathcal{P}}$, where $\mu_1 \in (0, 1)$ captures the user-defined threshold on the ratio of explored voxels over the total number of voxels in $\hat{\mathcal{P}}_i$; thus, $\left\{\widetilde{\mathcal{P}}_1, \ldots, \widetilde{\mathcal{P}}_{\tilde{n}_{\mathcal{P}}}\right\}$ are sufficiently unexplored partitions of $\hat{\mathcal{P}}_0$. Finally, the UAV's *goal point* $\hat{r}_{\widetilde{\mathcal{P}},q}$, $q = 0, \ldots, n_{\text{g}}$, is defined as the barycenter of the partition $\widetilde{\mathcal{P}} \in \left\{\widetilde{\mathcal{P}}_1, \ldots, \widetilde{\mathcal{P}}_{\tilde{n}_{\mathcal{P}}}\right\}$ that is accessible to the UAV and contains the point

$$\mu_2 \hat{r}_{\mathcal{P}_{\text{prox}}} + (1 - \mu_2)\hat{r}_{\mathcal{P}_{\text{max}}}, \tag{3.1}$$

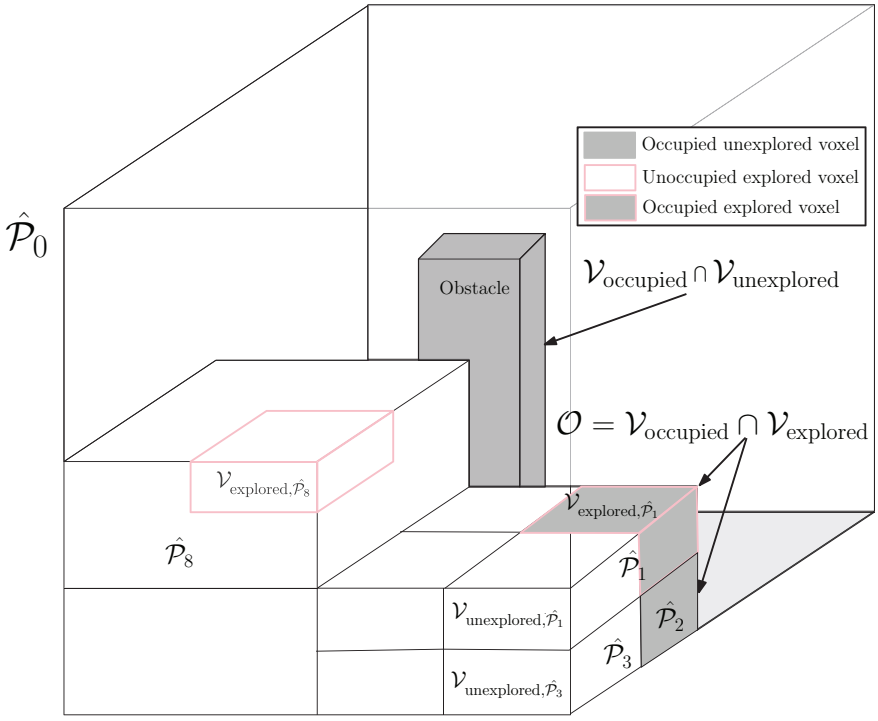

Fig. 3.1 A voxel map and its partitions $\hat{\mathcal{P}}_i$, $i = 1, \ldots 8$. Occupied unexplored voxels $\mathcal{V}_{\text{occupied}} \cap \mathcal{V}_{\text{unexplored}}$ are marked in gray, and occupied explored voxels $\mathcal{O}$ are marked in gray with a pink boundary. Free unexplored voxels fill the remaining space. The sets $\mathcal{V}_{\text{unexplored},\hat{\mathcal{P}}_1}$ and $\mathcal{V}_{\text{unexplored},\hat{\mathcal{P}}_3}$ are shown to illustrate the unexplored subsets of the parallelepipeds $\hat{\mathcal{P}}_1$ and $\hat{\mathcal{P}}_3$. The sets $\mathcal{V}_{\text{explored},\hat{\mathcal{P}}_1}$ and $\mathcal{V}_{\text{unexplored},\hat{\mathcal{P}}_8}$ illustrate explored and unexplored subsets of $\hat{\mathcal{P}}_1$ and $\hat{\mathcal{P}}_8$, respectively

where $\mu_2 \in [0, 1]$ is user-defined, $\hat{r}_{\mathcal{P}_{\text{prox}}} \in \mathcal{P}_{\text{prox}}$ denotes the barycenter of $\mathcal{P}_{\text{prox}}$, $\mathcal{P}_{\text{prox}} \in \left\{\widetilde{\mathcal{P}}_1, \ldots, \widetilde{\mathcal{P}}_{\tilde{n}_{\mathcal{P}}}\right\}$ denotes the partition that is closest to the UAV along its direction of motion, $\hat{r}_{\mathcal{P}_{\max}} \in \mathcal{P}_{\max}$ denotes the barycenter of $\mathcal{P}_{\max}$, and $\mathcal{P}_{\max} \in \left\{\widetilde{\mathcal{P}}_1, \ldots, \widetilde{\mathcal{P}}_{\tilde{n}_{\mathcal{P}}}\right\}$ denotes the partition containing the largest number of voxels. In this chapter, the condition whereby the barycenter of $\widetilde{\mathcal{P}}$ is accessible to the UAV is verified by applying a greedy A^* algorithm [19, pp. 604–608]. Partitions and goal points are recomputed as soon as the UAV's navigation system maps the voxel containing $\hat{r}_{\widetilde{\mathcal{P}}}$. If there does not exist a partition whose ratio of explored voxels is smaller than μ_1, then the user-defined set $\mathcal{V}$ is considered as mapped.

At each iteration of Algorithm 3.1, $\hat{\mathcal{P}}_0 \subseteq \mathcal{V}$ is partitioned in $n_{\mathcal{P}}$ rectangular parallelepipeds according to Algorithm 3.2. In Algorithm 3.2, if two sets of conditions are verified, then $\hat{\mathcal{P}}_0$ or any of its partitions are divided into smaller parallelepipeds, whose aspect ratio is the same as the aspect ratio of $\hat{\mathcal{P}}_0$. The first set of these conditions requires that both the ratio of explored voxels over the total number of voxels in $\hat{\mathcal{P}}_i$,

Algorithm 3.1 Generate goal points $\hat{r}_{\tilde{\mathcal{P}},q}$, $q = 0, \ldots, n_{\mathrm{g}}$, for the UAV

Set $q = 0$
while $\exists \tilde{\mathcal{P}}_i$ s.t. $\frac{n_{\mathrm{explored},\tilde{\mathcal{P}}_i}}{n_{\tilde{\mathcal{P}}_i}} < \mu_1$ **do**
  Partition $\hat{\mathcal{P}}_0$ in $\left\{\hat{\mathcal{P}}_1, \ldots, \hat{\mathcal{P}}_{n_\mathcal{P}}\right\}$ by executing Algorithm 3.2
  Find $\left\{\widetilde{\mathcal{P}}_1, \ldots, \widetilde{\mathcal{P}}_{\tilde{n}_\mathcal{P}}\right\} \subseteq \left\{\hat{\mathcal{P}}_1, \ldots, \hat{\mathcal{P}}_{n_\mathcal{P}}\right\}$ s.t. $\frac{n_{\mathrm{unexplored},\widetilde{\mathcal{P}}_i}}{n_{\widetilde{\mathcal{P}}_i}} \geq 1 - \mu_1, i \in \{1, \ldots, n_\mathcal{P}\}$
  Compute $\hat{r}_{\mathcal{P}_{\mathrm{prox}}}, \hat{r}_{\mathcal{P}_{\max}}$
  Determine the goal point as the barycenter of the parallelepiped containing (3.1)
  while Navigation system has not mapped $\hat{r}_{\tilde{\mathcal{P}},q}$ **do**
    wait
  end while
  Increment $q = q + 1$
end while

Algorithm 3.2 Octree iterative algorithm to partition $\hat{\mathcal{P}}_0$

Initialize $\hat{\mathcal{P}}_0 =$ minimum bounding box, $n_\mathcal{P} = 1$
Find $n_{\hat{\mathcal{P}}_0}, n_{\mathrm{explored},\hat{\mathcal{P}}_0}, n_{\mathrm{occupied},\hat{\mathcal{P}}_0}, \ell_{\hat{\mathcal{P}}_0,\min}$
if $\frac{n_{\mathrm{explored},\hat{\mathcal{P}}_0}}{n_{\hat{\mathcal{P}}_0}} < \mu_1$ **then**
  Divide$(\hat{\mathcal{P}}_0)$
end if
Divide $\hat{\mathcal{P}}_i$
Compute vertices of child parallelipipeds $\{\hat{\mathcal{P}}_{n_\mathcal{P}+1}, \ldots, \hat{\mathcal{P}}_{n_\mathcal{P}+8}\}$
Compute $n_{\hat{\mathcal{P}}_j}, n_{\mathrm{explored},\hat{\mathcal{P}}_j}, n_{\mathrm{occupied},\hat{\mathcal{P}}_j}, \ell_{\hat{\mathcal{P}}_j,\min}$, $j \in \{n_\mathcal{P} + 1, \ldots, n_\mathcal{P} + 8\}$
$n_\mathcal{P} = n_\mathcal{P} + 8$
for $j = (n_\mathcal{P} - 7)$:$n_\mathcal{P}$ **do**
  if $\frac{n_{\mathrm{explored},\hat{\mathcal{P}}_j}}{n_{\hat{\mathcal{P}}_j}} < \mu_1$ **and** $\frac{\ell_{\hat{\mathcal{P}}_j,\min}}{2} \geq \mu_3$ **then**
    if $n_{\mathrm{explored},\hat{\mathcal{P}}_j} \geq \mu_4$ **or** $\frac{n_{\mathrm{occupied},\hat{\mathcal{P}}_j}}{n_{\hat{\mathcal{P}}_j}} \geq \mu_5$ **then**
      Divide$(\hat{\mathcal{P}}_j)$
    end if
  end if
end for

$i = 0 \ldots, n_\mathcal{P}$, is smaller than the user-defined parameter $\mu_1 \in (0, 1)$ and the length of the smallest side of $\hat{\mathcal{P}}_i$ is larger than $2\mu_3$, where $\mu_3 > 0$ is user-defined. The second set of conditions requires that the number of explored voxels in $\hat{\mathcal{P}}_i$, $i = 0 \ldots, n_\mathcal{P}$, is larger than the user-defined parameter $\mu_4 \in \mathbb{N}$ or the ratio of occupied voxels over the total number of voxels in $\hat{\mathcal{P}}_i$ is larger than the user-defined parameter $\mu_5 \in (0, 1)$.

Algorithm 3.2 produces partitions of $\hat{\mathcal{P}}_0$ that are larger than some user-defined threshold and contain sufficiently many explored voxels, sufficiently many occupied voxels, or sufficiently many unexplored voxels. Setting larger values of μ_1 and smaller values of μ_3, μ_4, and μ_5 Algorithm 3.2 produces smaller partitions containing primarily unexplored voxels that are at the edge of explored areas, larger partitions containing primarily unexplored voxels that are located away from explored areas,

and larger partitions containing primarily explored voxels. Thus, it is expected that larger values of μ_1 and smaller values of μ_3, μ_4, and μ_5 imply longer flight times, since the UAV must collect more information about the environment. Setting larger values of $\mu_2 \in [0, 1]$, Algorithms 3.1 and 3.2 enable a systematic approach to cover $\hat{\mathcal{P}}_0$, whereas, by setting smaller values of $\mu_2 \in [0, 1]$, Algorithms 3.1 and 3.2 enable a greedy approach to the coverage problem. Therefore, it is expected that larger values of μ_1 and smaller values of $\mu_2, \ldots, \mu_5$ imply longer flight times since the UAV is tasked to cover smaller partitions of $\hat{\mathcal{P}}_0$ more greedily.

3.3.3.1 Study on the Computational Cost of Algorithm 3.2

In this section, we present the results of numerical simulations aimed at analyzing the performance of Algorithm 3.2 as a function of the user-defined parameters μ_1, μ_3, μ_4, and μ_5. These simulations have been performed on a computer hosting an i7-11800H 2.3 GHz CPU, 16 GB RAM, and operates Ubuntu 18.04. This single-board computer is used also for the software-in-the-loop simulations presented in Sect. 3.5 and future flight tests.

To perform these numerical simulations, 10 randomly generated obstacle sets $\mathcal{V}_{\text{occupied}}$ are employed, and, for each obstacle set, 10 sets of explored voxels $\mathcal{V}_{\text{explored}}$ are considered. Each obstacle set is generated by incrementally placing obstacles in the shape of cubes of 1 m side, starting from an empty map, whose volume is 2400 m^3, until 30% of the map is occupied. Given an obstacle set, each set of explored voxels is generated as the set of voxels contained in a sphere of 4 m radius and centered at one of 10 pre-determined locations, which are equally spaced in $\hat{\mathcal{P}}_0$. This approach allows to simulate maps that, similar to maps produced by UAVs at the beginning of a coverage mission, contain explored voxels that are clustered and occupy approximately 10% of $\hat{\mathcal{P}}_0$. A uniformly distributed random number generator is used to select the locations of the obstacles. A volume of 2400 m^3 was chosen because it is sufficiently representative of common industrial or office spaces, where the proposed guidance system for scouting unknown environments could be employed. Placing a cap on the density of obstacles is necessary since at higher densities, path planners similar to the one proposed in this chapter become unsuccessful on voxel maps, whose obstacles are scattered [2].

Since the first condition of Algorithm 3.2 to further divide a partition of $\hat{\mathcal{P}}_0$ concerns the pair (μ_1, μ_3) and its second condition concerns the pair (μ_4, μ_5), the proposed numerical study is performed by varying (μ_1, μ_3) and (μ_4, μ_5) separately. The user-defined parameter μ_1 is varied over the interval [0.30, 0.90], since, in the proposed simulation setup, if $\mu_1 < 0.30$, then the result of Algorithm 3.2, in general, is not refined enough to be useful. Furthermore, if $\mu_1 < 0.10$, then Algorithm 3.2 does not produce any partition. If μ_1 is less than or equal to 0.10 (since the initial sphere leave 10% of the map explored), and if $\mu_1 > 0.90$, then Algorithm 3.2 produces partitions containing a small number of voxels. The parameter μ_3 is spanned over the interval [0.20, 2.40], since the length of the smallest side of a voxel is 0.20 m and, considering the scopes of this chapter, at least two voxels should be contained

in a partition. Additionally, 2.40/2 m is sufficiently representative of the width of some space to be explored by the UAV. The parameter μ_4 is spanned over the set $\{4, \ldots, 160\}$, since the size of the UAV employed in this research is approximately equivalent to 4 voxels, and the field of view of the cameras employed in this research allows to explore a cone, whose diameter is approximately as large as 160 voxels. Finally, μ_5 is spanned over the interval [0.00, 0.30], since it is statistically impossible for a small UAV, which employs a guidance system similar to the proposed one, to find a viable path in a space occupied by 30% of randomly generated voxels. To reduce the computational effort involved with the proposed set of simulations, not all admissible pairs of (μ_1, μ_3) are tested, but only the pairs $(\mu_1, \mu_3) \in \mathcal{S}_1$, where $\mathcal{S}_1 \triangleq \{(\mu_1,\mu_3) = (0.35 + 0.05i, 0.20 + 0.20i), i = 0, \ldots, 11\}$ is an ordered set, since the computational time of Algorithm 3.2 increases for increasing values of μ_1 and decreasing values of μ_3, whereas it is unclear a priori whether the computational cost of Algorithm 3.2 is supposed to increase for increasing values of μ_1 and increasing values of μ_3 or not. Similarly, not all admissible pairs of (μ_4, μ_5) are tested, but only the pairs $(\mu_4, \mu_5) \in \mathcal{S}_2 \triangleq \{(\mu_4, \mu_5) = (160 - 12i, 0.01 + 0.022i), i = 0, \ldots, 13\}$, since the computational time of Algorithm 3.2 increases for decreasing values of μ_4 and decreasing values of μ_5, whereas it is unclear a priori whether the computational cost of Algorithm 3.2 is supposed to increase for decreasing values of μ_4 and increasing values of μ_5 or not.

The boxplots in Fig. 3.2 capture some of the statistical data of the 1000 simulations performed to test Algorithm 3.2 as functions of $(\mu_1, \mu_3) \in \mathcal{S}_1$, averaged over all $(\mu_4, \mu_5) \in \mathcal{S}_2$. From these plots, it appears that, varying the pair (μ_1, μ_3) over $\mathcal{S}_1$, the average computational time varies nonlinearly, captured by the best-fitting curve

$$p(\tau) = 0.0002\tau^7 - 0.0116\tau^6 + 0.2317\tau^5 - 2.1865\tau^4 + 10.0547\tau^3 - 22.3011\tau^2 + 36.8072\tau + 220.3555, \qquad \tau \in [1, 12],$$

where $p(1)$ approximates the computational time of Algorithm 3.2 for $(\mu_1, \mu_3) = (0.35, 0.20)$ and $p(12)$ approximates the computational time of Algorithm 3.2 for $(\mu_1, \mu_3) = (0.90, 2.40)$, the average interquartile range is 1 ms with maximum deviance of 1 ms, and the average length of the whiskers is 4 ms with maximum deviance of 2 ms. The boxplots in Fig. 3.3 capture some of the statistical data of the performed simulations as functions of $(\mu_4, \mu_5) \in \mathcal{S}_2$, averaged over all $(\mu_1, \mu_3) \in \mathcal{S}_1$. These plots show that varying the pair (μ_4, μ_5) over $\mathcal{S}_2$, the computational time remains substantially constant and approximately equal to 291 ms with maximum deviance of 3 ms, the interquartile range is in the order of 30 ms with maximum deviance of 1 ms, and the average length of the whiskers is 101 ms with maximum deviance of 3 ms.

From this analysis, we deduce that the computational time of Algorithm 3.2 increases for both increasing values of μ_1 and μ_3 and increasing values of μ_1 and decreasing values of μ_3. Therefore, μ_1 has a stronger impact than μ_3 on the computational time for Algorithm 3.2. From this analysis, we also deduce that the computational time of Algorithm 3.2 remains constant for decreasing values of μ_4

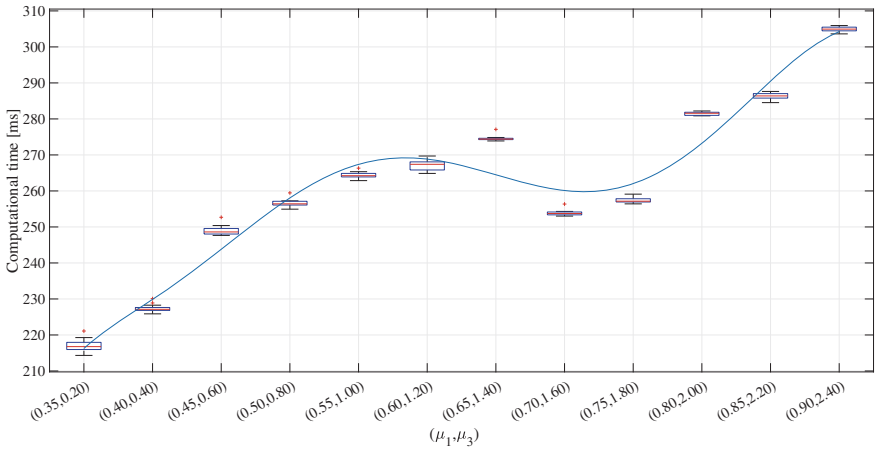

Fig. 3.2 Boxplots capturing statistical data on the computational time of Algorithm 3.2 as a function of $(\mu_1, \mu_3) \in \mathcal{S}_1$, averaged over $(\mu_4, \mu_5) \in \mathcal{S}_2$. As both μ_1 and μ_3 increase, the computational time increases of Algorithm 3.2 increases. A seventh-order polynomial over the interval [1, 12] best interpolates the average computational times

and increasing values of μ_5 and increases for decreasing values of μ_4 and decreasing values of μ_5. Therefore, to limit the computational cost of executing Algorithm 3.2, it is recommended to employ smaller values of μ_1, larger values of μ_3, smaller values of μ_4, and larger values of μ_5. However, in this case, $\hat{\mathcal{P}}_0$ is divided in larger partitions and all partitions contain a larger number of both explored and unexplored voxels. Alternatively, larger values of μ_1 and smaller values of μ_3, μ_4, and μ_5 imply higher computational costs, but produce partitions that contain either mostly explored voxels or mostly unexplored voxels.

