Journal of Intelligent & Robotic Systems (2023) 108:54
https://doi.org/10.1007/s10846-023-01835-0

REGULAR PAPER

UAV Indoor Exploration for Fire-Target Detection and Extinguishing

Halil Utku Unlu[1,2] · Dimitris Chaikalis[1,2] · Athanasios Tsoukalas[2] · Anthony Tzes[2,3]

Received: 30 November 2022 / Accepted: 13 February 2023 / Published online: 10 July 2023

Abstract
A UAV equipped with an RGB-D camera and a downward facing optical flow sensor is set to explore an unknown indoor 3D-volume. The UAV's visual odometry is accomplished through the fusion of IMU, depth readings and the optical flow sensor. While the UAV is moving using a skeletal path to avoid collisions between the surrounding obstacles, it computes features from the environment. These features assist the UAV to localize itself in a self-identified map using the RTAB-Map SLAM. The UAV moves to the fire-alike target-set and for the unexplored area it uses a Chebyshev shortest path from the boundary free cells to the boundary of the target-set. Due to SLAM-induced errors, the UAV employs virtual potential fields and local depth measurements for obstacle avoidance. Upon detection of the target, comprised as a fiducial marker to emulate a fire-source, the UAV uses a ballistic trajectory to propel its payload (fire extinguishing material) towards the target. Simulation studies using Unity for photo-realistic indoor imaging, Gazebo for the UAV dynamics and ROS as intermediate between these components is used to validate the suggested scheme.

Keywords Indoor exploration · Skeletal path · Chebyshev shortest path · RTAB-SLAM · Unity

1 Introduction

Exploration of unknown environments using Unmanned Aerial Vehicles (UAVs) is an active research area, driving advancements in obstacle avoidance, mapping and autonomous navigation. Especially the case of indoor exploration, indicating UAVs navigating unknown environments with confined areas and no GNSS coverage, requires robust solutions to the problems of simultaneous localization and mapping (SLAM) and safe navigation.

✉ Halil Utku Unlu
utku@nyu.edu

✉ Dimitris Chaikalis
dimitris.chaikalis@nyu.edu

Athanasios Tsoukalas
athanasios.tsoukalas@nyu.edu

Anthony Tzes
anthony.tzes@nyu.edu

1 Electrical & Computer Engineering, New York University, Brooklyn 11201, New York, USA

2 Electrical Engineering, New York University Abu Dhabi, Saadiyat Island 129188, Abu Dhabi, UAE

3 Center for Artificial Intelligence and Robotics, New York University Abu Dhabi, Saadiyat Island 129188, Abu Dhabi, UAE

In [32] the UAV navigates in an unknown environment while requiring previous knowledge of certain features. Similarly, [18] presents an algorithm for accurate indoor UAV navigation, using several fixed radio stations to improve the UAV positioning. Such solutions are however not applicable to navigation of purely unknown environments.

Motion planning for the autonomous UAV is another crucial aspect. In case an accurate map of the area is known apriori, there is extensive literature on algorithms capable of planning the shortest possible paths [2, 33], often taking into account the UAV's kinodynamics [17]. However, in the absence of a complete and fully accurate map, path planning is typically preferred to focus on the safety aspect, since an optimally shortest path to the target is unavailable [8].

Other works utilise deep learning with computer vision, in order to achieve autonomous exploration of unknown areas [21, 31]. However, such methods offer no assurances of their viability and are thus less widely implemented than model-based methods. Nevertheless, in the absence of external positioning systems or a-priori maps, computer vision can be a great tool to localization.

Researchers in [3] compared the efficiency of deep neural networks at providing localization in indoor environments. Likewise, [5, 20] outline methods for extracting visual odometry from camera views combined with IMU readings. Coupling visual-inertial odometry localization methods

with the loop-closure capability afforded by SLAM algorithms [11], allows for on-board long-term localization. All visual algorithms however, need to be evaluated and trained on real-world data, otherwise performance suffers upon transition to experiments. This problem is commonly referred to as simulation to reality gap, and recent developments have been surveyed in [36]. Hence the need for photo-realistic simulators, which is addressed in this paper by combining well-established physics engines with state-of-the-art visual rendering software.

In this paper, the UAV is relying only on its own cameras and inertial sensors in order to perform SLAM tasks and plan safe, collision-free paths towards vaguely defined target regions. In order to reliably validate vision-based frameworks in a simulation capacity, it was chosen to utilise a photo-realistic rendering environment (Unity [29]), while an established physics engine (Open Dynamics Engine [25] in Gazebo [14]) is running on the background, accurately simulating vehicle dynamics [22]. This work is an extension of the work in [30], contributing:

- Extensions of the specification of target coordinate to a target region, providing a way to encode uncertainty,
- Demonstrations of the capabilities of specifying multiple disjoint target coordinates,
- Demonstration of performance on higher fidelity synthetic vision data, and
- Demonstration of target detection and ballistic delivery of the payload.

The rest of the paper is organized as follows. In Section 2, the studied problem, stemming from the ICUAS 2022 aerial competition, is analysed to provide a clear understanding of the research goals and the methods utilised. Section 3 describes how the UAV utilizes SLAM and uses the generated map and available sensors to plan collision-free motions. Section 4 shows the flight controllers utilised on the UAV, while Section 5 shows how the agent deals with tracking the target region of interest and performs target acquisition respectively. Finally, Section 6 presents the simulation environments and findings.

2 Problem Statement

The problem of UAV navigation in an unexplored, unknown indoor space, with the goal of reaching a target location is considered in this paper. In order for the UAV to retain full controllability and path-finding capabilities in a GNSS-denied environment, it is equipped with an expanded on-board sensing suite and corresponding algorithms. The sensors include an Inertial Measurement Unit (IMU) providing angular velocity and linear acceleration measurements, an optical flow sensor (OF) comprising of a downward-facing camera and a laser range sensor, and a front-facing RGB-D camera, which allows for online SLAM as well as providing visual odometry for the aerial agent.

Parts of this work were inspired by and developed for fire-fighting competitions, like in the ICUAS 2022 aerial competition, where the UAV is equipped with a fire-extinguishing ball it can use to suppress a small fire. Thus, instead of pure exploration, the end-goal of the system is for the UAV to navigate the unknown area in order to recognise and localize a target of interest.