In this work, the proposed guidance system is implemented on an Intel NUC single-board computer, which is sufficiently fast to execute an implementation of Algorithm 3.2 coded in C++ with low computational times, while producing partitions that contain either mostly explored voxels or mostly unexplored voxels. Thus, we set $(\mu_1, \mu_3, \mu_4, \mu_5) = (0.90, 1.00, 16, 0.28)$, and tested the performance of Algorithm 3.2 with in three sets of software-in-the-loop simulations on this single-board computer. These simulations were performed on the voxel map shown in the left-most plot of Fig. 3.4. This map captures an open space, high-bay area with few scattered obstacles on the north side, and low-ceiling offices on the south side. In the first simulation, we set $n_{\text{explored},\hat{\mathcal{P}}_0} = 63319$, in the second simulation, we set $n_{\text{explored},\hat{\mathcal{P}}_0} = 97697$, and in the third simulation, we set $n_{\text{explored},\hat{\mathcal{P}}_0} = 152394$. To collect statistically relevant data, each simulation was performed 10 times.

In the first, second, and third columns on the right of Fig. 3.4, the explored voxels are indicated by purple dots, the boundaries of the partitions produced by Algorithm 3.2 are denoted by red lines, and the unexplored set of voxels $\mathcal{V}_{\text{unexplored}}$ are unmarked. In average, the first set of simulation results was achieved in 213 ms with

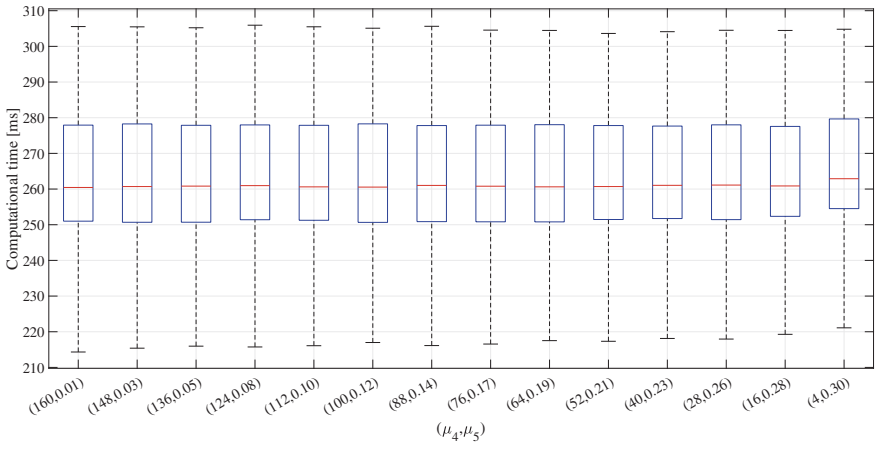

Fig. 3.3 Boxplots capturing statistical data on the computational time of Algorithm 3.2 as a function of $(\mu_4, \mu_5) \in \mathcal{S}_2$, averaged over $(\mu_1, \mu_3) \in \mathcal{S}_2$. Each set of results shows similar computational time, median, and variance

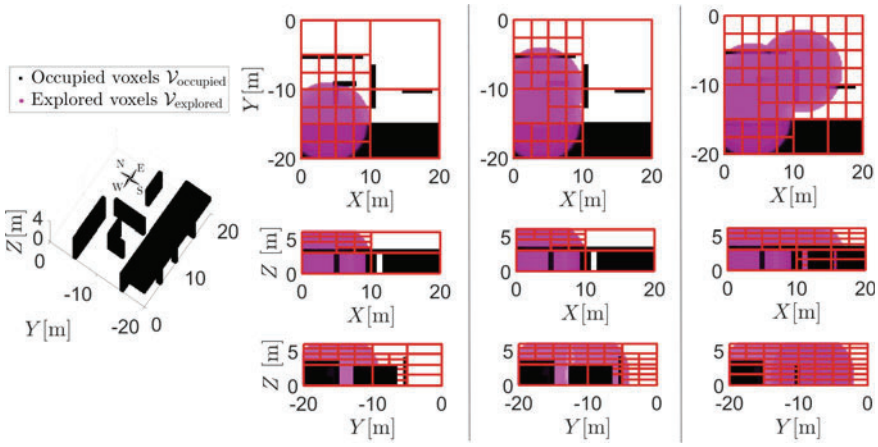

Fig. 3.4 Voxel map employed to evaluate the performance of Algorithm 3.2 and graphical representation of three sets of simulations. The occupied voxels are denoted by black dots, the explored voxels are indicated by purple dots, the boundaries of the partitions produced by Algorithm 3.2 are denoted by red lines, and the unexplored set of voxels $\mathcal{V}_{\text{unexplored}}$ are unmarked. By setting $(\mu_1, \mu_3, \mu_4, \mu_5) = (0.90, 1.00, 16, 0.28)$, Algorithm 3.2 produces partitions that cluster explored voxels into large parallelepipeds, partitions that cluster unexplored voxels into large parallelepipeds, and smaller partitions on the border of the explored set

a standard deviation of 3 ms, the second set of simulation results was achieved in 372 ms with a standard deviation of 9 ms, and the third set of simulation results was achieved in 617 ms with a standard deviation of 11 ms; these results show that Algorithm 3.2 can be executed in real time on a multi-rotor UAV. These simulations also show that, regardless of the size $\mathcal{V}_{\text{explored}}$, Algorithm 3.2 produces partitions that cluster explored voxels into large parallelepipeds, partitions that cluster unexplored voxels into large parallelepipeds, and smaller partitions on the border of the explored set. This is an ideal result, since it enables a greedy behavior by setting μ_2 in (3.1) closer to 0 and, hence, allowing to select goal points for the UAV in areas that are large and mostly unexplored, while ignoring large and mostly explored areas, or a systematic behavior by setting μ_2 closer to 1.

3.3.4 Numerical Solution of the Path Planning Problem

Given the sequence of goal points $\{\hat{r}_{\widetilde{\mathcal{P}},q}\}_{q=0}^{n_{\text{g}}}$, the proposed path planning algorithm generates the *UAV's reference path* as the sequence $\{\hat{r}_k\}_{k=0}^{n_{\text{w}}} \subset \mathbb{R}^3 \setminus O$ such that $\hat{r}_0 = \hat{r}_{\widetilde{\mathcal{P}},q}, q \in \{0, \ldots, n_{\text{g}} - 1\}$, and $\hat{r}_{n_{\text{w}}} = \hat{r}_{\widetilde{\mathcal{P}},q+1}$. The sequence $\{\hat{r}_k\}_{k=0}^{n_{\text{w}}}$ minimizes the cost function

$$f_{k,q} \triangleq g_k + h_{k,q}, \qquad k \in \{1, \ldots, n_{\text{w}}\}, \qquad q \in \{0, \ldots, n_{\text{g}} - 1\}, \tag{3.2}$$

where

$$g_k \triangleq \sum_{p=1}^{k} \left[\kappa(\mathrm{d}_2(\hat{r}_p, O))\mathrm{d}_2(\hat{r}_p, \hat{r}_{p-1})\right] \tag{3.3}$$

denotes the *cost-to-come function*,

$$h_{k,q} \triangleq \mu_8 \mathrm{d}_2(\hat{r}_k, \hat{r}_{\widetilde{\mathcal{P}},q+1}) \tag{3.4}$$

denotes the *heuristic function*,

$$\kappa(\alpha) \triangleq \begin{cases} \mu_8 + 0.5(1 - \mu_8)\left[1 + \cos\frac{2\pi(\alpha - \mu_1)}{\mu_7 - \mu_6}\right], & \alpha \in [\mu_6, \mu_7], \\ 1 & \alpha \in [0, \mu_6) \cup (\mu_7, \infty), \end{cases} \tag{3.5}$$

denotes the *weighing function*, and $\mu_7 > \mu_6 > 0$ and $\mu_8 \in [0, 1)$ are user-defined parameters. The cost function (3.2) is the weighted sum of the length of the UAV's reference path, namely, (3.3), and an underestimate of the Euclidean distance between the voxel occupied by the UAV and the goal point $\hat{r}_{\widetilde{\mathcal{P}},q}, q \in \{1, \ldots, n_{\text{g}}\}$, namely, (3.4).

The weighing function $\kappa(\cdot)$ is continuous, its minimum is equal to μ_8 and is attained at $\alpha = (\mu_6 + \mu_7)/2$, and it encourages tactical behaviors by rewarding paths

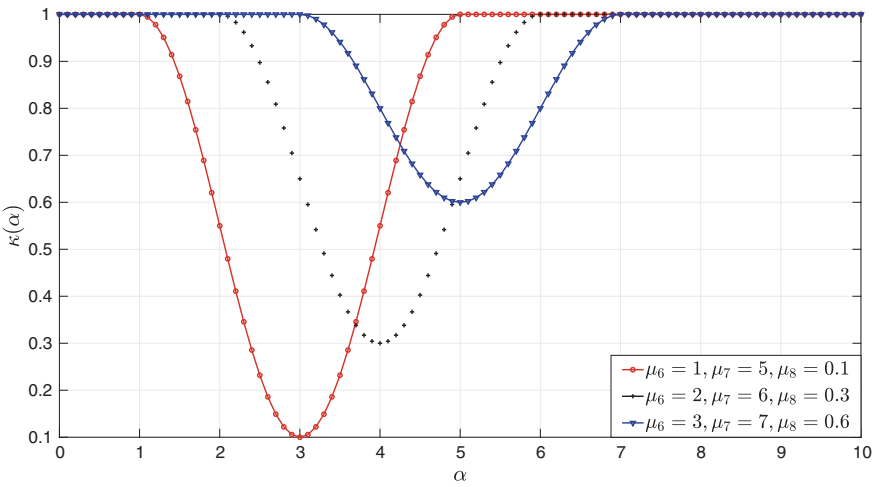

Fig. 3.5 Plot of the weighing function $\kappa(\cdot)$ as a function of $\alpha > 0$ with $(\mu_6, \mu_7, \mu_8) = (1, 5, 0.1)$, $(\mu_6, \mu_7, \mu_8) = (2, 6, 0.3)$, and $(\mu_6, \mu_7, \mu_8) = (1, 5, 0.1)$. This weighing function is used to encourage the proposed guidance system to search for paths close to the obstacles set, so that the UAV may exploit obstacles for shelter

that are closer to the obstacles' set $\mathcal{O}$. Thus, it follows from (3.3) that for larger values of μ_8, the obstacles' attractive effect is less enhanced, and the UAV exhibits more reckless behavior. For all $\alpha \in [\mu_6, \mu_7]$, it holds that $\kappa(\alpha) < 1$ and hence, $[\mu_6, \mu_7]$ captures the region of influence of the obstacles set. If the UAV occupies a voxel that is either closer than μ_6 to $\mathcal{O}$ or further from $\mathcal{O}$ than μ_7, then the weighing function is equal to unity, and the cost-to-come function reduces to the cost-to-come function employed in classical heuristics-based path planning algorithms. Figure 3.5 shows a graphical representation of $\kappa(\cdot)$ for multiple values of μ_6, μ_7, and μ_8.

Assuming that the UAV is able to move to any adjacent unoccupied voxel, the reference path $\{\hat{r}_k\}_{k=0}^{n_w}$ that minimizes (3.2) can be computed by applying any optimization algorithm based on the use of a heuristic function over a graph, for instance, the A^* [19, Appendix C] or the LPA^* algorithm [5]. Since μ_8 scales the distance between the UAV's position and the next goal point $\hat{r}_{\mathcal{P},q}$, $q \in \{1, \ldots, n_g\}$, in (3.4), and $\kappa(\alpha) \geq \mu_8$ for all $\alpha \geq 0$, it follows from the triangle inequality that the heuristic function $h_{k,q}$ is consistent and hence, the proposed path planning subsystem does not search voxels that were already visited in previous iterations of the heuristics-based search algorithm, and the reference path $\{\hat{r}_k\}_{k=0}^{n_w}$ that minimizes (3.2) will be the terminating path.

An alternative weighing function for tactical path planning has been proposed in [2]. A key difference between (3.5) and the weighing function in [2] lies in the fact that the user is now able to set explicitly the width of the region of influence of the obstacles' set. Detailed discussions on the role of weighing functions, such as (3.5),

in designing path planner for tactical vehicles, while employing heuristics-based search algorithms, are presented in [2, 20].

3.3.5 Numerical Solver Selection

In this section, we discuss the selection process for the optimization algorithm underlying the proposed path planner. To this goal, two algorithms were considered, namely, the A^* algorithm [19, Appendix C] and the LPA^* algorithm [5]. An advantage of LPA^* over A^* lies in the fact that, if part of the search graph varies over time, then LPA^* leverages those parts of the graph that remain unchanged, whereas A^* must be re-initiated each time the search graph varies. Thus, LPA^* is computationally more efficient than A^*. This feature of LPA^* is highly relevant for applications such as those considered in this chapter, wherein the obstacles' set is updated as the UAV proceeds in its coverage task, and the path planning process must be reiterated as the voxel map evolves.

To verify the computational advantage of LPA^* over A^*, these algorithms have been tested by means of four sets of 650 software-in-the-loop simulations performed on an Intel *i*7 processor with 8 cores, whose maximum frequency is 4.30 GHz, and which executes a Linux operating system. Two sets of simulations are aimed at testing the computational efficiency of LPA^* over A^*, while seeking tactical paths, and two sets of simulations are aimed at testing the computational efficiency of LPA^* over A^*, while seeking reckless paths. In particular, tactical paths have been tested by seeking paths that minimize the cost function (3.2) with $\mu_6 = 4$, $\mu_7 = 8$, and $\mu_8 = 0.45$, and reckless paths have been tested by seeking paths that minimize the cost function (3.2) with $\mu_6 = 3$, $\mu_7 = 7$, and $\mu_8 = 0.75$. In all simulations, the UAV reference path connects a given initial position to a given final position, and must traverse the voxel map of some office space captured by employing the camera-based navigation system presented in [2]; this voxel map is shown in Figure 3.4. In order to challenge the optimization algorithm's speed in dynamic environments, an incremental number of obstacles is introduced along the planned paths over the course of each simulation. These obstacles are placed so that their distance is not smaller than one UAV length and, hence, considering the size of the voxel map and the size of the UAV, a maximum of 12 obstacles have been introduced. Fixed the number of obstacles to be introduced in the voxel map along the UAV's path, 50 simulations have been performed to assess the computational speed of each optimization algorithm.

The results of the simulations obtained by employing the A^* algorithm and the LPA^* algorithm with a tactical parameter set are shown in Figs. 3.6 and 3.7, respectively, and the results of the simulations obtained by employing the A^* algorithm and the LPA^* algorithm with a tactical parameter set are shown in Figs. 3.8 and 3.9, respectively. It appears from Figs. 3.6 and 3.8 that the average computational time for the A^* algorithm remains substantially constant with the number of obstacles introduced into the voxel map. Additionally, it appears from Figs. 3.6, 3.7, 3.8 and 3.9 that, if no obstacle or one obstacle is introduced along the UAV's path, then

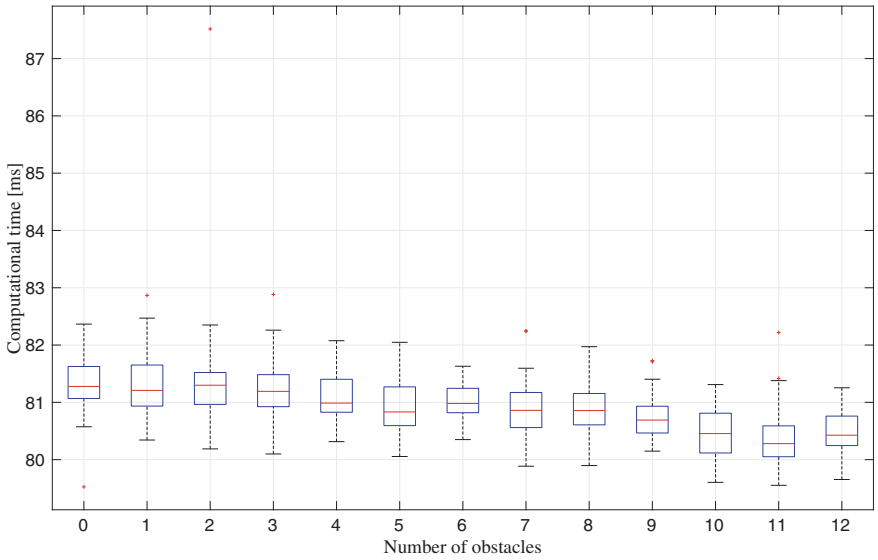

Fig. 3.6 Statistical data on the computational time of the A^* optimization algorithm as a function of the number of obstacles introduced along the UAV's path, while seeking tactical paths. The red line indicates the average computational time over the 50 simulations, fixed the number of obstacles, the blue box indicates the 25–75% range, the black lines indicate the non-outlier minimum and maximum, and the red dots indicate outliers. The average computational time required to produce a path is approximately constant with the number of obstacles and is 80.93 ms with average standard deviation of 0.34 ms

the average computational time for the LPA^* algorithm is higher than the average computational time for the A^* algorithm. However, for more than one obstacle, the average computational time of LPA^* is lower than the average computational time of A^*. In light of these results, the LPA^* algorithm has been employed for the proposed path planner. Finally, the average computational time of the LPA^* algorithm appears to increase with the number of obstacles being introduced. However, introducing obstacles at a distance smaller than one UAV length is not relevant to the scopes of this research.