This target is represented by an artificial reality tag, and a specialized controller is designed to enable the UAV to launch its fire-extinguishing ball at said target from as large a distance as possible, emulating the expected performance in the case of a real fire target.

The proposed framework allows for an uncertain region expected to contain the target to be communicated to the UAV, if available, in order to enable faster convergence to the fire target. The region(s) of interest can be specified as a set of coordinates that are disjoint. The UAV navigator is then capable of planning safe minimum-distance paths to the target region, while probabilistically updating the region's definition based on its own observations.

In the interest of safety, a 2D medial axis-based method is used to provide the UAV with a path guaranteed to lie on the maximum possible distance of all explored obstacles, while a method inspired by potential fields is used to ensure collision avoidance with dynamic or incorrectly-mapped obstacles.

3 SLAM, State Estimation & Path Planning

3.1 RTAB-MAP SLAM

In this work, the RTAB-Map framework [15] is utilised for SLAM given measurements from the front-facing RGB-D camera and UAV's odometry estimates. RTAB-Map uses OctoMap [12] as back-end to generate 3D occupancy maps based on maximum likelihood, while handling the processing of the new map with optimized poses upon loop closures.

RTAB-Map requires odometry information for integrating sensor data to the pose graph. RTAB-Map's own Visual-Inertial Odometry (VIO) is generated based on front-facing camera images with high accuracy. However, in feature-less environments (e.g. blank wall) the algorithm returns null odometry messages. In order to avoid loss of localization, it was selected to fuse the front-facing camera VIO with an expanded sensor suite.

As seen in Fig. 1, apart from the front-facing RGB-D camera, the UAV is endowed with an Optical Flow (OF) [24] sensor comprised of a bottom-facing camera and accompanying laser range finder, along with an Inertial Measurement

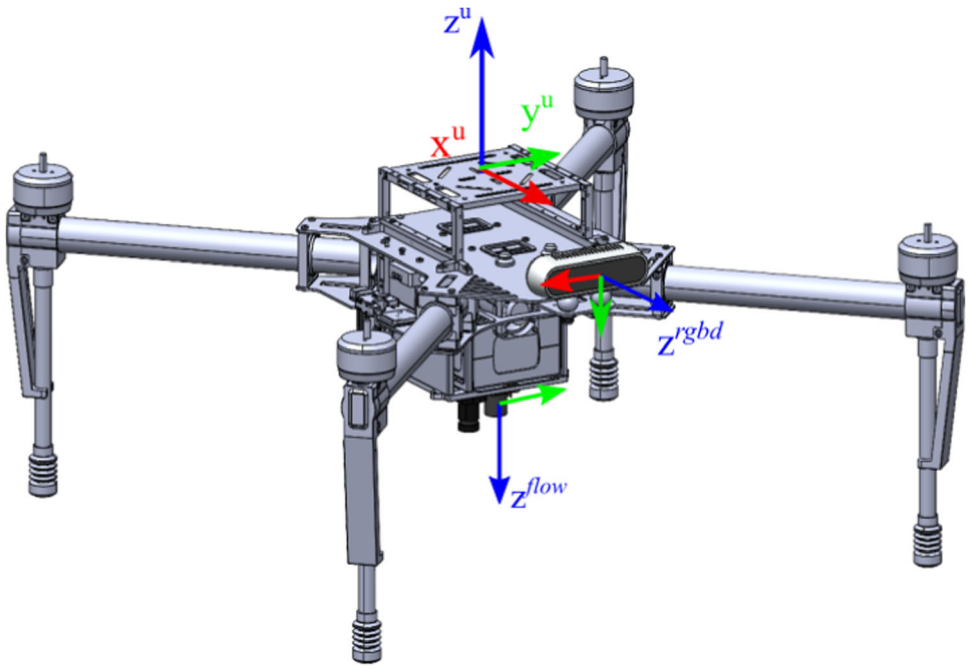

Fig. 1 DJI Matrice 100 Quadrotor with cameras and corresponding coordinate frames

Unit (IMU). The simulated UAV is a DJI Matrice 100 Quadrotor, although the framework is agnostic of the exact UAV platform.

3.2 UAV State Estmation

An Extended Kalman Filter (EKF) is implemented, fusing measurements from all the above sensors, namely angular velocity and linear acceleration measurements from the IMU, spatial velocity and altitude measurements from the OF and spatial velocity and orientation measurements from the VIO.

Let an earth-fixed arbitrary inertial frame w and a frame u fixed on the UAV's center and aligned with its IMU. Then the UAV's state, as maintained at the EKF, is $\chi = \left[\mathbf{p}, \mathbf{v}, \mathbf{\Phi}\right]^\top$, where $\mathbf{p} = [x, y, z]^\top$ the UAV's 3D-position in w, $\mathbf{v} = \dot{\mathbf{p}}$ and $\mathbf{\Phi} = [\phi, \theta, \psi]^\top$ the roll, pitch and yaw Euler angles of the UAV respectively.

The EKF's setup follows [27], with the IMU readings being used in order to perform prediction steps, while OF and VIO measurement updates are fused to improve state estimates. The final odometry output of the EKF is fed to RTAB at a set rate in order to enable localization in the current map as well as loop closures.

3.3 Path Planning & Obstacle Avoidance

The generated RTAB-Map is of 3D occupancy represented as octree voxels. Due to the regular geometry of artificial environments (i.e. indoor environment), the knowledge of obstacles around which the UAV is moving is sufficient for collision-free navigation. To that end, a tomographic 2D section of the 3D binary occupancy map is extracted to obtain a 2D binary occupancy map.

Since the UAV can change its altitude, it is crucial to extract the 2D occupancy map from its 3D counterpart around the UAV's altitude. Using the knowledge of the the altitude h in the map frame and height a of the UAV, the voxels in range $a \pm h/2$ are projected in z-axis to obtain a 2D representation of the map around the UAV altitude. For a safer flight in the face of altitude uncertainty, a larger range can be used.

Let the voxel center coordinates in 3D and 2D occupancy maps be

$$\mathbf{p}_{xyz} = [p_x, p_y, p_z]^\top \in \mathbb{Z}^3 \quad ; \quad \mathbf{q}_{xy} = [q_x, q_y]^\top \in \mathbb{Z}^2$$

respectively. These coordinates relate to the real coordinates if they are multiplied by the voxel grid size v. Furthermore, let $\mathcal{P}$ denote the set of $\mathbf{p}_{xyz}$ whose z-coordinate is within the range $a \pm h/2$:

$$\mathcal{P} = \left\{ \mathbf{p}_{xyz} \ : \ |p_z \cdot v - a| \leq \frac{h}{2} \right\} \tag{1}$$

The occupancy of 2D coordinate $\mathbf{q}_{xy} = [p_x, p_y]^\top$, denoted as $Oc(\mathbf{q}_{xy}) : \mathbb{Z}^2 \rightarrow \{0, 1\}$, is determined using

$$Oc(\mathbf{q}_{xy}) = \begin{cases} 1, & \exists\, \mathbf{p}_{xyz} \in \mathcal{P} \mid Oc(\mathbf{p}_{xyz}) = 1 \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

where $p_x = q_x$ and $p_y = q_y$. The proposed projection places higher priority to occupied cells, as the 2D coordinate is considered occupied even if a single cell in the column of voxels at (p_x, p_y) is occupied.

The definition of free 2D coordinates follows a similar construction. $Fr(\mathbf{q}_{xy}) : \mathbb{Z}^2 \rightarrow \{0, 1\}$ denotes the free cell function and is computed as

$$Fr(\mathbf{q}_{xy}) = \begin{cases} 1, & \forall\, \mathbf{p}_{xyz} \in \mathcal{P} \mid Fr(\mathbf{p}_{xyz}) = 1 \\ 0, & \text{otherwise} \end{cases}$$

The binary 2D occupancy is further processed to provide safety buffers against collision. Since each pixel in the 2D map corresponds to a real area, the occupied cells are expanded by the corresponding dimension of the UAV diameter plus a safety margin in map pixels to obtain **untraversable** region. The set of **traversable** points in the map are comprised of all free cells that are not in the untraversable set.

The traversable region need not be connected. However, the traversable region that is important for the UAV navigation is the region that contains the UAV, which is selected by performing a point test on the polygon defined by the contours of traversable regions. The term 'traversable' region will refer to the region containing the UAV.

A **frontier** (exit) point is a point in the contour of traversable region that is not neighboring a point in the untraversable set. As such, the frontiers are adjacent to the unknown space that needs to be explored.

Figure 2 provides a visualization that demonstrates the aforementioned terminology.

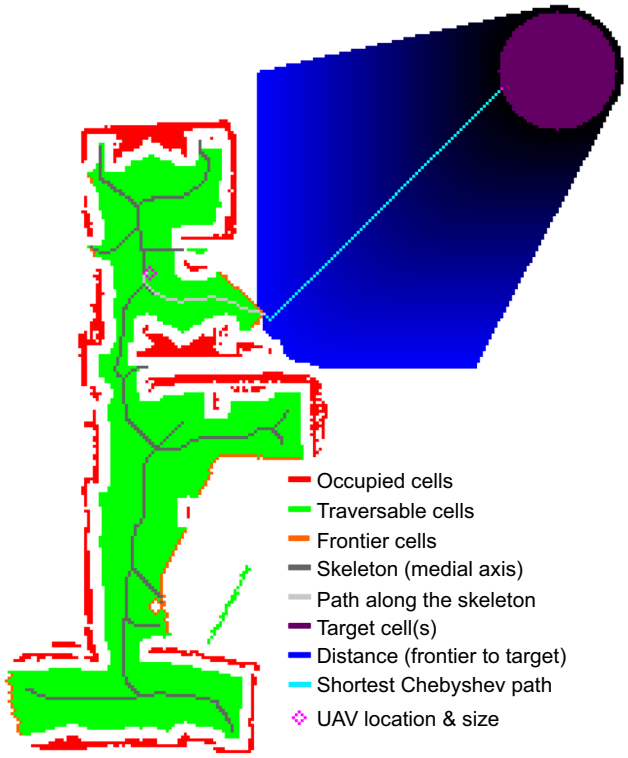

Fig. 2 Resultant path, comprised of a safe path inside the explored (traversable) region (light gray) and shortest Chebyshev path (cyan) between exit points (orange) and target coordinate(s) (purple). Only the path inside the traversable region is passed to the controller

3.3.1 Path Planning

Path planning is comprised of two parts. First is the path between the frontier points to the target location in the unknown space. This path exists only if there are no target point(s) inside the traversable region. The frontier point closest to the target point(s) is selected as the navigation target. The second part of the path is the path within the traversable region between the UAV location and the selected frontier point.

Path from Frontier to Target If there are no target points inside the traversable region, the system needs to know from which exit point it should leave the traversable region and explore more of the map. To that end, a fast wave-propagation algorithm is employed [26]. Any point that is not in the untraversable region is considered a valid coordinate that the UAV can occupy and is added as a valid neighbor during the propagation. The propagation is initiated from the target point(s) and continued until one of the frontier points is reached.

In order to reduce the computational overload, the propagation is contained within the convex hull of the explored space and the target coordinate(s).

Since the computed path is entirely in the unexplored regions of the map, it is not safe to navigate through the generated path. Instead, the frontier identified using the shortest Chebyshev distance is used as a greedy way of choosing an exit point from the frontiers of the traversable region.

Path from UAV to Frontier / Target Given that there are no target point(s) inside the traversable region, the exit point identified as outlined before is set as the navigation goal. Algorithms like A^* would provide a distance-optimal path that bends around the contours of traversable region. However, due to measurement errors in sensors, imprecise localization information, or controller tracking errors, navigation around the contours may not be collision-free. To minimize the chances of collision with the environment or the obstacles, a safer path along the medial axis of the traversable region is selected.

Medial axis, or skeleton, of a closed region is defined as the set of points inside the region that have more than one closest point on a region's boundary [6]. As such, it provides a viable candidate to maximize the distance from the boundaries of the environment.

The path from UAV to the exit point is broken down into 3 segments: a linear line-of-sight path from UAV to the closest

skeleton point $\mathbf{q}_s^{UAV}$, a linear path from the closest skeleton point with a line-of-sight to the exit point $\mathbf{q}_s^e$, and a path along the skeleton that connects $\mathbf{q}_s^{UAV}$ to $\mathbf{q}_s^e$.

In the cases that a target point falls inside the traversable region, the path computation is similar. Among the target points that fall inside the traversable region, the closest one to one of the skeleton points is selected as the navigation target, and the path computation is performed between $\mathbf{q}_s^{UAV}$ to that targeted point.