3.3.6 Safety Measures

The objective of Algorithm 3.1 is to deduce goals $\hat{r}_{\tilde{\mathcal{P}},q}, q = 0, \ldots, n_g$, which lead the UAV's navigation system to under-explored areas of the voxel map $\mathcal{V}$. In some cases, $\hat{r}_{\tilde{\mathcal{P}},q}$ lies in an area that the navigation system can not observe from the UAV's current position $r_k(i\Delta T)$, irrespectively of the yaw angle $\psi_k(i\Delta T)$ and pitch angle $\theta_k(i\Delta T)$, due to the bounds on the cameras' field-of-view captured by $\psi_{\max}$ and the maximum pitch angle $\theta_{\max}$. As a result, the planned path may attempt to traverse unexplored,

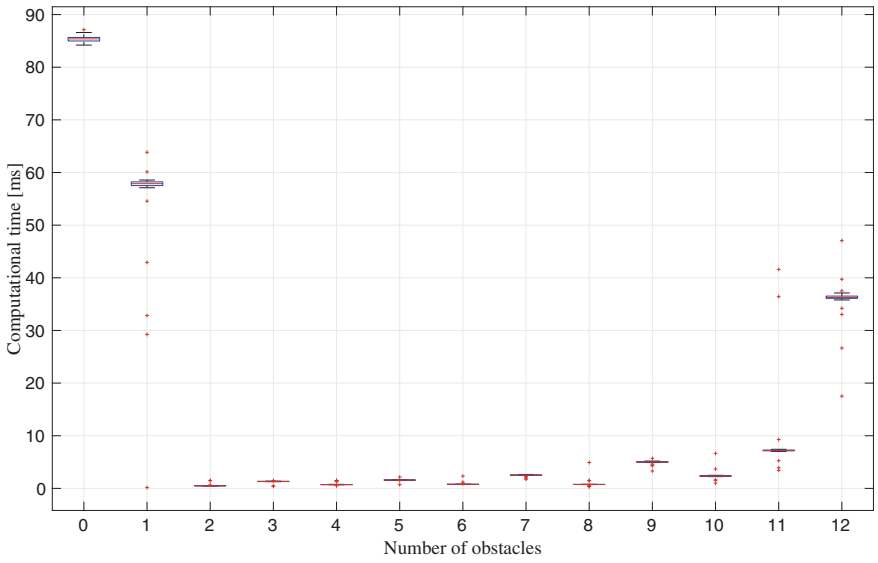

Fig. 3.7 Statistical data on the computational time of the the LPA^* optimization algorithm as a function of the number of obstacles introduced along the UAV's path while seeking tactical paths. The average computational time required to produce a path is approximately constant with the number of obstacles and is 15.45 ms with average standard deviation of 26.86 ms. If no obstacle or only one obstacle is introduced along the UAV's path, then, on average, then LPA^* requires more time than A^* to recalculate the vehicle's path. If more than one obstacle is introduced along the UAV's path, then, on average, the computational time is less than the computational time required by A^*

unobservable voxels that are occupied and, hence, jeopardizes the UAV's safety; see Fig. 3.10 for a schematic representation of this case.

In this section, we present a novel approach to address the challenge of planning a path while accounting for unobservable occupied voxels. This approach, which is captured by Algorithm 3.3 and is schematically represented by Fig. 3.11, consists in penalizing the path that leads the vehicle toward unobservable voxels. Given $\hat{r}_k \in \mathcal{V} \setminus \mathcal{O}$, the navigation system's field-of-view $\psi_{\max}$, and the maximum pitch angle $\theta_{\max}$, the set of unobservable voxels is approximated by the closed cone $\overline{\mathcal{K}}_{h,\theta_{\text{blind}}}\left(\hat{r}_k\right) \subset \mathbb{R}^3$, whose apex is at $\hat{r}_k$, whose axis is given by the UAV's yaw axis, and whose semi-aperture is $\theta_{\text{blind}} \triangleq \frac{\pi - \psi_{\max}}{2} - \theta_{\max}$, and whose height is $h > 0$. Given a goal point $\hat{r}_{\tilde{\mathcal{P}},q}$, $q = 0, \ldots, n_{\text{g}}$, firstly, we determine if $\hat{r}_{\tilde{\mathcal{P}},q} \in \overline{\mathcal{K}}_{h,\theta_{\text{blind}}}\left(\hat{r}_k\right)$. If so, then we define the closed right circular cylinder $\mathcal{C}_{\text{danger}} \triangleq \overline{C}_{r_{\text{danger}},h_{\text{danger}}} \subset \mathcal{V} \setminus \mathcal{O}$ centered in the UAV, whose axis is given by the UAV's yaw axis, and whose semi-height is given by $h_{\text{danger}} > 0$. Thus, we perform ray tracing across the voxels contained in the circle $\mathcal{C}_{\text{origins}} \triangleq \overline{C}_{r_{\text{danger}},0} \subset \mathcal{V} \setminus \mathcal{O}$ contained in the plane orthogonal to the UAV's yaw axis and containing the center of the UAV, where $r_{\text{danger}} > 0$ denotes the circle's radius, and along the direction $\frac{\hat{r}_k - \hat{r}_{\tilde{\mathcal{P}},q}}{\|\hat{r}_k - \hat{r}_{\tilde{\mathcal{P}},q}\|}$. If any ray intersects $\mathcal{V}_{\text{unexplored}}$, then the Euclidean

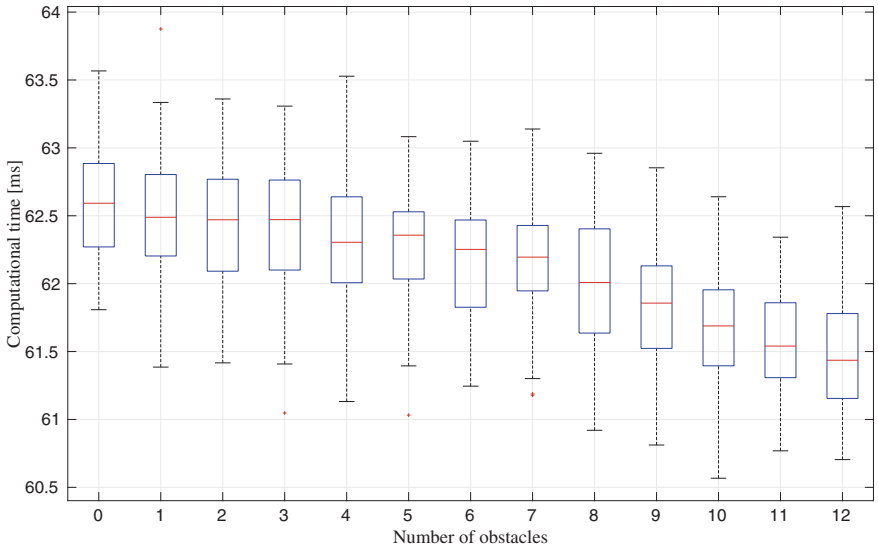

Fig. 3.8 Statistical data on the computational time of the A^* optimization algorithm as a function of the number of obstacles introduced along the UAV's path while seeking reckless paths. The average computational time required to produce a path is approximately constant with the number of obstacles and is 62.11 ms with average standard deviation of 0.37 ms

distance between any two points in $\hat{r}_p, \hat{r}_q \in C_{\text{danger}}$ when evaluating (3.2) is replaced with

$$\mathrm{d}_2(\hat{r}_p, \hat{r}_q, M) = \sqrt{(\hat{r}_p - \hat{r}_q)^{\mathrm{T}} \mathrm{diag}(1, 1, M)(\hat{r}_p - \hat{r}_q)}, \tag{3.6}$$

where $M \geq 1$ denotes a user-defined bias set arbitrarily large to discourage any vertical motion within C_{danger}. To penalize positive changes in altitude and not penalize negative ones, M in (3.6) can be set as $M = \max\left\{\max\left\{\text{sign}(\hat{r}_{p,3} - \hat{r}_{q,3}), 1\right\}, \text{sign}(\hat{r}_{p,3} - \hat{r}_{q,3})M_1\right\}$ where $M_1 > 1$ is arbitrarily large. Alternatively, to penalize negative changes in altitude and not penalize positive ones, M in (3.6) can be set as $M = \max\left\{\max\left\{-\text{sign}(\hat{r}_{p,3} - \hat{r}_{q,3}), 1\right\}, -\text{sign}(\hat{r}_{p,3} - \hat{r}_{q,3})M_1\right\}$. If a ray intersects O, then ray tracing along that particular ray is terminated.

Next, the path planning problem is solved by employing the solver outlined in Sect. 3.3.4. Once the search algorithm leaves C_{danger}, the process above is repeated until a new goal $\hat{r}_{\tilde{\mathcal{P}},q+1}$ is generated; in this case, the original distance function $\mathrm{d}_2(\cdot, \cdot, M)$ is restored, that is, M is set to one.

To validate Algorithm 3.3, the proposed path planning system is executed in three simulations involving the same environment, namely, a two-story building with an access door per floor; see Fig. 3.12. In all simulations, we consider the same initial position, namely, $r_k(i\Delta T) = [17.0, 17.0, 1.4]^{\mathrm{T}}$, which is on the first floor of the simulated map, and the same goal point, namely, $\hat{r}_{\tilde{P},q} = [17.0, 17.0, 4.6]^{\mathrm{T}}$, which is on

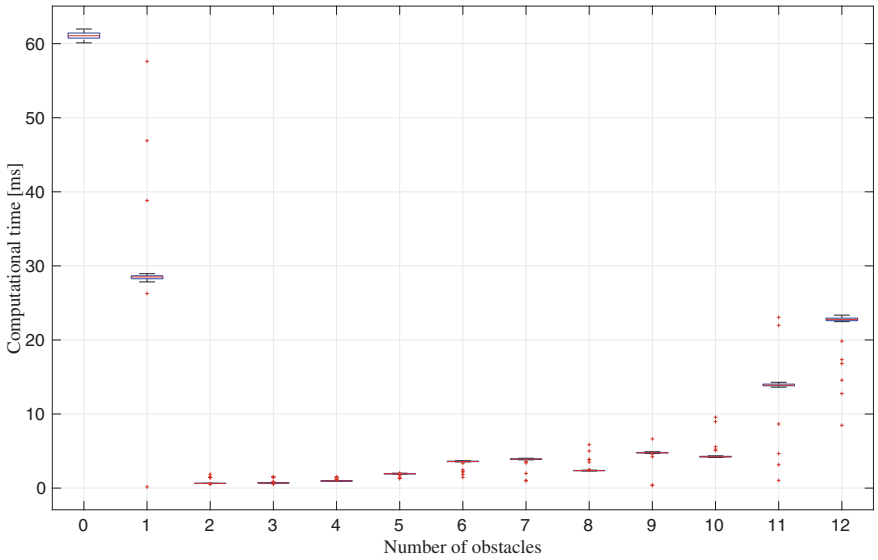

Fig. 3.9 Statistical data on the computational time of the the LPA^* optimization algorithm as a function of the number of obstacles introduced along the UAV's path while seeking tactical paths. The average computational time required to produce a path is approximately constant with the number of obstacles and is 11.45 ms with average standard deviation of 17.34 ms. Similar to the search for tactical paths, if no obstacle or only one obstacle is introduced along the UAV's path, then, on average, then LPA^* requires more time than A^* to recalculate the vehicle's path. If more than one obstacle is introduced along the UAV's path, then, on average, the computational time is less than the computational time required by A^*

Algorithm 3.3 Evaluate safety of the goal point $\hat{r}_{\tilde{\mathcal{P}},q}$ and bias the cost function (3.2)

1: **if** $\frac{\hat{r}_k - \hat{r}_{\tilde{\mathcal{P}},q}}{\|\hat{r}_k - \hat{r}_{\tilde{\mathcal{P}},q}\|} \cdot (\hat{r}_{k,3} - \hat{r}_{\tilde{\mathcal{P}},q,3})\mathbf{e}_{3,3} \geq \cos\theta_{\text{blind}}$ **then**
2: **Compute** $\mathcal{C}_{\text{origins}}$
3: **RayTracing**($\mathcal{C}_{\text{origins}}$)
4: **if** Ray tracing encountered $\mathcal{V}_{\text{unexplored}}$ **then**
5: $\forall(\hat{r}_p, \hat{r}_q) \in \mathcal{C}_{\text{danger}} \times \mathcal{C}_{\text{danger}}$, $p \neq q$, $\hat{r}_{p,3} \neq \hat{r}_{q,3}$ use (3.6) to compute (3.3).
6: **end if**
7: **end if**

the second floor. Thus, we observe the outcome of the proposed path planner with and without Algorithm 3.3, while assuming that the voxels of the floor between the start and goal points are occupied and unexplored; see the left image in Fig. 3.12. Thus, we observe the outcome of the proposed path planner without Algorithm 3.3 assuming that the voxels of the floor between the start and goal points are occupied and explored; see the right image in Fig. 3.12. It is apparent how, while employing Algorithm 3.3, the proposed path planner avoids the unexplored, occupied, and unobservable, and hence, unsafe voxels between the start and goal point. Furthermore, employing Algorithm 3.3, the proposed planner produces a path similar to the

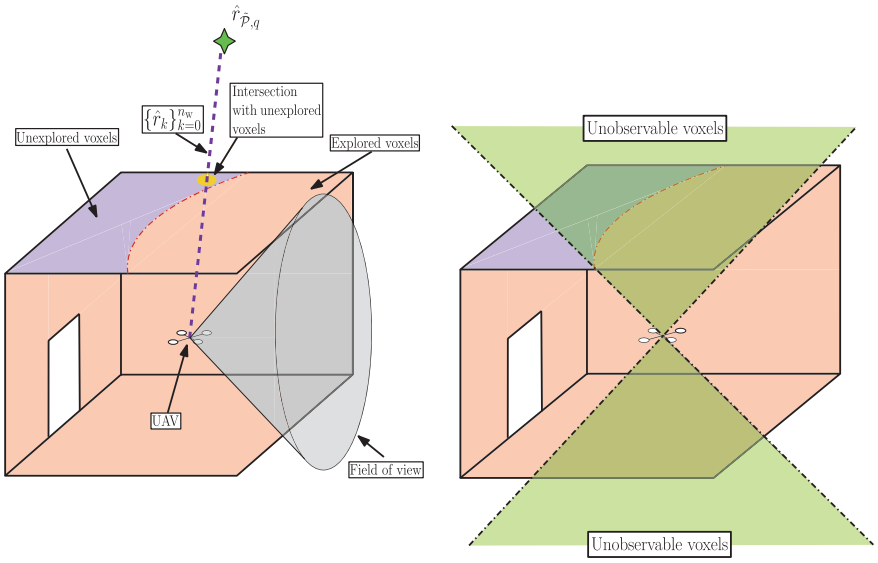

Fig. 3.10 Schematic representation of a challenging scenario. In the left image, the UAV enters a room through a doorway, and, there is a large cluster of unexplored and occupied voxels (marked in blue) in the ceiling of the room. If the goal point $\hat{r}_{\hat{P},q}$ generated by Algorithm 3.1 is very close to the UAV's vertical axis, then the planned path $\{\hat{r}_k\}_{k=0}^{n_w}$ may intersect the cluster of unexplored voxels in the ceiling, since, in this work, unexplored voxels are considered unoccupied. Because of the cameras' limited field of view, the UAV's constraints on the attitude, which are imposed by the trajectory planner (see Sect. 3.4 below), and the geometry of the room, there may be a set of voxels that is impossible to observe for any yaw and pitch angles (marked in green in the right image). If this set of unobservable voxels intersects the set of unexplored voxels, then this scenario may jeopardize the UAV's safety

path obtained assuming that the entire voxel map is explored. Conversely, without Algorithm 3.3 and assuming that the voxels between the start and goal points are unexplored and unobservable, the UAV attempts to traverse the floor between the first and second floors.

3.4 Trajectory Planning System for Tactical Coverage

3.4.1 Overview

In order to exploit the UAV's nonlinear dynamics while traversing the waypoints outlined by the path planner, the proposed guidance system employs a trajectory planner. In particular, reference trajectories for the UAV's position and yaw angle

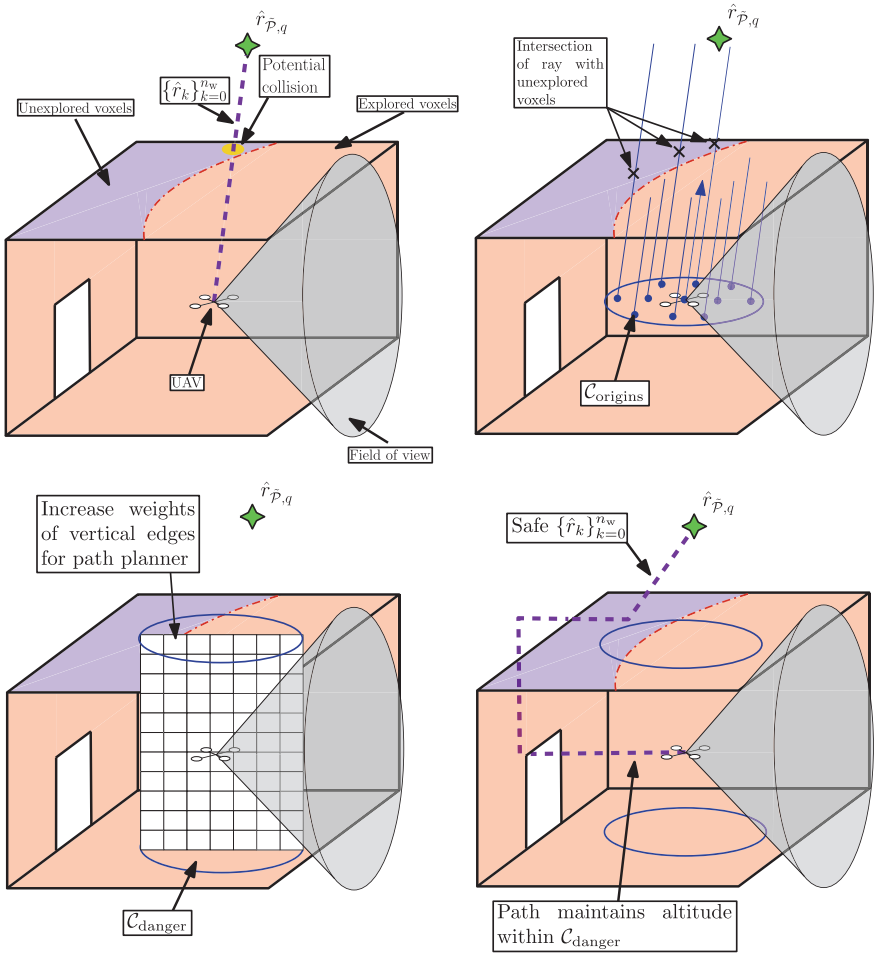

Fig. 3.11 Graphical representation of Algorithm 3.3 to assess the safety of the goal point $\hat{r}_{\tilde{P},q}$ and avoid traversing unobservable, occupied voxels. The first step is illustrated in the top left image, in which we deduce if $\hat{r}_{\tilde{P},q}$ lies in the set of unobservable voxels. Next, in the top right image, the second step requires performing several ray tracings in the voxel map to determine if there are unexplored voxels in the direction of $\hat{r}_{\tilde{P},q}$ relative to the UAV's position $r_k(i\Delta T)$. If any ray intersects an unexplored voxel, then there is a potential for a collision with an unobservable occupied voxel. Next, if any ray intersects $\mathcal{V}_{\text{unexplored}}$, then the Euclidean distance of traversing from $\hat{r}_p \in \mathcal{C}_{\text{danger}} \setminus \mathcal{O}$ to $\hat{r}_q \in \mathcal{C}_{\text{danger}} \setminus \mathcal{O}$, $p \neq q$, is modified according to (3.6); see bottom left image. Finally, Algorithm 3.3 is repeated until $r_k(i\Delta T) \notin \mathcal{C}_{\text{danger}}$; see bottom right image. The changes to the transition costs persist until the goal is updated

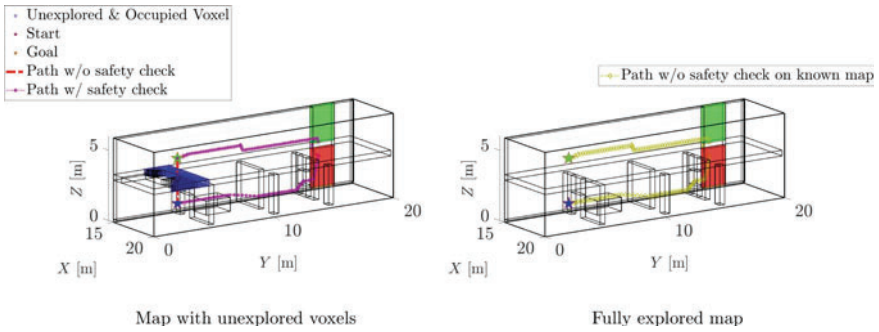

Fig. 3.12 Results of three simulations involving the proposed path planner. In all simulations, the UAV is tasked with reaching a point directly above its initial position. However, a set of occupied voxels lies between the start and goal points. The simulation results in the left image show the UAV's path employing Algorithm 3.3 and not employing this algorithm, assuming that the voxels between the start and goal points are unobservable. In the first case, the UAV avoids the unobservable occupied voxels and safely reaches the goal point. In the second point, the UAV attempts to traverse the unobservable occupied voxels. The simulation results in the right image show the UAV's path employing assuming that all voxels are explored. In this case, the UAV's path is similar to the path obtained assuming that the voxels between the start and goal points are unobservable and employing Algorithm 3.3

are computed as solutions of a linear–quadratic optimal control problem applied to the feedback-linearized equations of motion of the UAV.