A demonstration of the generated path using the proposed method is given in Fig. 2. 2D shortest Chebyshev-path to the set of target points is denoted with cyan pixels, consisting of only perpendicular or ±45° inclined lines. The 'blue-colored' gradient is used to indicate the distance from the target set denoted in purple on the top right. The algorithm computes the wave-propagation in the convex hull between the explored space and the target set. The green pixels indicate traversable pixels and the red ones correspond to the slice of the discovered obstacles in the environment.

The skeleton in the 2D-discovered interior is shown with gray color and the current UAV position is shown with a magenta circle, sized according to the UAV dimensions. The drone is commanded to move along the skeleton shown in the light gray path.

3.4 Potential Fields for Obstacle Avoidance

Since the skeleton is computed on the discovered RTAB-Map, it is subject to inaccuracies, due to drift, slow updates or erroneous sensor readings. Thus for example, an area could be assumed free when in fact there are obstacles present, that have not yet been accurately incorporated in the map. In order to avoid such collisions, a local action is added to the path planning, constituting a force field emanating from nearby obstacles as seen on the depth image.

At every depth camera reading, the range value d of each pixel is used to compute the virtual force magnitude F_v as:

$$F_v = \max\left(0, -\log_{10}\left(\frac{d}{c_o}\right)\cdot\sigma\right),$$

where c_o, σ represent a cut-off value and a scale value.

Next, given the distance of each pixel from the camera, expressed at the camera's frame as D_x, D_z, then the virtual force is projected on each individual axis of the camera frame as

$$F_x = \max_{F_v}\left|F_v\frac{D_x}{R}\right|, \quad F_z = \max_{F_v}\left|F_v\frac{D_z}{R}\right|. \tag{3}$$

Finally forces F_x, F_z are transformed to the IMU frame with a fixed rotation into potential field force vector $\mathbf{F}_o$. Force vector $\mathbf{F}_o$ is then applied on the velocity command of the UAV, as seen in Section 4, altering the given path such that unintended collisions will be avoided.

4 Aerial Vehicle Control

The controller of the aerial vehicle is separated in three distinct modules (a) Position control, (b) Velocity Control, (c) Attitude Control.

The position controller is active during navigation stages and provides the desired velocity in order for the UAV to follow the path computed in world(map) coordinates. Let $\mathbf{P} \in \mathbb{R}^4$ the latest measured pose of the UAV in the map, described as $\mathbf{P} = [x, y, z, \psi]^\top$ and $\mathbf{P}^d$ the next desired pose in the latest computed path. Also, let $\mathbf{F}_o$ the vector of potential field forces expressed in the UAV frame u. Then the position controller action is

$$\mathbf{V}^d = -\mathbf{H}(\psi)\cdot\mathbf{K}\cdot\left(\mathbf{P}-\mathbf{P}^d\right)+\mathbf{F}_o, \tag{4}$$

where $\mathbf{K}$ a diagonal positive gain matrix and

$$\mathbf{H}(\psi) = \begin{bmatrix} R_{z,\psi}^\top & 0_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix}. \tag{5}$$

Matrix $R_{z,\psi}$ in (5) represents the elementary rotation matrix $R \in SO(3)$ computed about axis z by an angle ψ. The vector $\mathbf{V}^d \in \mathbb{R}^4$ from (4) holds the body-frame (u) velocity commands transmitted to the velocity controller.

The velocity controller accepts both desired velocity and direct acceleration commands. As seen in later sections, direct acceleration commands are directly transformed into roll, pitch angles and total thrust force, effectively bypassing the velocity controller, which is only used during the short-lived ballistic trajectory execution phase.

Let the gravity vector $\mathbf{g} = \begin{bmatrix}0 & 0 & 9.81\end{bmatrix}^\top$, the rotation matrix ${}^wR_u = R_{x,\phi}R_{y,\theta}R_{z,\psi} \in SO(3)$ and subsequently the linear velocity vector expressed in the body frame u as $v_b = {}^wR_u\cdot v$. When a desired velocity vector $\mathbf{V}^d = \begin{bmatrix} v_b^d \\ \omega_z^d \end{bmatrix}$ is commanded to the velocity controller, a desired acceleration vector is computed as

$$\alpha^d = \mathbf{g} - \mathbf{K}_p\left(v_b - v_b^d\right) - \mathbf{K}_I\int\left(v_b - v_b^d\right),$$

where $\mathbf{K}_p, \mathbf{K}_I$ diagonal positive gain matrices.

These desired linear accelerations $\alpha^d = [\alpha_x^d, \alpha_y^d, \alpha_z^d]^\top$ are subsequently transformed into roll, pitch (ϕ, θ) and thrust(T)

setpoints for the attitude controller based on (6)–(7).

$$T = \frac{m\,\alpha_z}{\cos(\phi)\cos(\theta)}, \tag{6}$$

$$\phi^d = -\frac{m\,\alpha_x}{T}, \quad \theta^d = \frac{m\,\alpha_y}{T}, \tag{7}$$

where m is the mass of the UAV.

The attitude controller accepts roll, pitch, yaw rate and thrust $(\phi^d, \theta^d, \omega_z^d, T)$ setpoints. Desired angular velocity terms are generated for the vehicle's x, y axes as: $\omega_x^d = -K_\phi\left(\phi - \phi^d\right)$, $\omega_y^d = -K_\theta\left(\theta - \theta^d\right)$, resulting in desired angular velocity vector $\Omega^d = \left[\omega_x^d, \omega_y^d, \omega_z^d\right]^\top$. This vector can be compared with the angular velocity vector Ω measured by the gyroscopes, facilitating angular velocity error vector $\mathbf{e}_\omega \triangleq \Omega - \Omega^d$.

A PID controller is used to compute attitude control moments $\boldsymbol{\tau} \in \mathbb{R}^3$ in order to regulate the angular velocity error vector to zero

$$\boldsymbol{\tau} = -K_\omega \mathbf{e}_\omega - K_{\dot{\omega}}\hat{\dot{\mathbf{e}}}_\omega - K_{\omega I}\int \mathbf{e}_\omega,$$

where the $\hat{\dot{\mathbf{e}}}_\omega$ term is estimated by numerical differentiation followed by low-pass filtering.