The optimal control problem's time horizon is user-defined, and the cost function captures the UAV's competing needs of reaching the next waypoints and coasting the obstacles' set to seek shelter from potential threats. The underlying equality constraints capture both the UAV's output-feedback linearized, discretized equations of motion and the need to interpolate the sequence of waypoints given by the UAV's reference path $\left\{\hat{r}_k\right\}_{k=0}^{n_w}$ between the goal point $\hat{r}_{\tilde{\mathcal{P}},q}$, $q \in \{0, \ldots, n_g - 1\}$, and $\hat{r}_{\tilde{\mathcal{P}},q+1}$. The inequality constraints capture the UAV's need of avoiding obstacles, maintaining the next goal point in the cameras' fields of view at all times, verifying the bounds on the Euler angles, and assuring that each propeller's thrust force is nonnegative. Lastly, the boundary conditions to the optimal trajectory planning problem capture the need to reach the next waypoint and orient the onboard cameras' focal axes toward the next waypoint.

The proposed trajectory planning system has several distinctive features. The linear–quadratic optimal control problem underlying the proposed trajectory planner is solved numerically as a quadratic optimization problem by employing the model predictive control methodology. The feedback-linearized equations of motion allow to exploit the vehicle's nonlinear dynamics and map areas above and below the UAV by rotating the cameras, rather than varying the UAV's altitude. Collision avoidance constraints are captured by convex sets with affine boundaries, which are identified by an original algorithm that allows to plan reference trajectories over sufficiently long time horizons. Finally, applying the approach presented in [2], *fast* solutions

to the trajectory planning problem are obtained by employing logarithmic barrier functions to embed inequality constraints in the cost function and exploiting the block-tridiagonal structure of the matrices that capture the quadratic cost function and the constraint equations.

A key difference between the trajectory planning algorithm presented in this chapter and the trajectory planning algorithm presented in [2] lays in the fact that, at each time step, the proposed trajectory planner accounts for a user-defined number of waypoints ahead of the UAV, whereas the trajectory planner discussed in [2] only accounts for the next waypoint. This approach allows the proposed guidance system to trajectories that account for the UAV's tactical needs over larger distances. Additionally, the proposed trajectory planner allows to choose how closely these waypoints need to be followed, whereas the trajectory planner in [2] requires the UAV to traverse all waypoints. As shown in [20], accounting for one waypoint at the time and requiring to traverse all waypoints, it is particularly difficult to plan trajectories that are both tactical and sufficiently short. Being able to coarsely follow waypoints allows to set the path planner the UAV is drawn to the next goal point through the shortest path, and the trajectory planner so that a more tactical behavior is enforced by coasting the obstacles' set more closely.

Compared to the collision avoidance algorithm in [2], the proposed collision avoidance algorithm allows to compute larger convex sets, where the reference trajectory can be computed and hence, allows to investigate a larger search space. Finally, in [2], the MPC algorithm is applied irrespectively of whether the reference trajectory traverses explored or unexplored voxels, whereas, in this work, the UAV's reference trajectory is not recomputed at each time step over those segments that traverse explored voxels; this features allows to further reduce the trajectory planner's computational time.

3.4.2 Notation

Time is denoted by $t \geq 0$, and we assume that the UAV is able to fly from $\hat{r}_k$, $k \in \{0, \ldots, n_\mathrm{w} - 1\}$, 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 $k \in \{0, \ldots, n_\mathrm{w} - 1\}$ is omitted for simplicity of exposition.

The UAV's *mass* is denoted by $m > 0$ and the *gravitational acceleration* is denoted by $g > 0$. Assuming that the UAV's roll, pitch, and yaw axes are principal axes of inertia, the UAV's *matrix of inertia* is denoted by $I \in \mathbb{R}^{3\times 3}$, which is diagonal and positive-definite. The *UAV's position* is captured by $r_k : [0, n_\mathrm{t} \Delta T] \to \mathbb{R}^3 \setminus O$, $k \in \{0, \ldots, n_\mathrm{w} - 1\}$, expressed in a conveniently located inertial reference frame $\mathbb{I}$. The *UAV's roll angle* is denoted by $\phi_k : [0, n_\mathrm{t} \Delta T] \to \left(-\frac{\pi}{2}, \frac{\pi}{2}\right), k \in \{0, \ldots, n_\mathrm{w} - 1\}$, 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 $v_k : [0, n_t \Delta T] \to \mathbb{R}^3$, the UAV's *angular velocity* with respect to $\mathbb{I}$ is denoted by $\omega_k : [0, n_t \Delta T] \to \mathbb{R}^3$, and the the UAV's *state vector* is denoted by $x_k(j\Delta T) \triangleq \left[r_k^T(j\Delta T), \phi(j\Delta T), \theta(j\Delta T), \psi(j\Delta T), v_k^T(j\Delta T), \omega_k^T(j\Delta T)\right]^T$, $j \in \{i, \ldots, n_t\}$, $i \in \{0, \ldots, n_t\}$, $k \in \{0, \ldots, n_w - 1\}$.

Let $\eta_{1,k} : [0, n_t \Delta T] \to \mathbb{R}$, $k \in \{0, \ldots, n_w - 1\}$, denote the *total thrust force's virtual control input*, the *total thrust force* produced by the UAV's propellers is defined as $u_{1,k}(\cdot)$ such that $u_{1,k}(t) = [1, 0]\, \delta_k(t)$, $t \in [0, n_t \Delta T]$, and

$$\dot{\delta}_k(t) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \delta_k(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \eta_{1,k}(t), \quad \begin{bmatrix} u_{1,k}(0) \\ u_{1,k}(n_t \Delta T) \end{bmatrix} = \begin{bmatrix} u_{1,0,k} \\ u_{1,f,k} \end{bmatrix}, \quad t \in [0, n_t \Delta T]. \tag{3.7}$$

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 control input* is defined as $u_k(t) \triangleq \left[u_{1,k}(t), u_{2,k}(t), u_{3,k}(t), u_{4,k}(t)\right]^T$, $t \in [0, n_t \Delta T]$, $k \in \{0, \ldots, n_w - 1\}$, and the *vector of thrust forces* produced by each propeller is defined as

$$T_k(t) \triangleq M_{T,u} u_k(t), \qquad t \in [0, n_t \Delta T], \tag{3.8}$$

where the ith component of $T_k(\cdot)$, $i = 1, \ldots, 4$, namely, $T_{i,k} : [0, n_t \Delta T] \to [0, \infty)$, denotes the thrust force produced by the ith propeller, $M_{T,u} \triangleq \frac{1}{4} \begin{bmatrix} 1 & 0 & 2l^{-1} & -c_T^{-1} \\ 1 & -2l^{-1} & 0 & c_T^{-1} \\ 1 & 0 & -2l^{-1} & -c_T^{-1} \\ 1 & 2l^{-1} & 0 & c_T^{-1} \end{bmatrix}$, $l > 0$ denotes the *distance of the propellers from the vehicle's barycenter*, and $c_T > 0$ denotes the *propellers' drag coefficient* [21].

3.4.3 Output-Feedback Linearized Equations of Motion

Neglecting the aerodynamic drag, the inertial counter-torque, and the gyroscopic effect [21], which are small for Class I aircraft such as those considered in this chapter, the UAV's continuous-time equations of motion are given by

$$\dot{r}_k(t) = v_k(t), \qquad r_k(0) = r_{\text{init},k}, \qquad t \in [0, n_{\text{t}} \nu_{\text{s},k} \Delta T], \qquad k \in \{0, \ldots, n_{\text{w}} - 1\}, \tag{3.9a}$$

$$\dot{v}_k(t) = m^{-1} R(\phi_k(t), \theta_k(t), \psi_k(t))[0, 0, u_{1,k}(t)]^{\text{T}} - [0, 0, g]^{\text{T}}, \quad r_k(n_{\text{t}} \nu_{\text{s},k} \Delta T) = r_{\text{end},k}, \tag{3.9b}$$

$$\begin{bmatrix} \dot{\phi}_k(t) \\ \dot{\theta}_k(t) \\ \dot{\psi}_k(t) \end{bmatrix} = \Gamma^{-1}(\phi_k(t), \theta_k(t)) \omega_k(t), \qquad \begin{bmatrix} \phi_k(0) \\ \theta_k(0) \\ \psi_k(0) \end{bmatrix} = \begin{bmatrix} \phi_{0,k} \\ \theta_{0,k} \\ \psi_{0,k} \end{bmatrix}, \tag{3.9c}$$

$$\dot{\omega}_k(t) = I^{-1} \left(\begin{bmatrix} u_{2,k}(t) \\ u_{3,k}(t) \\ u_{4,k}(t) \end{bmatrix} - \omega_k^{\times}(t) I \omega_k(t) \right), \qquad \begin{bmatrix} \phi_k(n_{\text{t}} \nu_{\text{s},k} \Delta T) \\ \theta_k(n_{\text{t}} \nu_{\text{s},k} \Delta T) \\ \psi_k(n_{\text{t}} \nu_{\text{s},k} \Delta T) \end{bmatrix} = \begin{bmatrix} \phi_{\text{f},k} \\ \theta_{\text{f},k} \\ \psi_{\text{f},k} \end{bmatrix}, \tag{3.9d}$$

where

$$R(\phi_k, \theta_k, \psi_k) \triangleq \begin{bmatrix} \cos\psi_k & -\sin\psi_k & 0 \\ \sin\psi_k & \cos\psi_k & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_k & 0 & \sin\theta_k \\ 0 & 1 & 0 \\ -\sin\theta_k & 0 & \cos\theta_k \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi_k & -\sin\phi_k \\ 0 & \sin\phi_k & \cos\phi_k \end{bmatrix},$$
$$(\phi_k, \theta_k, \psi_k) \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right) \times \left(-\frac{\pi}{2}, \frac{\pi}{2}\right) \times [0, 2\pi), \tag{3.10}$$

captures the UAV's attitude relative to the inertial reference frame $\mathbb{I}$,

$$\Gamma(\phi_k, \theta_k) \triangleq \begin{bmatrix} 1 & 0 & -\sin\theta_k \\ 0 & \cos\phi_k & \cos\theta_k \sin\phi_k \\ 0 & -\sin\phi_k & \cos\theta_k \cos\phi_k \end{bmatrix}, \tag{3.11}$$

and $\nu_{\text{s},k} \in \{1, \ldots, n_{\text{w}} - k\}$ denotes the user-defined *path stride*; we recall that $\Gamma(\phi_k, \theta_k)$, $k \in \{0, \ldots, n_{\text{w}} - 1\}$, is invertible for all $(\phi_k, \theta_k) \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right) \times \left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$ [22, Chap. 2].

Detailed discussions on the path stride are provided in Sects. 3.4.6 and 3.4.7 below, and a discussion on the boundary conditions $r_{\text{init},k}$ and $r_{\text{end},k} \in \mathbb{R}^3$ is provided in Sect. 3.4.7. The initial conditions $\phi_{0,k}$, $\theta_{0,k}$, and $\psi_{0,k}$, $k \in \{0, \ldots, n_{\text{w}} - 1\}$, are user-defined. Employing (3.9a) and (3.9b) to produce reference trajectories, $\phi_{0,k}$, $\theta_{0,k}$, and $\psi_{0,k}$, $k \in \{0, \ldots, n_{\text{w}} - 1\}$, are set as the UAV's angular position at the time the MPC algorithm is initiated. The endpoint conditions for (3.9c) and (3.9d) are set so that $\phi_{\text{f},k}$, $k \in \{0, \ldots, n_{\text{w}} - 1\}$, $\theta_{\text{f},k}$, and $\psi_{\text{f},k}$ are the Euler angles of the 3-2-1 rotation sequence underlying the rotation matrix

$$R_{\text{f},k} \triangleq \left[\hat{n}_{x,k}, \hat{n}_{y,k}, \hat{n}_{z,k}\right]^{\text{T}}, \tag{3.12}$$

where $\hat{n}_{x,k} \triangleq \|\hat{r}_{k+\nu_{\text{s},k}} - \hat{r}_k\|^{-1} \left(\hat{r}_{k+\nu_{\text{s},k}} - \hat{r}_k\right)$, $\hat{n}_{y,k} \triangleq \hat{n}_{x,k}^{\times} [0, \varepsilon_k, g]^{\text{T}} / \|\hat{n}_{x,k}^{\times} [0, \varepsilon_k, g]^{\text{T}}\|$, $\hat{n}_{z,k} \triangleq \hat{n}_{x,k}^{\times} \hat{n}_{y,k}$, and ε_k is such that if $\hat{n}_{x,k} \neq [0, 0, g]^{\text{T}}$, then $\varepsilon_k = 0$, and if $\hat{n}_{x,k} = [0, 0, g]^{\text{T}}$, then $\varepsilon_k \neq 0$ is small and user-defined. The rotation matrix $R_{\text{f},k}$, $k \in \{0, \ldots, n_{\text{w}} - 1\}$, captures the attitude of a UAV, whose roll axis intersects both $\hat{r}_k$ and $\hat{r}_{k+\nu_{\text{s},k}}$. This matrix is well-defined since the LPA^* algorithm underlying the

proposed path planner does not generate two identical consecutive waypoints, that is, $\hat{r}_k \neq \hat{r}_{k+1}$, for all $k \in \{0, \ldots, n_\mathrm{w} - 1\}$. Moreover, $\hat{n}_{y,k}$, $k \in \{0, \ldots, n_\mathrm{w} - 1\}$, has been designed so that $\hat{n}_{x,k}$ and $[0, \varepsilon_k, g]^\mathrm{T}$ are not collinear, and hence $R_{\mathrm{f},k}$ is always a rotation matrix.

Quadcopter UAVs are underactuated [21], since only four of their six degrees of freedom can be controlled directly. In this chapter, we are interested in regulating the UAV's position $r_k(\cdot)$, $k \in \{0, \ldots, n_\mathrm{w} - 1\}$, and its yaw angle $\psi_k(\cdot)$ since the cameras' focal axes is aligned with the UAV's roll axis. To this goal, setting $\left[r_k^\mathrm{T}(t), \psi_k(t)\right]^\mathrm{T}$, $t \in [0, n_\mathrm{t} \nu_{\mathrm{s},k} \Delta T]$, $k \in \{0, \ldots, n_\mathrm{w} - 1\}$, as the *measured output*, and applying Proposition 5.1.2 of [23], we verify that the dynamical system given by (3.9) and (3.7) has vector relative degree $\{4, 4, 4, 2\}$, and

$$
\begin{aligned}
\begin{bmatrix} r_k^{(4)}(t) \\ \ddot{\psi}_k(t) \end{bmatrix} &= f\left(r_k(t), \phi_k(t), \theta_k(t), \psi_k(t), \omega_k(t), u_{1,k}(t)\right) \\
&\quad + G^{-1}\left(r_k(t), \phi_k(t), \theta_k(t), \psi_k(t), u_{1,k}(t)\right) \begin{bmatrix} \eta_{1,k}(t) \\ u_{2,k}(t) \\ u_{3,k}(t) \\ u_{4,k}(t) \end{bmatrix}, \\
\begin{bmatrix} r_k(0) \\ r_k(n_\mathrm{t} \nu_{\mathrm{s},k} \Delta T) \end{bmatrix} &= \begin{bmatrix} r_{\mathrm{init},k} \\ r_{\mathrm{end},k} \end{bmatrix}, \quad \begin{bmatrix} \ddot{r}_k(0) \\ \ddot{r}_k(n_\mathrm{t} \nu_{\mathrm{s},k} \Delta T) \end{bmatrix} = \begin{bmatrix} \ddot{r}_{\mathrm{init},k} \\ \ddot{r}_{\mathrm{end},k} \end{bmatrix}, \\
\begin{bmatrix} \psi_k(0) \\ \psi_k(n_\mathrm{t} \nu_{\mathrm{s},k} \Delta T) \end{bmatrix} &= \begin{bmatrix} \psi_{0,k} \\ \psi_{\mathrm{f},k} \end{bmatrix}, \qquad t \in [0, n_\mathrm{t} \nu_{\mathrm{s},k} \Delta T],
\end{aligned} \tag{3.13}
$$

where

$$
G^{-1}(r_k, \phi_k, \theta_k, \psi_k, u_{1,k}) \triangleq \begin{bmatrix}
m\mathrm{s}\phi_k\mathrm{s}\psi_k + m\mathrm{c}\phi_k\mathrm{c}\psi_k\mathrm{s}\theta_k & m\mathrm{c}\phi_k\mathrm{s}\psi_k\mathrm{s}\theta_k - m\mathrm{c}\psi_k\mathrm{s}\phi_k & m\mathrm{c}\phi_k\mathrm{c}\theta_k & 0 \\
\frac{I_1 m(\mathrm{c}\phi_k\mathrm{s}\psi_k - \mathrm{c}\psi_k\mathrm{s}\phi_k\mathrm{s}\theta_k)}{u_{1,k}} & -\frac{I_1 m(\mathrm{c}\phi_k\mathrm{c}\psi_k + \mathrm{s}\psi_k\mathrm{s}\phi_k\mathrm{s}\theta_k)}{u_{1,k}} & -\frac{I_1 m\mathrm{c}\theta_k\mathrm{s}\phi_k}{u_{1,k}} & 0 \\
\frac{I_2 m\mathrm{c}\psi_k\mathrm{c}\theta_k}{u_{1,k}} & \frac{I_2 m\mathrm{c}\theta_k\mathrm{s}\psi_k}{u_{1,k}} & -\frac{I_2 m\mathrm{s}\theta_k}{u_{1,k}} & 0 \\
-\frac{I_3 m\mathrm{c}\psi_k\mathrm{c}\theta_k\mathrm{s}\phi_k}{u_{1,k}\mathrm{c}\phi_k} & -\frac{I_3 m\mathrm{c}\theta_k\mathrm{s}\psi_k\mathrm{s}\phi_k}{u_{1,k}\mathrm{c}\phi_k} & \frac{I_3 m\mathrm{s}\theta_k\mathrm{s}\phi_k}{u_{1,k}\mathrm{c}\phi_k} & \frac{I_3 m\mathrm{c}\theta_k}{\mathrm{c}\phi_k}
\end{bmatrix}, \tag{3.14}
$$

$\mathrm{c}\alpha = \cos\alpha$, $\alpha \in \mathbb{R}$, $\mathrm{s}\alpha = \sin\alpha$, $I = \mathrm{diag}(I_1, I_2, I_3)$, and $f : \mathbb{R}^3 \times \left(-\frac{\pi}{2}, \frac{\pi}{2}\right) \times \left(-\frac{\pi}{2}, \frac{\pi}{2}\right) \times [0, 2\pi) \times \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}^4$; an expression for $f(\cdot, \cdot, \cdot, \cdot, \cdot, \cdot)$ is omitted for brevity. It holds that $\det G(r_k, \phi_k, \theta_k, \psi_k, u_{1,k}) = \frac{u_{1,k}^2 \cos\phi_k}{m^3 \det(I) \cos\theta_k}$, $(r_k, \phi_k, \theta_k, \psi_k, u_{1,k}) \in \mathbb{R}^3 \times \left(-\frac{\pi}{2}, \frac{\pi}{2}\right) \times \left(-\frac{\pi}{2}, \frac{\pi}{2}\right) \times [0, 2\pi) \times (0, \infty)$, and hence, $G(\cdot, \cdot, \cdot, \cdot, \cdot, \cdot)$ is invertible if and only if $u_1 \neq 0$ since $\phi_k \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$. The constraints on the UAV's roll and pitch angles and the control input, namely, the conditions whereby $\phi_k(t) \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$, $t \in [0, n_\mathrm{t} \nu_{\mathrm{s},k} \Delta T]$, $k \in \{0, \ldots, n_\mathrm{w} - 1\}$, $\theta_k(t) \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$, and $u_{1,k}(t) > 0$, are enforced by the proposed trajectory planning algorithm; for details, see Sect. 3.4.6 below.

It follows from (3.9) that for the UAV's attitude to be captured by the triplet $(\phi_{0,k}, \theta_{0,k}, \psi_{0,k})$ at $t = 0$ and the triplet $(\phi_{\mathrm{f},k}, \theta_{\mathrm{f},k}, \psi_{\mathrm{f},k})$ at $t = n_\mathrm{t} \nu_{\mathrm{s},k} \Delta T$, it must

hold that

$$\ddot{r}_{\mathrm{init},k} = m^{-1} R(\phi_{0,k}, \theta_{0,k}, \psi_{0,k})[0, 0, u_{1,0,k}]^{\mathrm{T}} - [0, 0, g]^{\mathrm{T}}, \quad k \in \{0, \ldots, n_{\mathrm{w}} - 1\}, \tag{3.15}$$

$$\ddot{r}_{\mathrm{end},k} = m^{-1} R(\phi_{\mathrm{f},k}, \theta_{\mathrm{f},k}, \psi_{\mathrm{f},k})[0, 0, u_{1,\mathrm{f},k}]^{\mathrm{T}} - [0, 0, g]^{\mathrm{T}}. \tag{3.16}$$

Thus, given $\ddot{r}_{\mathrm{init},k}, \ddot{r}_{\mathrm{end},k} \in \mathbb{R}^3$ as user-defined, we deduce $u_{1,0,k}, u_{1,\mathrm{f},k} \in \mathbb{R}$, namely, the boundary conditions of (3.7), from (3.15) and (3.16). The boundary conditions on the UAV's acceleration $\ddot{r}_{\mathrm{init},k}$ and $\ddot{r}_{\mathrm{end},k} \in \mathbb{R}^3$ are provided in Sect. 3.4.7 below.

Next, let

$$\begin{aligned}\zeta(r_k, \phi_k, \theta_k, \psi_k, \omega_k, u_{1,k}, \lambda_k) \triangleq G(r_k, \phi_k, \theta_k, \psi_k, \omega_k, u_{1,k})\Bigg(- f(r_k, \phi_k, \theta_k, \psi_k, \omega_k, u_{1,k}) \\ + \begin{bmatrix} A_{r,0} r_k(t) + A_{r,1} \dot{r}_k(t) + A_{r,2} \ddot{r}_k(t) + A_{r,3} r_k^{(3)}(t) \\ A_{\psi,0} \psi_k(t) + A_{\psi,1} \dot{\psi}_k(t) \end{bmatrix} + \begin{bmatrix} B_r \\ B_\psi \end{bmatrix} \lambda_k \Bigg), \\ (r_k, \phi_k, \theta_k, \psi_k, \omega_k, u_{1,k}, \lambda_k) \in \mathbb{R}^3 \times \left(-\frac{\pi}{2}, \frac{\pi}{2}\right) \times \left(-\frac{\pi}{2}, \frac{\pi}{2}\right) \times [0, 2\pi) \times \mathbb{R}^3 \times \mathbb{R} \times \mathbb{R}^4, \\ k \in \{0, \ldots, n_{\mathrm{w}} - 1\},\end{aligned} \tag{3.17}$$

where

$$\tilde{A}_r \triangleq \begin{bmatrix} 0_{3\times3} & \mathbf{1}_3 & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & \mathbf{1}_3 & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & \mathbf{1}_3 \\ A_{r,0} & A_{r,1} & A_{r,2} & A_{r,3} \end{bmatrix} \in \mathbb{R}^{12\times12}, \qquad \tilde{A}_\psi \triangleq \begin{bmatrix} 0 & 1 \\ A_{\psi,0} & A_{\psi,1} \end{bmatrix} \in \mathbb{R}^{2\times2}, \tag{3.18}$$

are Hurwitz,

$$\tilde{B}_r \triangleq \begin{bmatrix} 0_{9\times4} \\ B_r \end{bmatrix} \in \mathbb{R}^{12\times4}, \qquad \tilde{B}_\psi \triangleq \begin{bmatrix} 0 \\ B_\psi \end{bmatrix} \in \mathbb{R}^{2\times4} \tag{3.19}$$

and the pairs $(\tilde{A}_r, \tilde{B}_r)$ and $(\tilde{A}_\psi, \tilde{B}_\psi)$ are controllable. If

$$\left[\eta_{1,k}(t), u_{2,k}(t), u_{3,k}(t), u_{4,k}(t)\right]^{\mathrm{T}} = \zeta_k(t), \quad t \in [0, n_{\mathrm{t}} \nu_{\mathrm{s},k} \Delta T], \quad k \in \{0, \ldots, n_{\mathrm{w}} - 1\}, \tag{3.20}$$

where $\zeta_k(t)$ denotes $\zeta(r_k(t), \phi_k(t), \theta_k(t), \psi_k(t), \omega_k(t), u_{1,k}(t), \lambda_k(t))$ for brevity, then the UAV's equations of motion (3.9) are output-feedback linearized and

$$\dot{\chi}_k(t) = \tilde{A} \chi_k(t) + \tilde{B} \lambda_k(t), \qquad t \in [0, n_{\mathrm{t}} \nu_{\mathrm{s},k} \Delta T], \tag{3.21}$$

where $\chi_k(t) \triangleq \left[r_k^{\mathrm{T}}(t), \dot{r}_k^{\mathrm{T}}(t), \ddot{r}_k^{\mathrm{T}}(t), r_k^{(3)\mathrm{T}}(t), \psi_k(t), \dot{\psi}_k(t)\right]^{\mathrm{T}}$, $\tilde{A} \triangleq \mathrm{blockdiag}(\tilde{A}_r, \tilde{A}_\psi)$ and $\tilde{B} \triangleq \left[\tilde{B}_r^{\mathrm{T}}, \tilde{B}_\psi^{\mathrm{T}}\right]^{\mathrm{T}}$. Thus, it follows from (3.21) that the discrete-time, linearized,

zero-order hold [24], output-feedback linearized equations of motion of a quadcopter UAV are given by

$$\chi_k((j+1)\Delta T) = A\chi_k(j\Delta T) + B\lambda_k(j\Delta T),$$
$$j \in \{i, \ldots, n_{\mathrm{t}}\nu_{\mathrm{s},k} - 1\}, \qquad i \in \{0, \ldots, n_{\mathrm{t}}\nu_{\mathrm{s},k} - 1\}, \qquad k \in \{0, \ldots, n_{\mathrm{w}} - 1\}, \tag{3.22}$$

where $A = e^{\tilde{A}\Delta T}$ and $B = \int_0^{\Delta T} e^{\tilde{A}\sigma}\mathrm{d}\sigma\,\tilde{B}$; the boundary conditions for (3.21) and (3.22) are the same boundary conditions as for (3.13). Equation (3.22) serves as an equality constraint for the model predictive control algorithm employed to outline tactical reference trajectories for UAVs.

Given $\lambda_k(j\Delta T)$, $j \in \{i, \ldots, n_{\mathrm{t}}\nu_{\mathrm{s},k} - 1\}$, $i \in \{0, \ldots, n_{\mathrm{t}}\nu_{\mathrm{s},k} - 1\}$, $k \in \{0, \ldots, n_{\mathrm{w}} - 1\}$, the total thrust force's virtual control input $\eta_{1,k}(j\Delta T)$, the roll moment produced by the UAV's propellers $u_{2,k}(j\Delta T)$, the pitch moment $u_{3,k}(j\Delta T)$, and the yaw moment $u_{4,k}(j\Delta T)$ are deduced from (3.20) at $t = j\Delta T$ employing $r_k(j\Delta T)$, $\phi_k(j\Delta T)$, $\theta_k(j\Delta T)$, $\psi_k(j\Delta T)$, and their derivatives estimated by the UAV's navigation system. Then, the total thrust force is computed as $u_{1,k}(j\Delta T) = [1, 0]\delta_k(j\Delta T)$, $j \in \{i, \ldots, n_{\mathrm{t}} - 1\}$, $i \in \{0, \ldots, n_{\mathrm{t}} - 1\}$, $k \in \{0, \ldots, n_{\mathrm{w}} - 1\}$, where

$$\delta_k((j+1)\Delta T) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}\delta_k(j\Delta T) + \frac{1}{2}\begin{bmatrix} \Delta T^2 \\ 2\Delta T \end{bmatrix}\mathbf{e}_{1,4}^{\mathrm{T}}\zeta(j\Delta T), \tag{3.23}$$

is the discrete-time zero-order hold counterpart of (3.7). Finally, the thrust force produced by each propeller is computed by applying (3.8).

3.4.4 Collision Avoidance Constraints

The constraint set, which contains the UAV, excludes obstacle points, and is sufficiently large to contain viable reference trajectories, is captured by

$$F_{r,k}(i\Delta T)r_k(i\Delta T) \leq\leq f_{r,k}(i\Delta T), \qquad i \in \{0, \ldots, n_{\mathrm{t}}\nu_{\mathrm{s},k}\}, \qquad k \in \{0, \ldots, n_{\mathrm{w}} - 1\}, \tag{3.24}$$

where the $F_{r,k} : \mathbb{R} \to \mathbb{R}^{n_{\mathrm{coll}} \times 3}$ is such that

$$\mathbf{e}_{q,n_{\mathrm{coll}}}^{\mathrm{T}} F_{r,k}(i\Delta T) \triangleq \begin{bmatrix} \cos(\mu_9 + 2\mu_9(q-1))\cos(\mu_{10} + 2\mu_{10}(q-1)) \\ \sin(\mu_9 + 2\mu_9(q-1))\cos(\mu_{10} + 2\mu_{10}(q-1)) \\ (-1)^q \sin(\mu_{10} + 2\mu_{10}(q-1)) \end{bmatrix}^{\mathrm{T}},$$
$$q \in \{1, \ldots, n_{\mathrm{coll}}\}, \tag{3.25}$$

$\mu_9 > 0$ and $\mu_{10} > 0$ are user-defined and such that $n_{\text{coll}} \triangleq 16\pi^2/(\mu_9\mu_{10})$ is an even integer and denotes the number of collision avoidance constraints, $f_{r,k} : \mathbb{R} \to \mathbb{R}^{n_{\text{coll}}}$ is such that

$$\mathbf{e}_{q,n_{\text{coll}}}^{\mathrm{T}} f_{r,k}(i\Delta T) \triangleq \min \left\{ \min_{p \in \mathcal{S}_q \cap \mathcal{O}_{\text{c},i,k}} \mathbf{e}_{q,n_{\text{coll}}}^{\mathrm{T}} F_{r,k}(i\Delta T)p, \right. \\ \left. \min_{s \in \{1,\ldots,n_{\text{coll}}\}\setminus\{q\}} \mathbf{e}_{q,n_{\text{coll}}}^{\mathrm{T}} F_{r,k}(i\Delta T)p_s(i\Delta T) \right\}, \tag{3.26}$$

$\mathcal{S}_q \subset \mathbb{R}^3$ is a right square spherical pyramid, whose apex is centered in the UAV, whose bisector is given by (3.25), and whose sides have user-defined length,

$$p_s(i\Delta T) \triangleq \arg\min_{p \in \mathcal{S}_s \cap \mathcal{O}_{\text{c},i,k}} F_{r,k}^{\mathrm{T}}(i\Delta T)\mathbf{e}_{s,n_{\text{coll}}}\mathbf{e}_{s,n_{\text{coll}}}^{\mathrm{T}} F_{r,k}(i\Delta T)p, \qquad s \in \{1,\ldots,n_{\text{coll}}\}, \tag{3.27}$$

and $\mathcal{O}_{\text{c},i,k} \subseteq \mathcal{O}$ captures a set of obstacles' points that are more likely to be impacted by the UAV. In practice, the convex constraint set with affine boundaries captured by (3.24) is produced by partitioning a sphere centered in the UAV in n_{coll} right square spherical pyramids, and constructing hyperplanes orthogonal to the bisector of each pyramid [25]. The use of the affine inequalities (3.24) to capture collision avoidance constraints is essential to guarantee fast numerical solutions of the proposed trajectory planning problem.

To deduce $\mathcal{O}_{\text{c},i,k}$, we propose an approach that we named the *Bubble Bath algorithm*. This original method selects those obstacle points that are within some user-defined radius from the UAV's position, are contained in the half-space defined by the UAV's velocity vector, and are directly accessible to the UAV. In the Bubble Bath algorithm, first, we define the closed set

$$\overline{\mathcal{H}}_\rho(r_k(i\Delta T)) \triangleq \left\{x \in \overline{\mathcal{B}}_\rho(r_k(i\Delta T)) \cap \mathcal{O} : x^{\mathrm{T}} v_k(i\Delta T) \geq 0\right\}, \\ i \in \{0,\ldots,n_{\text{t}}\nu_{\text{s},k}\}, \qquad k \in \{0,\ldots,n_{\text{w}}-1\}, \tag{3.28}$$

where $\rho \triangleq \max\{\|v_k(i\Delta T)\|, \overline{v}\}i\Delta T$ and $\overline{v} > 0$ is user-defined. This closed set captures all obstacle points that are within a distance ρ from the UAV's position at $t = i\Delta T$ and are included in the closed half-space determined by the UAV's velocity vector. It follows from the definition of ρ that the size of $\mathcal{O}_{\text{c},i,k}$ increases with the UAV's velocity; the parameter $\overline{v}$ captures a user-defined safety margin. Next, we compute the closed *parent ellipsoid* $\overline{\mathcal{E}}(P_k(i\Delta T), r_k(i\Delta T), c_k(i\Delta T)/\mu_{11})$, $i \in \{0,\ldots,n_{\text{t}}\nu_{\text{s},k}\}$, $k \in \{0,\ldots,n_{\text{w}}-1\}$, where $P_k(i\Delta T) \in \mathbb{R}^{3\times 3}$ and $c_k(i\Delta T) \in \mathbb{R}$ are solutions of the quadratic programming problem with cost function

$$\min c_k(i\Delta T) \tag{3.29}$$

and constraints

$$g_1(P_k(i\Delta T), r_k(i\Delta T), w_\alpha, c_k(i\Delta T)) \geq 0, \qquad \alpha \in \{1, \ldots, n_{\text{UAV}}\}, \tag{3.30a}$$
$$g_2(P_k(i\Delta T), r_k(i\Delta T), w_\beta, c_k(i\Delta T)) \leq 0, \qquad \beta \in \{1, \ldots, n_{\mathcal{H}}\}, \tag{3.30b}$$
$$g_3(P_k(i\Delta T)) \leq 0_{3\times 3}, \tag{3.30c}$$
$$g_4(c_k(i\Delta T)) < 0, \tag{3.30d}$$

$g_1(P, r, w, c) \triangleq (w - r)^{\mathrm{T}} P (w - r) - c, \qquad (P, r, w, c) \in \mathbb{R}^{3\times 3} \times \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}$, $g_2(P, r, w, c) \triangleq (w - r)^{\mathrm{T}} P (w - r) - c$, $g_3(P) \triangleq P + \mathbf{1}_3$, $g_4(c) \triangleq c$, $\mu_{11} > 1$ is a user-defined safety margin, and $n_{\mathcal{H}}$ denotes the cardinality of $\overline{\mathcal{H}}_\rho(r_k(i\Delta T))$; conditions (3.30c) and (3.30d) guarantee that $\overline{\mathcal{E}}(P_k(i\Delta T), r_k(i\Delta T), c_k(i\Delta T)/\mu_{11})$ is an ellipsoid, and (3.30a) and (3.30b) guarantee that the parent ellipsoid includes all the UAV's points $w_\alpha, \alpha \in \{1, \ldots, n_{\text{UAV}}\}$, and excludes all points of $\overline{\mathcal{H}}_\rho(r_k(i\Delta T))$. Successively, for all $i \in \{0, \ldots, n_{\text{t}} \nu_{\text{s},k}\}$ and $k \in \{0, \ldots, n_{\text{w}} - 1\}$, we compute ℓ closed *children ellipsoids*, $\overline{\mathcal{E}}(\tilde{P}_{k,l}(i\Delta T), \tilde{r}_{k,l}(i\Delta T), \tilde{c}_{k,l}(i\Delta T))$, $l \in \{1, \ldots, \ell\}$, where ℓ is user-defined, $\tilde{P}_{k,l}(i\Delta T) \in \mathbb{R}^{3\times 3}$ and $\tilde{c}_{k,l}(i\Delta T) \in \mathbb{R}$ are solutions of the quadratic programming problem with cost function

$$\min \tilde{c}_{k,l}(i\Delta T) \tag{3.31}$$

and constraints given by (3.30b)–(3.30d) with $P_k(\cdot)$ replaced by $\tilde{P}_{k,l}(\cdot)$, $r_k(\cdot)$ replaced by $\tilde{r}_{k,l}(\cdot)$,

$$\tilde{r}_{k,l}(i\Delta T) \triangleq \sqrt{\frac{c_k(i\Delta T) P_k^{-1}(i\Delta T)}{\mu_{11}}} \begin{bmatrix} \cos\gamma_l \cos\eta_l \\ \sin\gamma_l \cos\eta_l \\ \sin\eta_l \end{bmatrix} + r_k(i\Delta T), \tag{3.32}$$

$c_k(\cdot)$ replaced by $\tilde{c}_{k,l}(\cdot)$, and $\gamma_l, \eta_l \in \mathbb{R}$ are user-defined. Thus, children ellipsoids exclude all obstacle points in $\overline{\mathcal{H}}_\rho(r_k(i\Delta T))$, and it follows from (3.32) that children ellipsoids are Finally, we define

$$\begin{aligned} \mathcal{O}_{\text{c},i,k} \triangleq & \left\{ r \in \overline{\mathcal{H}}_\rho(r_k(i\Delta T)) : \| c_k^{-1/2}(i\Delta T) P_k^{1/2}(i\Delta T)(r - r_k(i\Delta T)) \| \leq 1 + \mu_{12} \right\} \\ & \cup_{l=1}^{\ell} \left\{ r \in \overline{\mathcal{H}}_\rho(r_k(i\Delta T)) : \| \tilde{c}_{k,l}^{-1/2}(i\Delta T) \tilde{P}_{k,l}^{1/2}(i\Delta T)(r - \tilde{r}_{k,l}(i\Delta T)) \| \leq 1 + \mu_{12} \right\}, \\ & i \in \{0, \ldots, n_{\text{t}} \nu_{\text{s},k}\}, \qquad k \in \{0, \ldots, n_{\text{w}} - 1\}, \end{aligned} \tag{3.33}$$

where $\mu_{12} > 0$ denotes a user-defined safety margin. In practice, the subset of obstacle points $\mathcal{O}_{\text{c},i,k}$ is given by the set of all points that are within a distance smaller than or equal to μ_{12} from any of the children ellipsoids or $\overline{\mathcal{E}}(P_k(i\Delta T), r_k(i\Delta T), c_k(i\Delta T))$; note that it follows from (3.30b) that $\partial\overline{\mathcal{E}}(P_k(i\Delta T), r_k(i\Delta T), c_k(i\Delta T))$ comprises at least one point of the obstacles' set $\mathcal{O}$.