Finally, the individual motor commands are generated using $(T, \boldsymbol{\tau})$ given the aerial vehicle's mixing matrix [1]. The most prevalent model [7] for propeller thrust force f and moment M, as a result of motor rotational velocity ω is $f = c_f\omega^2$, $M = c_m\omega^2$, where c_f, c_m are positive constants. For a quadrotor UAV with given c_f, c_m parameters and arm half-length l, the mixing matrix can be written as

$$A = \begin{bmatrix} c_f & c_f & c_f & c_f \\ -lc_f & lc_f & lc_f & -lc_f \\ -lc_f & -lc_f & lc_f & lc_f \\ c_m & -c_m & c_m & -c_m \end{bmatrix}.$$

Thus finally motor speed commands $\omega^m \in \mathbb{R}^4$ can be computed as

$$\omega^m = A^{-1}\begin{bmatrix} T \\ \boldsymbol{\tau} \end{bmatrix}.$$

Motor speed commands ω^m are finally transmitted to the electronic speed controllers.

A diagram of system-component interaction is provided in Fig. 3.

5 Target Representation, Acquisition & Ballistic Payload Delivery

5.1 Target Representation

Due to the uncertainty pertaining to the exact location of the target, this is instead represented by an arbitrarily large target region, comprised of points in the map where the target is expected to reside. In theory, a target region encompassing the entire environment, can reduce the problem to that of pure exploration.

The implemented system keeps track of all of the target points and removes the coordinates from the set of targets, based on a confidence that the delivery target is not located inside the given location. To that end, a voxel-based representation with the same resolution as the 3D occupancy map is used to denote the target points.

Let $\mathcal{D}$ the set of points comprising the target region, expressed in the map frame. For any location p_t in $\mathcal{D}$, there exists a corresponding probability $\Pi(p_t) \in [0, 1]$ representing the likelihood of p_t not containing the target.

While navigating, the agent needs to discern which of the elements of $\mathcal{D}$ are within its field of perception and increase the probability value for all such elements only, depending on how clear the view of the location is. Any location with probability exceeding a certain threshold, is removed from

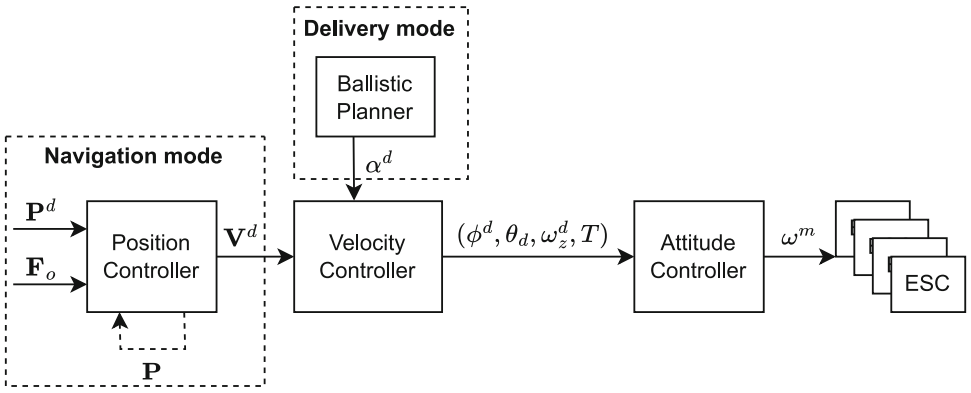

Fig. 3 Illustration of the interaction between different controller components

$\mathcal{D}$ as it is assumed to have been properly observed and found not to contain the target.

The perceptive field of the UAV can be most clearly defined in the body-fixed frame u. Thus, all candidate target locations are transformed to the u frame before discerning their observability. Let ψ the UAV heading angle and p its position in map, as specified in Section 3. Then every location $p_t \in \mathcal{D}$ can be transformed to the u frame by

$$p_t^u \triangleq \left[p_{tx}^u, p_{ty}^u, p_{tz}^u\right]^\top = R_{z,\psi}^\top (p_t - p)\,.$$

Also let F_f the camera Field of View and r_f the perceptive range of the camera. Then the set of all target locations which currently reside within the field of view of the camera, say $\mathcal{P}$, can be found as

$$\mathcal{P} = \left\{ p_t \in \mathcal{D} :\ |\text{atan2}(p_{ty}^u, p_{tx}^u)| \leq \frac{F_f}{2}\ ,\ ||p_t^u|| \leq r_f \right\}.$$

Every location p_v in $\mathcal{P}$ has successfully passed the observability check, then it needs to be confirmed that there exists a line-of-sight (occlusion check) and that p_v does not represent an occupied voxel (reachability check).

The occlusion check is performed by tying target region updates to incoming depth camera measurements. Let T_m^c the transformation matrix between map and camera frame and K_c the camera projection matrix obtained by camera calibration. Then, $\forall p_v \in \mathcal{P}$, the corresponding line-of-sight distance l_v is obtained by $l_v = [0, 0, 1]\, T_m^c p_v$, while the corresponding pixel coordinates for location p_v are reconstructed as:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = K_c T_m^c p_v, \quad \begin{bmatrix} x_p \\ y_p \end{bmatrix} = \frac{1}{Z} \begin{bmatrix} X \\ Y \end{bmatrix}.$$

Location p_v is only considered visible if the depth image I_D at pixel coordinates (x_p, y_p) has a value larger than l_v, indicating line-of-sight between camera and target location.

The reachability check is performed by cross-referencing each p_v location with the latest discovered map. Any point in $\mathcal{P}$ found to be representing a location determined to be occupied in the map, can be removed from the list of valid target points, as it is unreachable by the UAV.