This technique has been named Bubble Bath algorithm because the children ellipsoids are centered at the parent ellipsoid and cover unoccupied voxels surrounding the UAV and hence, mimic the behavior of bubbles in a bathtub. The larger the num-

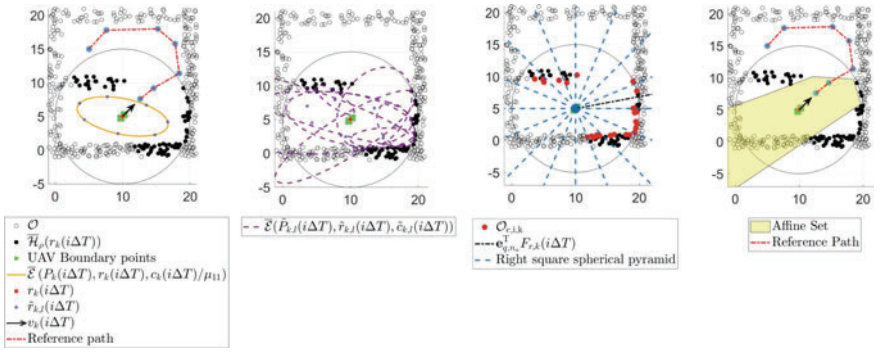

Fig. 3.13 Schematic representation of the proposed approach to generate affine constraint sets from the set of occupied voxels $\mathcal{O}$. Firstly, the parent ellipsoid, which contains the UAV and excludes all obstacle points, is created. Successively, multiple children ellipsoids, which are centered on the boundary of the parent ellipsoid, are generated. Then, the subset of obstacle points $\mathcal{O}_{c,i,k}$ are computed as the set of points of $\mathcal{O}$ within some user-defined distance from the parent or any of the children ellipsoids. Finally, the affine set is computed as the subset of the state space that contains the UAV and excludes all points of $\mathcal{O}_{c,i,k}$

ber of children ellipsoids, the more accurate is this technique in identifying occupied voxels that could be impacted by the UAV.

Figure 3.13 provides a graphical representation of the proposed approach to identifying the subset $\mathcal{O}_{c,i,k}$ of the obstacles' set $\mathcal{O}$ for some $i \in \{0, \ldots, n_t \nu_{s,k}\}$ and $k \in \{0, \ldots, n_w - 1\}$, and create the affine constraint set captured by (3.24). Although the proposed technique applies to three-dimensional problems, a two-dimensional example has been shown for clarity of visualization.

3.4.4.1 Study on the Computational Cost of the Bubble Bath Algorithm

To assess the efficiency of the Bubble Bath algorithm and compare its performance with the iterative regional inflation by SDP (IRIS) algorithm [26–28], the safe flight corridor (SFC) algorithm [29], and the minimum distance collision avoidance (MDCA) algorithm [2], we consider as a test case a two-story house comprising multiple rooms on both floors and a set of stairs; see Fig. 3.14 for top-down views of both floors. Within this map, we consider 10 paths, each departing from a different room on the first floor, converging at the bottom of the stairs, proceeding to the second floor parallel along the stairs, and reaching the same point on the second floor. For each path, the distance between waypoints is 25 voxels; the shortest path includes 175 waypoints, and the longest path includes 400 waypoints. Each of the 4 algorithms being tested is executed 50 times along each path on a personal computer equipped with an Intel 8Core *i*7, 4.5 GHz processor, a 32 Gb DDR4 memory, and MATLAB® 2021b.

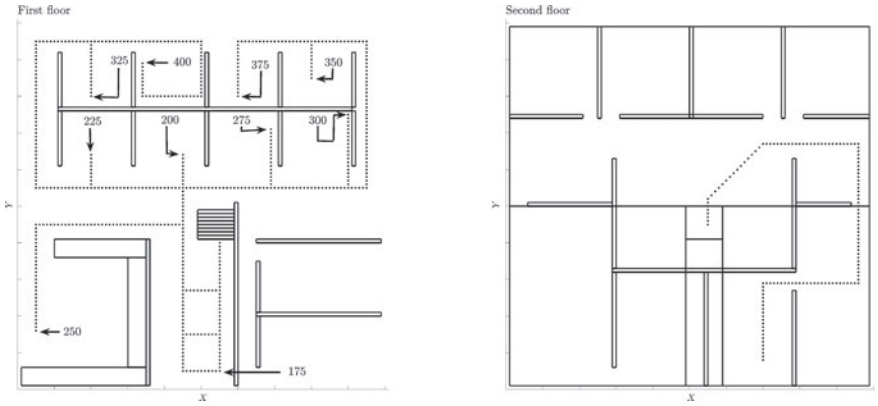

Fig. 3.14 Paths used for testing the computational efficiency of the Bubble Bath algorithm. The beginning of each path is marked with the numbers of waypoints

Table 3.1 Coefficients of the first-order linear regressions relating path length and computation times

Algorithm	Slope	Offset
Bubble Bath	8.669	64.354
MDCA	3.374	12.920
SFC	19.684	3.693
IRIS	12.537	16.510

As shown in Fig. 3.15, MDCA produces the lowest computational time, and the Bubble Bath algorithm produces the second-best computational time. Furthermore, for all algorithms, the computational time increases with the number of points in the path. From an analysis of the data of a first-order regression for each algorithm, it is apparent that MDCA grows the least with the number of waypoints, and Bubble Bath has the second slowest increasing computational time per number of waypoints; for details, see Table 3.1.

Figure 3.16 shows statistical information about the volumes of the sets produced by the Bubble Bath algorithm, MDCA, SFC, and IRIS. Although SFC and IRIS produce the largest average and the largest minimum constraint sets, the average maximum constraint set volume is similar across the four algorithms. Finally, Fig. 3.17 shows that, except for IRIS, all algorithms produce constraint sets, whose volumes' standard deviations are similar. Finally, it is worthwhile remarking that SFC is able to produce unbounded sets, but in a relatively small number.

In conclusion, for consistency of generated volumes and computational speed, MDCA and Bubble Bath are preferable. If the volume of the constraint set is more important than the computational time, then IRIS would be a preferable choice. Finally, the Bubble Bath algorithm has the most consistent volume generated and

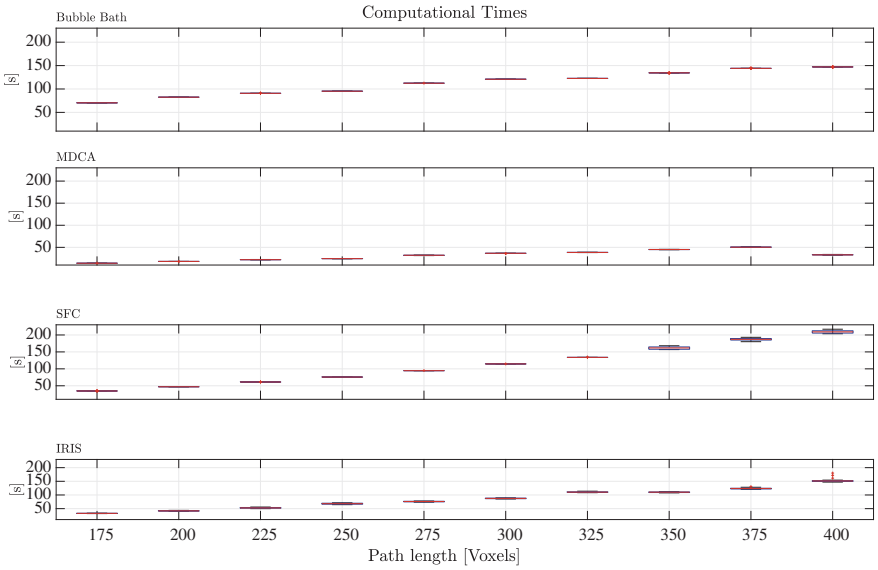

Fig. 3.15 Average computational times of the Bubble Bath algorithm, the MDCA, SFC, and IRIS along each path

computational time throughout testing, making it worthy of consideration for applications where predictability is a top priority.

3.4.5 Constraints on the UAV's Attitude

As discussed in Sect. 3.4.3, the output-feedback linearized equations of motion (3.22) yield for $\phi_k(j\Delta T) \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$, $j \in \{i, \ldots, n_{\mathrm{t}} \nu_{\mathrm{s},k}\}$, $i \in \{0, \ldots, n_{\mathrm{t}} \nu_{\mathrm{s},k}\}$, $k \in \{0, \ldots, n_{\mathrm{w}} - 1\}$, and $\theta_k(j\Delta T) \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$. Furthermore, the user may want to impose constraints on the yaw angle so that $\psi_k(j\Delta T) \in (-\psi_{\max}, \psi_{\max})$, $j \in \{i, \ldots, n_{\mathrm{t}} \nu_{\mathrm{s},k}\}$, $i \in \{0, \ldots, n_{\mathrm{t}} \nu_{\mathrm{s},k}\}$, $k \in \{0, \ldots, n_{\mathrm{w}} - 1\}$, where $\psi_{\max} > 0$. For instance, by setting $\psi_{\max} = \frac{\pi}{2}$, the user imposes that the UAV's roll axis always points in the direction of motion.

In this chapter, the constraints on the yaw angle are imposed by means of the inequality constraints

$$-\psi_k(j\Delta T) \leq -\psi_{\mathrm{f},k} + \psi_{\max}, \quad j \in \{i, \ldots, n_{\mathrm{t}} \nu_{\mathrm{s},k}\}, \quad i \in \{0, \ldots, n_{\mathrm{t}} \nu_{\mathrm{s},k}\}, \quad k \in \{0, \ldots, n_{\mathrm{w}} - 1\}, \tag{3.34a}$$

$$\psi_k(j\Delta T) \leq \psi_{\mathrm{f},k} + \psi_{\max}, \tag{3.34b}$$

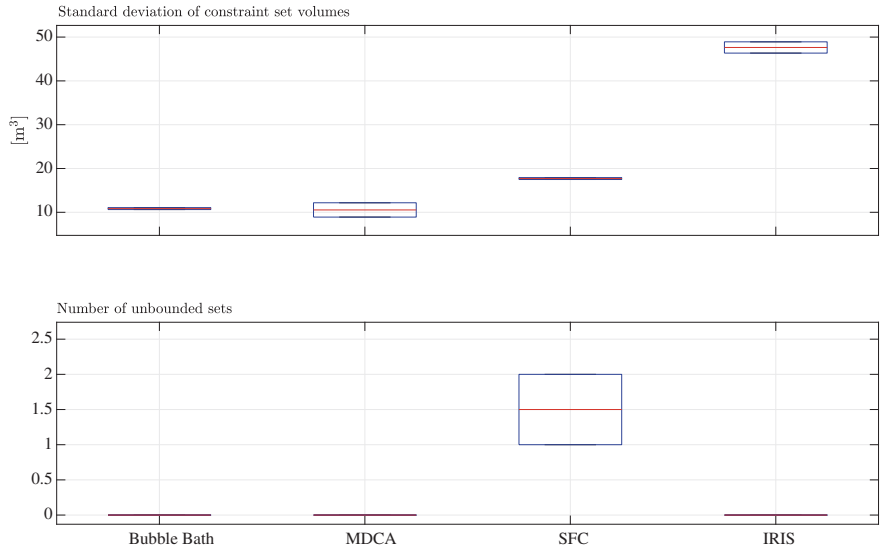

Fig. 3.16 Boxplots of some characteristics of the volumes of the collision avoidance sets produced by the Bubble Bath algorithm, MDCA, SFC, and IRIS

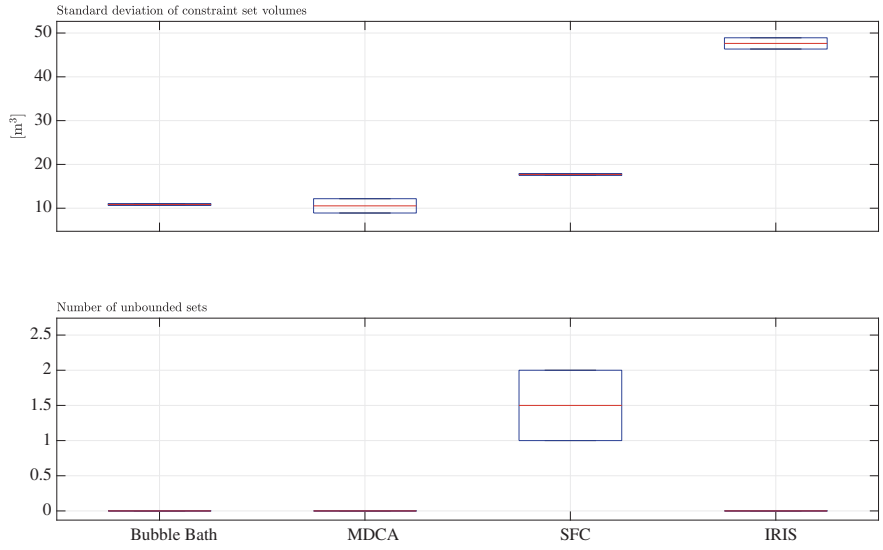

Fig. 3.17 Standard deviations of the volumes generated by each algorithm, the number of unbounded constraint sets per path, and the number of obstacles contained along each of the 10 paths

or, alternatively, by

$$F_\psi(i\Delta T)\psi_k(j\Delta T) \leq\leq f_{\psi,k}(i\Delta T), \quad i \in \{0, \ldots, n_\mathrm{t}\nu_{\mathrm{s},k}\}, \quad k \in \{0, \ldots, n_\mathrm{w} - 1\}, \tag{3.35}$$

where

$$F_\psi(i\Delta T) \triangleq [-1, 1]^\mathrm{T}, \tag{3.36a}$$

$$f_{\psi,k}(i\Delta T) \triangleq \left[\psi_{\max} - \psi_{\mathrm{f},k}, \psi_{\max} + \psi_{\mathrm{f},k}\right]^\mathrm{T}. \tag{3.36b}$$

As discussed in Sect. 3.4.6 below, the constraints on the pitch and roll angles are imposed by means of barrier functions.

3.4.6 Cost Function Definition

The proposed trajectory planner produces reference trajectories that minimize a cost function that encourages four objectives, namely, following waypoints produced by the path planner, reducing the control effort, encouraging tactical or reckless behaviors, and discouraging the UAV's pitch and roll angles from approaching the boundaries of the interval $(-\pi/2, \pi/2)$. A cost function that meets the first three objectives is given by

$$\begin{aligned}&\tilde{J}[r_k(0), \psi_k(0), \lambda_k(\cdot)]\\ &\triangleq \sum_{l=k}^{k+\nu_{\mathrm{s},k}-1} \nu_{\mathrm{dis}}^{l-k}\ell_\mathrm{f}(r_l(n_\mathrm{t}\Delta T), \psi_l(n_\mathrm{t}\Delta T)) + \sum_{l=k}^{k+\nu_{\mathrm{s},k}-1}\sum_{i=0}^{n_\mathrm{t}-1} \nu_{\mathrm{dis}}^{l-k}\tilde{\ell}(r_l(i\Delta T), \psi_l(i\Delta T), \lambda_l(i\Delta T)),\\ &\qquad\qquad k \in \{0, \ldots, n_\mathrm{w} - 1\},\end{aligned} \tag{3.37}$$

where $\nu_\mathrm{dis} \in (0, 1)$,

$$\tilde{\ell}(r_k, \psi_k, \lambda_k) \triangleq \begin{bmatrix}\tilde{r}_k\\ \lambda_k\end{bmatrix}^\mathrm{T} \tilde{R}_k \begin{bmatrix}\tilde{r}_k\\ \lambda_k\end{bmatrix} + q_\psi(\psi_k - \psi_{\mathrm{f},k})^2 + \tilde{q}_r^\mathrm{T}\tilde{r}_k + \tilde{q}_\lambda^\mathrm{T}\lambda_k, \quad (r_k, \psi_k, \lambda_k) \in \mathbb{R}^3 \times \mathbb{R} \times \mathbb{R}^4, \tag{3.38a}$$

$$\ell_\mathrm{f}(r_k, \psi_k) \triangleq \left(r_k - \hat{r}_{k+\nu_{\mathrm{s},k}}\right)^\mathrm{T} R_{r,\mathrm{f}} \left(r_k - \hat{r}_{k+\nu_{\mathrm{s},k}}\right) + q_\psi(\psi_k - \psi_{\mathrm{f},k})^2, \tag{3.38b}$$

$\tilde{R}_k \triangleq \begin{bmatrix}\tilde{R}_r & \tilde{R}_{r,\lambda}\\ \tilde{R}_{r,\lambda}^\mathrm{T} & R_\lambda\end{bmatrix}$, $\tilde{R}_r \in \mathbb{R}^{3\times3}$ is symmetric, $\tilde{R}_{r,\lambda} \in \mathbb{R}^{3\times4}$, and $R_\lambda \in \mathbb{R}^{4\times4}$ is positive-definite and such that

$$\tilde{R}_r - 2\tilde{R}_{r,\lambda}^\mathrm{T} R_\lambda^{-1} \tilde{R}_{r,\lambda} > 0, \tag{3.39}$$

$\tilde{q}_r \in \mathbb{R}^3$, $\tilde{q}_\lambda \in \mathbb{R}^4$,

$$\tilde{r}_k(i\Delta T) \triangleq \mu_{13}\left[r_k(i\Delta T) - \hat{r}_{k+\nu_{\mathrm{s},k}}\right] + (1-\mu_{13}) f_{\mathrm{sat}}\left(\mu_{14}(\hat{r}_k - r_{O_{\mathrm{c},i,k}})\right)\left[r_k(i\Delta T) - r_{O_{\mathrm{c},i,k}}\right], \quad i \in \{0,\ldots,n_{\mathrm{t}}\nu_{\mathrm{s},k}-1\}, \tag{3.40}$$

$\mu_{13} \in (0,1]$ and $\mu_{14} > 0$ are user-defined, $r_k(\cdot)$ verifies (3.22), $f_{\mathrm{sat}}(w) \triangleq \dfrac{\mathrm{sat}\,(\|w\|)}{\|w\|}$, $w \in \mathbb{R}^n$, $r_{O_{\mathrm{c},i,k}} \triangleq \mathrm{d}_2(\hat{r}_k, O_{\mathrm{c},i,k})$, and $R_{r,\mathrm{f}} \in \mathbb{R}^{3\times 3}$ is symmetric and nonnegative-definite, and $q_\psi > 0$. By the Schur complement condition on the positive-definiteness of block-matrices [30, pp. 7–8], (3.39) implies that $\tilde{R}$ is symmetric and positive-definite.

As discussed in Sect. 3.4.7 below, the path stride $\nu_{\mathrm{s},k}, k \in \{0,\ldots,n_{\mathrm{w}}-1\}$, allows to set boundary conditions for the trajectory planning problem at the waypoint $\hat{r}_{k+\nu_{\mathrm{s},k}}$. Thus, the reference trajectory does not traverse the waypoints $\hat{r}_{k+1},\ldots,\hat{r}_{k+\nu_{\mathrm{s},k}-1}$. However, for all $k \in \{0,\ldots,n_{\mathrm{w}}-1\}$, the need to outline a reference trajectory that approximates the waypoints $\hat{r}_{k+1},\ldots,\hat{r}_{k+\nu_{\mathrm{s},k}-1}$, is captured by the cost function (3.37).