Figure 4 shows an illustration of a case where part of the target region is visible by the UAV. The letters provide example cases for specific points of the region. Point A passes the observability check (lies within camera perceptive field), occlusion check (there exists line-of-sight) and reachability check (represents a valid position that the UAV can acquire). Point B passes the observability and may pass the occlusion check, if it lies close to the wall boundary, thus fails the reachability check. Subsequently, due to failing the reachability check Point B is also immediately removed from the list of points comprising the target region

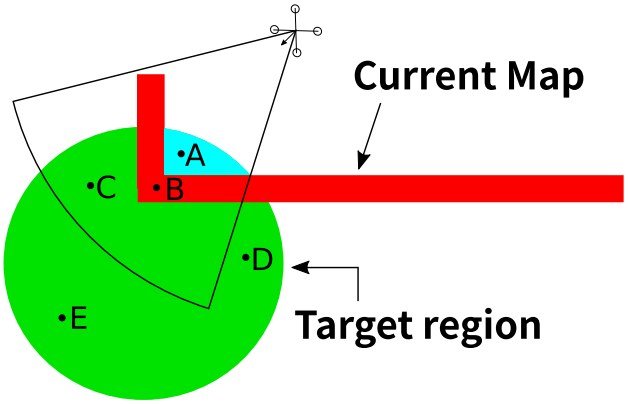

Fig. 4 Illustration of target region handling with example cases

On the other hand, Point C passes observability but fails the occlusion check. Points D and E fail the observability check by being too wide or too far respectively.

For all locations in $\mathcal{P}$ satisfying both the occlusion and reachability checks, the respective probability Π is updated based on a coverage metric taking into account distance and angle with respect to observed target location. The coverage metric f_c utilised here is simply

$$f_c = \frac{\left|\text{atan2}(p_{ty}^u, p_{tx}^u)\right|}{\frac{F_f}{2}} \cdot \frac{||p_t^u||}{r_f},$$

constituting a product of normalized view angle and distance, conforming f_c to lie in the range $[0, 1]$. A small f_c value means that at least one of the two coverage factors (view angle and distance) returned a high confidence value, while an f_c value close to 1, signifies low confidence for both coverage factors. Therefore, the following coverage confidence function was designed to handle the increment to be applied to probability value Π for observed location p_t^u:

$$\delta_\Pi = \begin{cases} 0.001 & \text{if } f_c \geq 0.75 \\ 0.2\frac{e^{(f_c)}}{1+e^{(f_c/2)}} & \text{else} \end{cases},$$

followed by $\Pi = \Pi + \delta_\Pi$.

Looking at Fig. 4, as the UAV navigates towards the points of the blue sub-region of the target region, these are also the points the probabilities of which are being updated based on their visibility. If the target does not exist in this sub-region, then the probabilities of said points will quickly exceed the threshold, removing them from the list of targets. This allows the UAV navigator to swift its focus towards navigating to the rest of the target region.

Handling the target points in this manner enables specifying arbitrary regions in the map frame as valid target locations. Without any prior knowledge, a disk (sphere in 3D) is a natural choice to account for drifting and measurement inaccuracies, but it is not necessary. The specified coordinates need not be connected, providing the user to specify multiple disjoint candidate locations in arbitrary shapes to guide the exploration.

5.2 Target Acquisition & Ballistic Delivery

The target of interest is represented by an artificial reality tag, specifically a marker from the ArUco library is used. The detection software is running quietly on the background during navigation, and activates after a number of successful marker detections.

Activation of the marker detection node signals the end of exploration phase and the beginning of target acquisition and delivery phase. The marker detection node performs the reconstruction of the marker pose and placement of the detected marker in the created map.

The UAV attempts to converge to a predefined desired relative pose with respect to the marker, based on which the new $\mathbf{P}^d$ poses are computed in the map frame. Upon convergence, a path for ballistically delivering the payload (fire-extinguishing ball) to the reconstructed target needs to be planned and executed, at the end of which the task is finally assumed to be completed.

Based on research performed in [28, 34, 35], it has been established that a ball performing ballistic motion, is affected by three main forces, namely: 1) gravitation pull, 2) air drag and 3) Coriolis forces due to spin. Since air drag forces and spin-induced forces require extensive knowledge of the properties of the ball and the environment in order to be predicted, along with the established fact of their effects being negligible compared to gravity, only the gravitational force is taken into account.

Moreover, the relative positioning of the UAV with respect to the target is such that the vehicle will face the target's plane. Thus, only motions along the UAV's x^u, z^u axes will be required to deliver the ball, reducing the problem to that of 2D planning for the ball's trajectory.

Ignoring all non-gravitational effects, the x, z position of the ball at $t = T$ seconds, with the release happening at $t = t^r$, can be predicted as

$$x(T) = v_x(t^r)\, T + x(t^r), \tag{8}$$

$$z(T) = -g\frac{T^2}{2} + v_z(t^r)\, T + z(t^r), \tag{9}$$

where $x(t^r), z(t^r)$ the x^u, z^u axis positions of the ball (and subsequently the UAV) at the point of release and $v_x(t^r), v_z(t^r)$ the UAV velocities at the point of release. Equation (9) show that for the ball to accurately hit a marker at positions x^m, z^m, release positions and velocities need to be computed, such that after a time span of $(T - t^r)$ seconds, the ball will hit the target.

The UAV needs to select an initial position from which to start its delivery, along with a precomputed trajectory (starting at $t = 0$) such that the UAV will reach the release positions $x(t^r), z(t^r)$ at time t^r with corresponding velocities $v_x(t^r), v_z(t^r)$ at which point the UAV will perform its release maneuver.

Let $\zeta = [x^u, v_x^u, z^u, v_z^u]^\top$ the 2D-state of the UAV in the frame aligned with the marker. Model predictive planning is an established method for planning robot behavior while taking into account constraints at the level of robot dynamics [16, 23]. Application of model predictive planning on UAVs is also found in [13]. Here, a model predictive planner is designed to solve the problem of computing an optimal trajectory for the UAV to deliver the ball, while respecting the UAVs dynamic constraints. The planner relies on the following simplified and discretized model of UAV position dynamics in x^u, z^u axes.

$$\zeta(t + dt) = A_d\zeta(t) + B_d\alpha(t) + G_d,$$

where

$$A_d = \begin{bmatrix} 1 & dt & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & dt \\ 0 & 0 & 0 & 1 \end{bmatrix},\ B_d = \begin{bmatrix} 0 & 0 \\ dt & 0 \\ 0 & 0 \\ 0 & dt \end{bmatrix},\ G_d = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -g\,dt \end{bmatrix}.$$

and $\alpha(t) = [\alpha_x(t), \alpha_z(t)]^\top$ is the vector of body-frame desired accelerations that the planner should select.

Inhere, the required time the UAV should 'consume' at each stage of the delivery was pre-selected. The control timestep was selected as $dt = 10$ ms, with the time to execute the release trajectory set to $t = 2$ s (or $N = 200$ steps) while the free flight time of the ball was $T = 0.9$ s.

Let Q_x, Q_z positive definite cost matrices, x_t, z_t the location of the target in the target-aligned frame, x_L, x_U the minimum and maximum x^u axis locations the UAV can acquire, extracted from the map and z_L, z_U the lower (floor) and upper (ceiling) altitudes the UAV can acquire. Let also α_M, v_M the maximum velocity and acceleration limits allowed for the UAV, taking the gravity acceleration into account for the altitude case.

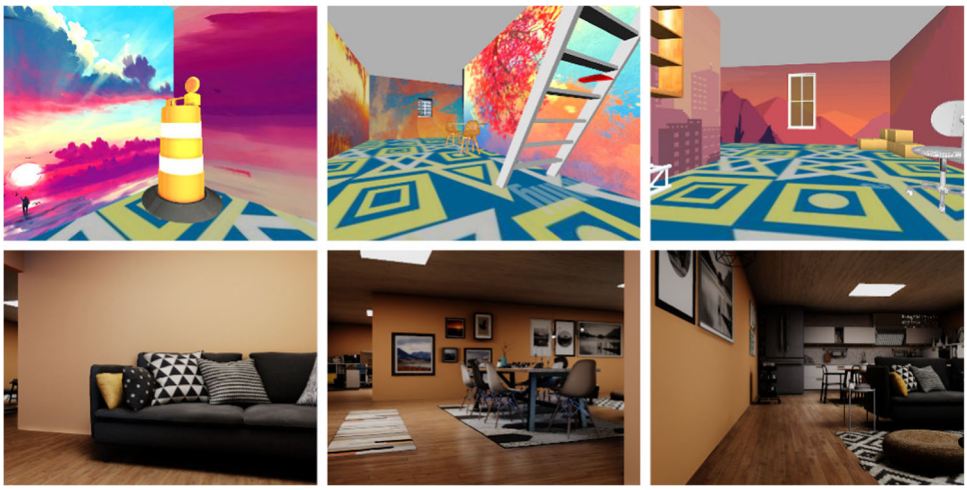

Fig. 5 Camera renderings from Gazebo (top row) and Unity (bottom row)

Then the model-predictive planner optimization routine can be dictated by the following set of equations.

$$\min_{\alpha_x(i),\alpha_z(i)\|i=0,\ldots,N} \alpha_x^\top Q_x \alpha_x + \alpha_z^\top Q_z \alpha_z - (x^u(N) - x_t)^2 - (z^u(N) - z_U)^2 \tag{10}$$

subject to:

$$\begin{aligned}
\zeta(i) &\triangleq [x^u(i), v_x^u(i), z^u(i), v_z^u(i)]^\top, \\
\zeta(i+1) &= A_d\zeta(i) + B_d\alpha(i) + G_d, \\
[x_L, -v_M, z_L, -v_M]^\top &\leq \zeta(i) \leq [x_U, v_M, z_U, v_M]^\top, \\
-\alpha_M &\leq \alpha_x(i) \leq \alpha_M, \\
-\alpha_M &\leq \alpha_z(i) - g \leq \alpha_M, \\
v_x^u(N)\, T + x^u(N) &= x^t, \\
-g\frac{T^2}{2} + v_z^u(N)\, T + z^u(N) &= z^t.
\end{aligned}$$

Fig. 6 Overhead visualization of the environment used in Gazebo-Unity simulation

The first set of constraints encompass the simplified motion dynamics and ensure the vehicle's state will not exceed position and velocity safety limits, while also enforcing limits for the acceleration the UAV can achieve. The last set of constraints guarantee that the final state of the UAV at the point of release will be such that the ball will hit its target.

The optimization problem is designed such that the UAV optimizes its accelerations while selecting a release trajectory that will keep it as far as possible from the surrounding walls. The model-predictive planner of Equation (10) is invoked once, and the resulting trajectory of UAV states and accelerations is returned to the navigation commander. The CVXOPT package [4] was used to form and solve the convex optimization problem.

6 Simulation Studies

6.1 Simulation Parameters

The proposed system is tested for an indoor navigation, target finding, and delivery scenario using a state of the art simulator setup. The software setup uses the Open Dynamics Engine (ODE) physics engine within Gazebo which is used for solving the dynamics, while the Unity game engine is utilized to render high fidelity visual data.

Gazebo [14] is a commonly adopted simulation engine that has integration with ROS and provides many options to modify the simulation environment for a particular scenario. Even though Gazebo provides flexibility in the simulation of dynamics, the visual fidelity of the images obtained from

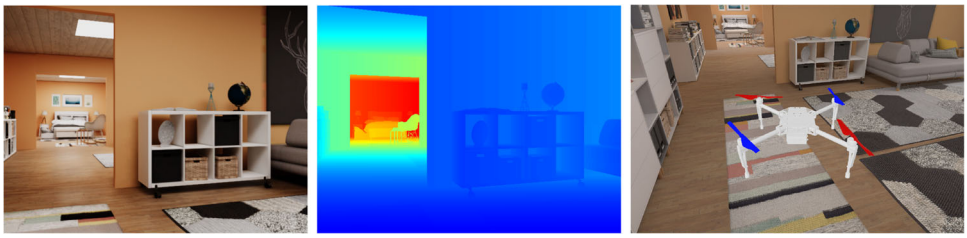

Fig. 7 Visual (left), depth (center) images and a user interface snapshot (right)

camera sensors is not realistic due to the missing ray casting from the various light sources.

The Unity game engine is well suited for realistic renderings from cameras, with built in tools for lightmap generation, shadow calculation, camera intrinsics adjustments, and post-processing effects. For comparison of the environments [22], Fig. 5 provides some sample camera renderings from Gazebo and Unity.

Unity provides an interface to communicate with a ROS server to publish data obtained from the simulation environment, and allows for parsing URDF files. Although, the entire simulation could take place in Unity, its integrated PhysX engine optimizes physics for speed rather than fidelity (e.g. Coriolis forces are ignored [9]), making it unsuitable for accurate dynamics simulations [19]. However, Unity is capable of rendering realistic-looking scenes in real time, making it useful especially for scenarios where visual feedback is used.

To utilize Unity for visual simulation, a depth camera (Intel D435i), an optical flow camera (PX4 Flow sensor), and a laser ranger for altitude is simulated and their data is communicated to ROS for use.