The term ν_{dis}^{l-k}, $k \in \{0,\ldots,n_{\mathrm{w}}-1\}$, in (3.37) is a *discount factor* applied to the objective function so that, while the UAV approaches $\hat{r}_{k+1}$, the influence of $\hat{r}_{k+2},\ldots,\hat{r}_{k+\nu_{\mathrm{s},k}}$ on the cost function is less marked. Furthermore, if l is considerably larger than k, then the reference trajectory is more likely to intersect the set of unexplored voxels $\mathcal{V}_{\mathrm{unexplored}}$. Therefore, the waypoints $\hat{r}_{k+2},\ldots,\hat{r}_{k+\nu_{\mathrm{s},k}}$ should contribute less to the overall cost, since confidence in the feasibility of these trajectories is lower.

The *Mayer's function* (3.38b) captures the UAV's need to reach the waypoint $\hat{r}_{k+\nu_{\mathrm{s},k}}$, $k \in \{0,\ldots,n_{\mathrm{w}}-1\}$, and point its roll axis toward this waypoint. The first term on the right-hand side of (3.40) captures the UAV's distance from $\hat{r}_{k+\nu_{\mathrm{s},k}}$, $k \in \{0,\ldots,n_{\mathrm{w}}-1\}$, and the second term on the right-hand side of (3.40) captures the UAV's distance from the obstacles' set O. The *Lagrangian function* (3.38a) captures the UAV's competing needs of reaching the waypoint $\hat{r}_{k+\nu_{\mathrm{s},k}}$, $k \in \{0,\ldots,n_{\mathrm{w}}-1\}$, and coasting the obstacles' set, while pointing the onboard cameras toward $\hat{r}_{k+\nu_{\mathrm{s},k}}$. Indeed, if $\mu_{13} = 1$, then $\tilde{r}_k(i\Delta T) = r_k(i\Delta T) - \hat{r}_{k+\nu_{\mathrm{s},k}}$, $i \in \{0,\ldots,n_{\mathrm{t}}\nu_{\mathrm{s},k}-1\}$, $k \in \{0,\ldots,n_{\mathrm{w}}-1\}$, and minimizing (3.37) induces a reckless behavior in the UAV, since its only goal is to reach the waypoint, and if $\mu_{13} \in (0,1)$, then coasting the obstacles' set becomes a higher priority. The function $f_{\mathrm{sat}}(\cdot)$ in (3.40) reduces the attractive effect of obstacles at a distance from the waypoint $\hat{r}_k, k \in \{0,\ldots,n_{\mathrm{w}}-1\}$, that is larger than μ_{14}^{-1}. Indeed, $f_{\mathrm{sat}}(w) \to 1$ as $\|w\| \to 0$, $f_{\mathrm{sat}}(\mu_{14}w) = 1$ for all $w \in \overline{\mathcal{B}}_{\mu_{14}^{-1}}(0_n)$, $f_{\mathrm{sat}}(\mu_{14}w) < 1$ for all $w \in \mathbb{R}^n \setminus \overline{\mathcal{B}}_{\mu_{14}^{-1}}(0_n)$, and $\lim_{w\to\infty} f_{\mathrm{sat}}(\mu_{14}w) = 0$. Detailed discussions on the role of the weighing matrices $\tilde{R}_r$, $\tilde{R}_{r,\lambda}$, R_λ, the weighing term q_ψ, and the user-defined parameters μ_{13} and μ_{14} in designing trajectory planners for tactical vehicles based on the model predictive control framework are presented in [2, 20].

To recast (3.37) as an explicit function of both the UAV's position $r_k(\cdot)$, $k \in \{0, \ldots, n_\mathrm{w} - 1\}$, and $\lambda_k(\cdot)$, we substitute (3.40) in (3.38a), and note that minimizing (3.37) is equivalent to minimizing

$$\sum_{l=k}^{k+\nu_{\mathrm{s},k}-1} \nu_{\mathrm{dis}}^{l-k}\hat{\ell}_\mathrm{f}(r_l(n_\mathrm{t}\Delta T), \psi_l(n_\mathrm{t}\Delta T)) + \sum_{l=k}^{k+\nu_{\mathrm{s},k}-1}\sum_{i=0}^{n_\mathrm{t}-1} \nu_{\mathrm{dis}}^{l-k}\ell(r_l(i\Delta T), \psi_l(i\Delta T), \lambda_l(i\Delta T)),$$
$$k \in \{0, \ldots, n_\mathrm{w} - 1\}, \quad (3.41)$$

where

$$\hat{\ell}(r_k, \psi_k, \lambda_k) \triangleq \begin{bmatrix} r_k \\ \lambda_k \end{bmatrix}^\mathrm{T} R_k \begin{bmatrix} r_k \\ \lambda_k \end{bmatrix} + q_\psi(\psi_k - \psi_{\mathrm{f},k})^2 + q_{r,k}^\mathrm{T}(\tau) r_k + q_{\lambda,k}^\mathrm{T}(\tau)\lambda_k,$$
$$(r_k, \psi_k, \lambda_k) \in \mathbb{R}^3 \times \mathbb{R} \times \mathbb{R}^4, \quad (3.42a)$$

$$R_k \triangleq \begin{bmatrix} R_{r,k} & R_{r,\lambda,k} \\ R_{r,\lambda,k}^\mathrm{T} & R_\lambda \end{bmatrix}, \quad (3.42b)$$

$$R_{r,k}(\tau) \triangleq \left[1 + \mu_{13}(1 - f_\mathrm{sat}\left(\mu_{14}(\hat{r}_k - r_{O_{\mathrm{c},i,k}})\right))\right] \tilde{R}_r, \quad (3.42c)$$

$$R_{r,\lambda,k}(\tau) \triangleq \left[1 + \mu_{13}(1 - f_\mathrm{sat}\left(\mu_{14}(\hat{r}_k - r_{O_{\mathrm{c},i,k}})\right))\right] \tilde{R}_{r,\lambda}, \quad (3.42d)$$

$$q_{r,k}(\tau) \triangleq \left[1 + \mu_{13}(1 - f_\mathrm{sat}\left(\mu_{14}(\hat{r}_k - r_{O_{\mathrm{c},i,k}})\right))\right] \Big(\tilde{q}_r - 2\tilde{R}_r \Big[\mu_{13}\hat{r}_{k+1} + (1 - \mu_{13}) f_\mathrm{sat}\left(\mu_{14}(\hat{r}_k - r_{O_{\mathrm{c},i,k}})\right) r_{O_{\mathrm{c},i,k}}\Big]\Big), \quad (3.42e)$$

$$q_{\lambda,k}(\tau) \triangleq \tilde{q}_\lambda - 2\tilde{R}_{r,\lambda}^\mathrm{T}\left[\mu_{13}\hat{r}_{k+1} + (1 - \mu_{13}) f_\mathrm{sat}\left(\mu_{14}(\hat{r}_k - r_{O_{\mathrm{c},i,k}})\right) r_{O_{\mathrm{c},i,k}}\right], \quad (3.42f)$$

Next, we modify (3.41) to meet the fourth design objective, namely, constraining the UAV's pitch and roll angles.

Employing a model predictive control framework to minimize (3.41), in this chapter the UAV's reference trajectories and the corresponding control inputs are computed iteratively at each time step over the discrete time horizon $\{i\Delta T, \ldots, n_\mathrm{t}\nu_{\mathrm{s},k}\Delta T\}$, $i \in \{0, \ldots, n_\mathrm{t}\nu_{\mathrm{s},k} - 1\}$, as minimizers of

$$J[i\Delta T, r_k(i\Delta T), \psi_k(i\Delta T), \lambda_k(\cdot)]$$
$$\triangleq \sum_{l=k}^{k+\nu_{\mathrm{s},k}-1} \nu_{\mathrm{dis}}^{l-k}\ell_\mathrm{f}(r_k(n_\mathrm{t}\Delta T), \psi_k(n_\mathrm{t}\Delta T)) + \sum_{j=i}^{n_\mathrm{t}-1} \ell(j\Delta T, r_k(j\Delta T), \psi_k(j\Delta T), \lambda_k(j\Delta T))$$
$$+ \sum_{l=k+1}^{k+\nu_{\mathrm{s},k}-1}\sum_{i=0}^{n_\mathrm{t}-1} \nu_{\mathrm{dis}}^{l-k}\ell(i\Delta T, r_l(i\Delta T), \psi_l(i\Delta T), \lambda_l(i\Delta T)),$$
$$i \in \{0, \ldots, n_\mathrm{t}\nu_{\mathrm{s},k} - 1\}, \quad k \in \{0, \ldots, n_\mathrm{w} - 1\}, \quad (3.43)$$

where

Table 3.2 Parameters needed to define the boundary conditions for (3.13) for $k \in \{0, \ldots, n_{\mathrm{w}} - 1\}$

$i = 0$	$r_{\mathrm{init},k} = r_k(n_{\mathrm{t}} \nu_{\mathrm{s},k} \Delta T)$,
	$\ddot{r}_{\mathrm{init},k} = \ddot{r}_k(n_{\mathrm{t}} \nu_{\mathrm{s},k} \Delta T)$,
	$r_{\mathrm{end},k} = \hat{r}_{k+\nu_{\mathrm{s},k}}$,
	$\ddot{r}_{\mathrm{end},k} = \hat{a}_{k+\nu_{\mathrm{s},k}}$
$i \in \{1, \ldots, n_{\mathrm{t}} \nu_{\mathrm{s},k} - 1\}$	$r_{\mathrm{init},k} = r_k(i \Delta T)$,
	$\ddot{r}_{\mathrm{init},k} = \ddot{r}_k(i \Delta T)$,
	$r_{\mathrm{end},k} = \hat{r}_{k+\nu_{\mathrm{s},k}}$,
	$\ddot{r}_{\mathrm{end},k} = \hat{a}_{k+\nu_{\mathrm{s},k}}$

$$\ell(j\Delta T, r_k, \psi_k, \lambda_k) \triangleq g_{\mathrm{barrier},k}^{-1}(j\Delta T)\hat{\ell}(j\Delta T, r_k, \psi_k, \lambda_k), \\ (j, r_k, \psi_k, \lambda_k) \in \{i, \ldots, n_{\mathrm{t}} - 1\} \times \mathbb{R}^3 \times \mathbb{R} \times \mathbb{R}^4, \tag{3.44}$$

$$g_{\mathrm{barrier},k}(j\Delta T) = \left[\phi_{\max}^2 - \phi_k^2(j\Delta T)\right]\left[\theta_{\max}^2 - \theta_k^2(j\Delta T)\right] \\ \cdot \prod_{p=1}^{4}\left[T_{p,k}(j\Delta T) - T_{\min}\right] \prod_{p=1}^{4}\left[T_{\max} - T_{p,k}(j\Delta T)\right], \tag{3.45}$$

$\phi_{\max} \in \left(0, \frac{\pi}{2}\right)$ and $\theta_{\max} \in \left(0, \frac{\pi}{2}\right)$ are user-defined. The role of the *penalty function* $g_{\mathrm{barrier},k}(\cdot)$ is that if $\phi_k(j\Delta T) \in (-\phi_{\max}, \phi_{\max})$, $\theta_k(j\Delta T) \in (-\theta_{\max}, \theta_{\max})$, $T_{p,k}(j\Delta T) > T_{\min}$, $p = 1, \ldots, 4$, and $T_{p,k}(j\Delta T) < T_{\max}$, then the model predictive control algorithm underlying the proposed trajectory planning subsystem penalizes all admissible solutions $\{r_k(j\Delta T), \psi_k(j\Delta T), \lambda_k(j\Delta T)\}_{j=i}^{n_{\mathrm{t}}\nu_{\mathrm{s},k}}$, $k \in \{0, \ldots, n_{\mathrm{w}}\}$, that induce a decreasing subsequence of $\left\{g_{\mathrm{barrier},k}(j\Delta T)\right\}_{j=i}^{n_{\mathrm{t}}\nu_{\mathrm{s},k}}$. The second term on the right-hand side of (3.43) includes a summation that recedes as the UAV approaches $n_{\mathrm{t}}\Delta T$, and a fixed summation for path segments $l > k$.

It follows from (3.13), (3.17), and (3.21) that the UAV's roll and pitch angles are not part of the state vector $\chi_k(\cdot)$, $k \in \{0, \ldots, n_{\mathrm{w}} - 1\}$, and their dynamics are unobservable in the measured output $\left[r_k^{\mathrm{T}}(\cdot), \psi_k(\cdot)\right]^{\mathrm{T}}$. Therefore, both $\phi_k(j\Delta T)$, $j \in \{i, \ldots, n_{\mathrm{t}}\nu_{\mathrm{s},k}\}$, $k \in \{0, \ldots, n_{\mathrm{w}} - 1\}$, and $\theta_k(j\Delta T)$ are deduced from (3.9c) and (3.9d) with control input given by (3.17) and $\lambda_k(j\Delta T)$ given by the model predictive control law. It is also worthwhile to note that constraints on $\phi_k(\cdot)$, $k \in \{0, \ldots, n_{\mathrm{w}} - 1\}$, and $\theta_k(\cdot)$ are imposed through the barrier function $g_{\mathrm{barrier}}(\cdot)$ Lagrangian function. Similarly, it follows from (3.20), (3.23), and (3.8) that each propeller's thrust force $T_{p,k}(\cdot)$ is a nonlinear, time-varying function of $\lambda_k(\cdot)$ and hence, constraints on the positivity of each propeller's thrust force must be imposed through $g_{\mathrm{barrier}}(\cdot)$. The user-defined parameter $T_{\min} > 0$ captures the fact that the majority of commercial-off-the-shelf motors and propellers for quadcopters do not allow to produce arbitrarily small thrust force. The user-defined parameter $T_{\max} > 0$ captures the saturation constraint for each propeller.

3.4.7 Boundary Conditions

The boundary conditions on the UAV's translational dynamic equations (3.13) or, alternatively, (3.21) and (3.22), are given in Table 3.2. The conditions on the UAV's initial position and acceleration namely, $r_{\text{init},k}$ and $\ddot{r}_{\text{init},k}$, ensure that the UAV's reference position and acceleration are equal to the UAV's position and acceleration when the MPC algorithm is initialized, respectively. The conditions on the UAV's final position, namely, $r_{\text{end},k}$, ensures that the reference trajectory traverses the waypoint $\hat{r}_{k+\nu_{\text{s},k}}$, $k \in \{0, \ldots, n_{\text{w}} - 1\}$. The condition on the UAV's final acceleration are defined so that

$$\hat{a}_{k+\nu_{\text{s},k}} \triangleq \hat{a}\left(\mathrm{d}_2(\hat{r}_{k+\nu_{\text{s},k}}, \mathcal{O})\right)\hat{d}_k, \qquad k \in \{0, \ldots, n_{\text{w}} - 1\}, \tag{3.46}$$

where

$$\hat{a}(\alpha) \triangleq \begin{cases} 0, & \text{if } \alpha \in [0, \mu_{15}], \\ \frac{\mu_{17}}{\mu_{16}-\mu_{15}}(\alpha - \mu_{15}), & \text{if } \alpha \in [\mu_{15}, \mu_{16}], \\ \mu_{17}, & \text{otherwise}, \end{cases} \tag{3.47a}$$

$$\hat{d}_k \triangleq \mu_{18}\frac{\hat{r}_{k+\nu_{\text{s},k}} - \hat{r}_{k+\nu_{\text{s},k}-1}}{\|\hat{r}_{k+\nu_{\text{s},k}} - \hat{r}_{k+\nu_{\text{s},k}-1}\|} + (1-\mu_{18})\frac{\hat{r}_{k+\nu_{\text{s},k}+1} - \hat{r}_{k+\nu_{\text{s},k}}}{\|\hat{r}_{k+\nu_{\text{s},k}+1} - \hat{r}_{k+\nu_{\text{s},k}}\|}, \tag{3.47b}$$

and $\mu_{15}, \mu_{16}, \mu_{17} > 0$ and $\mu_{18} \in [0, 1]$ are user-defined.

If $\mathrm{d}_2(\hat{r}_{k+\nu_{\text{s},k}}, \mathcal{O}) \leq \mu_{15}$, $k \in \{0, \ldots, n_{\text{w}} - 1\}$, that is, if the UAV is sheltered by an obstacle at a distance smaller than μ_{15}, then $\|\ddot{r}_{\text{end},k}\| = 0$, since it is desirable for the UAV not to accelerate while operating in safer areas. Alternatively, if $\mathrm{d}_2(\hat{r}_{k+\nu_{s,k}}, \mathcal{O}) > \mu_{16}$, $k \in \{0, \ldots, n_{\text{w}} - 1\}$, that is, if the UAV is too far from any obstacle to be considered sheltered, then $\|\ddot{r}_{\text{end},k}\| = \mu_{17}$, since it is desirable that the UAV rapidly traverses unsafer areas. Finally, if $\mathrm{d}_2(\hat{r}_{k+\nu_{\text{s},k}}, \mathcal{O}) \in [\mu_{15}, \mu_{16}]$, $k \in \{0, \ldots, n_{\text{w}} - 1\}$, then $\|\ddot{r}_{\text{end},k}\|$ is set to increase as a linear function of $\mathrm{d}_2(\hat{r}_{k+\nu_{\text{s},k}}, \mathcal{O})$ from 0 to μ_{17}. If $\mu_{18} = 1$, then $\ddot{r}_{\text{end},k}$, $k \in \{0, \ldots, n_{\text{w}} - 1\}$, is set to point in the direction joining the waypoint $\hat{r}_{k+\nu_{\text{s},k}}$, which is set as the endpoint for the trajectory, and $\hat{r}_{k+\nu_{\text{s},k}-1}$. Alternatively, if $\mu_{18} = 0$, then $\ddot{r}_{\text{end},k}$, $k \in \{0, \ldots, n_{\text{w}} - 1\}$, is set to point in the direction joining the waypoint $\hat{r}_{k+\nu_{\text{s},k}}$, which is set as the endpoint for the trajectory, and $\hat{r}_{k+\nu_{\text{s},k}+1}$. Setting $\mu_{18} = 1$ allows to align the endpoint condition on the acceleration with the direction the reference trajectory would follow by interpolating all waypoints, that is, by setting $\nu_{\text{s},k} = 1$. This option may induce overly aggressive maneuvers in the presence of sharp turns in the reference path. Setting $\mu_{18} = 0$ allows to align the endpoint condition on the acceleration with the direction the reference trajectory would follow by interpolating all waypoints after $\hat{r}_{k+\nu_{\text{s},k}}$ has been reached. This option may induce less aggressive maneuvers in the presence of sharp turns in the reference path.

For brevity, Table 3.2 does not include the initial conditions at the beginning of the mission and the endpoint conditions at the end of the mission. In this study, the UAV is designed to start and end its mission with zero acceleration.

3.4.8 Numerical Solution of the Trajectory Planning Problem

In this chapter, the pair $(\chi_k(j\Delta T), \lambda_k(j\Delta T))$ is computed for each $j \in \{i, \ldots, n_t - 1\}$, $i \in \{0, \ldots, n_t - 1\}$, and for all $k \in \{0, \ldots, n_w - 1\}$, as minimizers of the cost function (3.43) subject to (3.22), (3.24), and (3.35) by applying the framework presented in [2]. A difference with the work presented in [2] is that in this work, the cost function underlying the trajectory planner is a function of time, since the roll angle $\phi_k(\cdot)$, $k \in \{0, \ldots, n_w - 1\}$, the pitch angle $\theta_k(\cdot)$, and the thrust force produced by the pth propeller $T_{p,k}(\cdot)$, $p = 1, \ldots, 4$, are not a part of the state vector $\chi_k(\cdot)$. Thus, the quadratic programming algorithm used to solve the trajectory planning problem needs to be updated at each time step.

3.5 Numerical Simulations Results

In this section, we present and analyze the results from two hardware-in-the-loop simulations aimed at showing the features and the applicability of the proposed guidance system. In particular, Sect. 3.5.1 below shows the results of the proposed system while requiring a reckless trajectory to be followed. Successively, Sect. 3.5.2 shows the results of the proposed system while requiring a tactical trajectory to be followed. The results of the simulations discussed in the following as well as the optimization framework presented thus far can be also found at [31].

3.5.1 Reckless Guidance System Simulation

The results from the first simulation, in which the guidance system's parameters are set to instill a reckless behavior, are shown in Fig. 3.18. Solid black lines indicate the boundaries of obstacles, the green star indicates the UAV's initial position $r_0(0) = [1.0, 1.0, 1.0]^T$ m, the red line indicates the UAV's position $r_k(i\Delta T)$, the blue, purple, and green arrows indicate the orientation of the UAV's roll, pitch, and yaw axes, respectively. The pink and yellow contour plot represents the projection of the explored set $\mathcal{V}_{\text{explored}}$ onto the horizontal plane. Yellow shades indicate that a large percentage of voxels have been detected by the simulated camera, whereas pink shades indicate that a small percentage of voxels have been detected by the simulated camera.

At the beginning of the mission, the simulated camera gathers information for Algorithm 3.1, which generates the first goal point $\hat{r}_{\tilde{\mathcal{P}},0} = [4.6, 1.6, 1.8]^T$ m, a reference path and a reference trajectory are outlined, while avoiding the detected obstacles. Once the UAV rounds the corner near $[7.0, 5.0, 1.0]^T$ m, the first goal is detected at $t = 19.86$ s, triggering Algorithm 3.1 to execute and create a new goal point. This process is iterated 8 times when the final goal point $\hat{r}_{\tilde{\mathcal{P}},8} = [13.8, 3.8, 1.0]^T$ m was

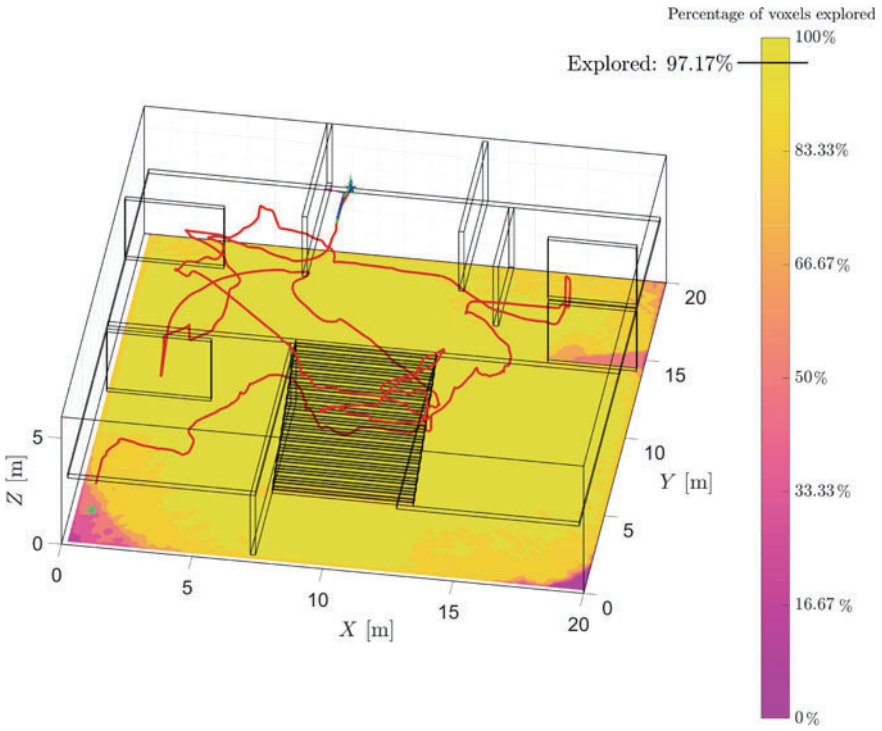

Fig. 3.18 Results from a flight simulation with a reckless parameter set. Solid black lines indicate the boundaries of obstacles, the green star indicates the UAV's start point $r_0(0)$, the red line indicates the UAV's trajectory, the blue star indicates the UAV's final goal point $\hat{r}_{\tilde{\mathcal{P}},8}$. The filled contour plot indicates the sum of the explored voxels at that a given location in the horizontal plane. Bright yellow indicates that every voxel has been detected by the simulated camera, and bright pink indicates that no voxel has been detected by the simulated camera. The coverage mission is completed in 262.70 s after having explored 97.17% of the total number of voxels. On average, 1028 voxels per second are explored, which corresponds to 0.34% of the total number of voxels per second

detected at $t = 262.70\,\text{s}$, and the mission is completed after having explored 97.17% of the total number of voxels. On average, 538.30 voxels per second are explored, which corresponds to 0.18% of the total number of voxels per second.

It is apparent how the UAV's trajectory traverses the open areas, and, hence, exposes the vehicle to potential threats. The average distance from the obstacles' set is 1.97 m, and the standard deviation of the distance from the obstacles' set is 1.22 m. The distance traveled by the UAV is 95.717 m, approximately 10 times the distance between $r_0(0)$ and $\hat{r}_{\tilde{\mathcal{P}},12}$. The average flight speed of 0.40 $\frac{\text{m}}{\text{s}}$ with a standard deviation of 0.51 $\frac{\text{m}}{\text{s}}$. The vehicle's maximum pitch angle $\theta_k(\cdot)$ and maximum roll angle $\phi_k(\cdot)$ were 8.39° and 8.82°, respectively, satisfying the constraints imposed by (3.45).

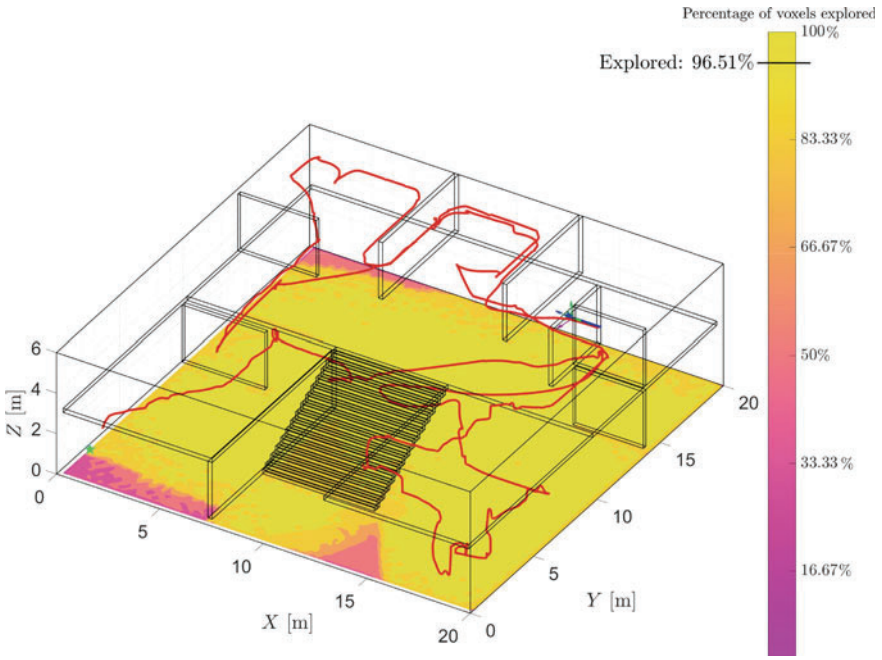

Fig. 3.19 Results from a flight simulation with a tactical parameter set. The coverage mission is completed in 216.12 s after having explored 96.65% of the total number of voxels. On average, 383.00 voxels per second are explored, which corresponds to 0.11% of the total number of voxels per second

3.5.2 Tactical Guidance System Simulation

The results from the second simulation, in which the guidance system's parameters are set to instill a tactical behavior, are shown in Fig. 3.19. As for the first simulation, the UAV starts at $r_0(0) = [1.0, 1.0, 1.0]^{\mathrm{T}}$ m. Once the simulation starts, the simulated camera gathers information for Algorithm 3.1, which determines the first goal point $\hat{r}_{\tilde{\mathcal{P}},0} = [15.0, 5.0, 4.6]^{\mathrm{T}}$ m. The first goal point is on the second floor, and, due to the unknown nature of the environment, the tactical path aims to intersect the ceiling of the first floor. As the vehicle follows the path, the ceiling's occupied voxels are detected, and the collision avoidance algorithm inhibits the vehicle from traveling any further. Thus, the UAV generates a new path, and, eventually, the staircase is discovered and used to reach the second floor. Once the vehicle reaches the second floor, the planned path coasts the walls nearby as it approaches the goal, enabling a tactical behavior. When the UAV rounds the corner of the obstacle near $[10.0, 5.0, 1.0]^{\mathrm{T}}$ m, the first goal point $\hat{r}_{\tilde{\mathcal{P}},0} = [8.6, 5.4, 0.6]^{\mathrm{T}}$ m is detected by the simulated camera at $t = 34.80$ s, triggering Algorithm 3.1 to execute and create a new goal point. By the end of the simulation, 13 goal points are generated. The final goal point, namely, $\hat{r}_{\tilde{\mathcal{P}},12} = [14.6, 16.0, 5.0]^{\mathrm{T}}$ m, is generated at $t = 301.47$ s,

and detected at $t = 308.93$ s. The percentage of the map explored is 96.51%, with approximately 3.30% being undetectable by the simulated camera sensor. The number of voxels explored per second is 383.00, which is approximately 0.11% of the total number of voxels in the map.

Observing Fig. 3.19, it is apparent how the UAV's trajectory exploits the obstacles' set to seek shelter and reduce its exposure. An animation of this trajectory shows how the UAV travels more rapidly across areas away from the obstacles' set. The average distance from the obstacles' set is 1.22 m, which is 0.75 m less than the average distance from the obstacles' set obtained in the reckless simulation, and the standard deviation of the distance from the obstacles' set is 1.01 m. The tactical guidance system enabled the vehicle to exploit shelter in areas that are not densely packed with obstacles. The distance traveled by the UAV is 244.05 m, which is 148.33 m longer than the distance traveled by the UAV with a reckless behavior. This distance is approximately 11 times the distance between $r_0(0)$ and $\hat{r}_{\tilde{\mathcal{P}},12}$. The flight time is 216.12 s, 66.02 s longer than the reckless counterpart. This flight time yields an average flight speed of $0.28\,\frac{\text{m}}{\text{s}}$ with a standard deviation of $0.38\,\frac{\text{m}}{\text{s}}$. The lower average flight speed can be attributed to the fact that, while coasting obstacles, the UAV is programmed to fly at a lower speed. The vehicle's maximum pitch angle $\theta_k(\cdot)$ and maximum roll angle $\phi_k(\cdot)$ are 8.60° and 8.42°, respectively, satisfying the constraints imposed by (3.45).

3.6 Conclusion and Future Work

This chapter presented the first guidance system for autonomous multi-rotor UAVs employed to cover unknown areas while implementing several tactics to minimize the risk of exposure to unknown, potential threats. These tactics include exploring obstacles to seek shelter, proceeding more slowly in safer areas, such as in proximity of obstacles, and accelerating, while traversing more hazardous areas. The extent these strategies need to be pursued can be tuned by user-defined parameters.

This guidance system comprises an optimization-based path planner that is suitable for cluttered, dynamic environments. An original goal selection algorithm allows the proposed path planner to prioritize the need for a systematic coverage or the need for covering largely unexplored, according to the user's preference. The proposed guidance system also includes an optimal control-based trajectory planner that interpolates the waypoints produced by the path planner, while accounting for the UAV's nonlinear dynamics. User-defined constraints on the maximum pitch, roll, and yaw angles are imposed by means of inequality constraints and barrier functions. This trajectory planner also allows the user to choose how closely the reference path needs to be followed. Thus, the proposed guidance system allows the user to request reckless reference paths, which guarantee shorter distances and flight times, and reference trajectories that, while not following the reference path too closely, guarantee sufficiently high levels of cautiousness. An original algorithm, named the bubble bath

algorithm, allows to define convex constraint sets in voxel maps, which define the search space for the trajectory planner.

In the near term, future work directions include verifying the computational advantage of the proposed path planner, which is based on the LPA^* algorithm over alternative search algorithms such as A^*. Additionally, the performance of the proposed Bubble Bath algorithm will be tested against state-of-the-art algorithms such as IRIS (Iterative Regional Inflation by Semidefinite programming) [26–28] and SFC (Safe Flight Corridors) [29]. Finally, flight tests will validate the proposed numerical simulation results.

A key point in the proposed guidance system revolves around optimal trajectory planning, which, after having feedback-linearized the UAV's equations of motion, has been cast as a linear–quadratic optimization problem to produce solutions in real time. The next chapter discusses an alternative approach to the optimal trajectory planning problem, which allows to account for nonlinear differential equations in the trajectory planning process.

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

References

1. Chien WY (2020) Stereo-camera occupancy grid mapping. Master's thesis, Aerospace Engineering
2. Marshall JA, Anderson RB, Chien W-Y, Johnson EN, L'Afflitto A (2021) A guidance system for tactical autonomous unmanned aerial vehicles. J Intell & Robot Syst 103(4):1–36
3. Peng C, Isler V (2019) Adaptive view planning for aerial 3d reconstruction. In: International conference on robotics and automation. IEEE, pp 2981–2987
4. Zhou X, Yi Z, Liu Y, Huang K, Huang H (2020) Survey on path and view planning for UAVs. Virtual Real & Intell Hardw 2(1):56–69
5. Koenig S, Likhachev M, Furcy D (2004) Lifelong planning A^*. Artif Intell 155(1):93–146
6. Koenig S, Likhachev M (2002) D^* lite. In: National conference on artificial intelligence, vol 15. AAAI, Alberta, Canada (2002), pp 476–483
7. Koenig S, Likhachev M (2005) Fast replanning for navigation in unknown terrain. Trans Robot 21(3):354–363
8. Cabreira TM, Brisolara LB, Ferreira PR Jr (2019) Survey on coverage path planning with unmanned aerial vehicles. Drones 3(1):4
9. Shakhatreh H, Sawalmeh AH, Al-Fuqaha A, Dou Z, Almaita E, Khalil I, Othman NS, Khreishah A, Guizani M (2019) Unmanned aerial vehicles (UAVs): a survey on civil applications and key research challenges. IEEE Access 7:48 572–48 634
10. Martz J, Al-Sabban W, Smith RN (2020) Survey of unmanned subterranean exploration, navigation, and localisation. IET Cyber-Syst Robot 2(1):1–13
11. Moysis L, Petavratzis E, Volos C, Nistazakis H, Stouboulos I, Valavanis K (2020) A chaotic path planning method for 3D area coverage using modified logistic map and a modulo tactic. In: International conference on unmanned aircraft systems. IEEE, pp 220–227
12. Wallar A, Plaku E, Sofge DA (2014) A planner for autonomous risk-sensitive coverage (PARCov) by a team of unmanned aerial vehicles. In: IEEE symposium on swarm intelligence. IEEE, pp 1–7
13. He P, Dai S (2013) Stealth coverage multi-path corridors planning for UAV fleet. In: International conference on mechatronic sciences, electric engineering and computer. IEEE, pp 2922–2926

14. Kreyszig E (1989) Introductory functional analysis with applications. Wiley, New York, NY
15. Bernstein DS (2009) Matrix mathematics: theory, facts, and formulas, 2nd edn. Princeton University Press, Princeton, NJ
16. Vasquez-Gomez JI, Gomez-Castaneda C, De Cote EM, Herrera-Lozada JC (2016) Multirotor UAV coverage planning under wind conditions. In: International conference on mechatronics, electronics and automotive engineering. IEEE, pp 32–37
17. Gramajo G, Shankar P (2017) An efficient energy constraint based UAV path planning for search and coverage. Int J Aerosp Eng 2017:1–13
18. Li T, Wang C, de Silva CW et al (2019) Coverage sampling planner for UAV-enabled environmental exploration and field mapping. In: International conference on intelligent robots and systems. IEEE, pp 2509–2516
19. Latombe J-C (2012) Robot motion planning, vol 14. Springer, Berlin, Germany
20. Amrite S (2021) A Taguchi-based approach to tune bio-inspired guidance system for tactical UAVs. Master's thesis, Mechanical Engineering
21. L'Afflitto A, Anderson RB, Mohammadi K (2018) An introduction to nonlinear robust control for unmanned quadrotor aircraft. IEEE Control Syst Mag 38(3):102–121
22. L'Afflitto A (2017) A mathematical perspective on flight dynamics and control. Springer, London, UK
23. Isidori A (1995) Nonlinear control systems. Springer, New York, NY
24. Tsai JS-H, Huang C-C, Guo S-M, Shieh L-S (2011) Continuous to discrete model conversion for the system with a singular system matrix based on matrix sign function. Appl Math Modell 35(8):3893–3904
25. Kavan L, Kolingerova I, Zara J (2006) Fast approximation of convex hull. In: International conference on advances in computer science and technology, Anaheim, CA, pp 101–104
26. Alonso-Mora J, Baker S, Rus D (2017) Multi-robot formation control and object transport in dynamic environments via constrained optimization. Int J Robot Res 36(9):1000–1021
27. Deits R, Tedrake R (2015) Computing large convex regions of obstacle-free space through semidefinite programming. In: Algorithmic foundations of robotics XI. Springer, pp 109–124
28. Deits R, Tedrake R (2015) Efficient mixed-integer planning for UAVs in cluttered environments. In: International conference on robotics and automation. IEEE, pp 42–49
29. Liu S, Watterson M, Mohta K, Sun K, Bhattacharya S, Taylor CJ, Kumar V (2017) Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-D complex environments. IEEE Robot Autom Lett 2(3):1688–1695
30. Boyd S, El Ghaoui L, Feron E, Balakrishnan V (1994) Linear matrix inequalities in system and control theory. SIAM, Philadelphia, PA
31. Marshall JA, Binder P, L'Afflitto A, Tactical coverage of environments using multi-rotor UAVs. YouTube. https://youtu.be/NjulQs-cTTo

Julius A. Marshall received a 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. He 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.

Paul Binder received a B.S. in Aerospace Engineering from the Pennsylvania State University in 2021. He joined the Advanced Control Systems Lab at Virginia Tech under the advisement of Dr. L'Afflitto in the fall of 2021. He is currently pursuing a Master's degree in Aerospace Engineering at Virginia Tech, where he is specializing in autonomous aircraft. His current research interests include object detection, controls, and guidance and navigation.

Andrea L'Afflitto received a B.S. degree in Aerospace Engineering and an M.S. degree in Aerospace Engineering and Astronautics from the University of Napoli "Federico II," Italy, in 2004 and 2006, respectively, an M.S. degree in Mathematics from Virginia Tech in 2010, and a Ph.D. in Aerospace Engineering from Georgia Tech in 2015. He is an Associate Professor with the Grado Department of Industrial and Systems Engineering at Virginia Tech and holds affiliate positions with the Department of Aerospace and Ocean Engineering, the Department of Mechanical Engineering, and the National Security Institute. His current research interests include nonlinear robust control, optimal control, and control of unmanned aerial systems. In 2018, he received the DARPA Young Faculty Award. He is a Senior Editor for the IEEE Transactions of Aerospace and Electronic Systems and is a member of the IEEE Aerospace Controls Technical Committee.