The physics of the UAV, as well as IMU sensor, are simulated using Gazebo [10].

A rendering of the environment used in Gazebo-Unity simulation is provided in Fig. 6, with the UAV's starting position and target location marked on the map with red and magenta circles, respectively, while the openings that can be used by the UAV to enter various rooms are marked in cyan. Similarly a snapshot of the visual as seen by the camera, the depth and the user interface that has included the UAV are shown in Fig. 7 at the left, center and right portions respectively.

The depth camera operates at 10 Hz while the optical flow and range sensor is operating at 100 Hz. We should point that the majority of the simulation time is due to the data transfer between the CPU and the needed GPU (utilized by Unity). The overall simulation time including controller software, path planning algorithm, and SLAM, was reduced at 25 ms (40 Hz) on an Intel NUC Enthusiast PC (Intel i7-1165G7, Nvidia GeForce RTX 2060). All the simulations as well as the software for the framework runs in real time.

6.2 Simulation Results

The aforementioned simulation environment is used to test two scenarios where: i) a single connected region is provided

Fig. 8 Left: Ground-truth path of the UAV (green) and target region (purple) overlaid on top of the simulation environment overhead view. Right: results of the path finding algorithm on the 2D tomographic section of the generated 3D occupancy map

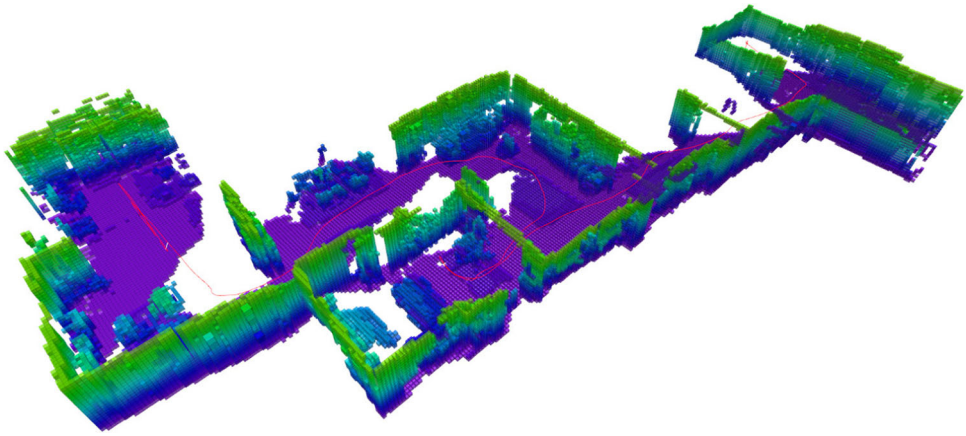

Fig. 9 Resultant 3D occupancy map from RTAB-Map for the single target region scenario

to guide the navigation discovery, and ii) multiple disjoint target regions are defined.

6.2.1 Single Target Region Scenario

In this scenario, the target region is specified as a single disk at the far end of the environment. Figure 8 provides the depictions of ground truth path of the UAV during its navigation in the navigation environment, and the tomographic section of the imperfect 3D occupancy map at the end of the task. The specified target region is highlighted in purple., while the medial axis of the traversable region is shown in gray, and the followed UAV path mimics the appearance of the medial axis. Despite the large drift in the mapping, the proposed system guides the UAV to the room with the target.

A depiction of the 3D occupancy map, along with the trajectory (red-colored) of the UAV in the 3D space is provided in Fig. 9.

Figure 10 shows the result of the UAV's attempt to ballistically hit the target with a ball. The arrows indicate the orientation of the UAV body-fixed frame's z axis, which shows how the UAV's release maneuver attempts a swift change in pitch, before actually releasing the ball. This helps the ball achieve a longer ballistic trajectory, as the angular velocity of the maneuver increases.

The included timestamps allow to keep track of the performed actions, where 1.76 s were needed by the UAV to reach the pre-planned release position, at which point the UAV releases the ball and performs arecovery maneuver. The UAV needed $2.31 - 1.76 = 0.55$ s to 'fully recover' from the large forward velocity it had acquired. This underlines the importance of the ballistic trajectory, as it allows the UAV to hit the target while maintaining a minimum allowed distance from it. In the case where the target is a fire, this minimum distance is crucial for the safety of the vehicle.

6.2.2 Multiple Disjoint Target Regions Scenario

For the multiple target scenario, three disjoint circular regions are provided as the locations that the UAV can expect to find the target marker. Figure 11 provides the ground truth UAV path and the specified target regions, as well as the resultant tomographic section of the occupancy map. The same coloring was adopted as in the single-target scenario, and overall a larger portion of the environment was explored to find the target marker.

A depiction of the 3D occupancy map, along with the trajectory of the UAV in the 3D space is provided in Fig. 12.

The evolution of the tomographic 2D occupancy map with traversable and occupied regions, medial axis, target regions and the distance from target to frontiers is provided in Fig. 13.

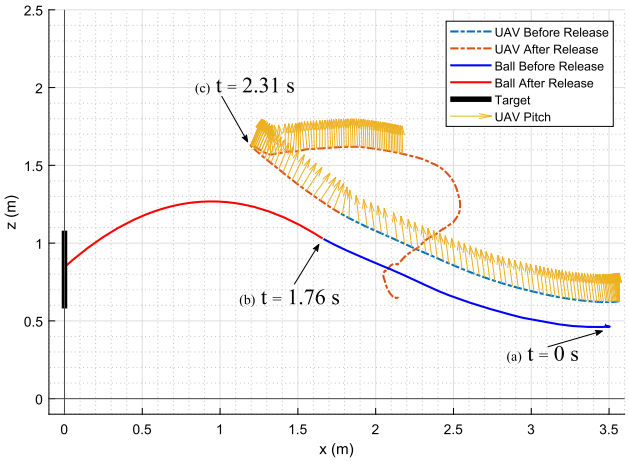

Fig. 10 Ballistic trajectory of ball and corresponding UAV motion with timestamps

Fig. 11 Left: Ground-truth path of the UAV (green) and target region (purple), overlaid on top of the simulation overhead view. Right: results of the path finding algorithm on the 2D tomographic section of the generated 3D occupancy map

Note that the number on the top left corner of each frame depicts the map's ascending order in the sequence.

Since the selection of the navigation target is done in a greedy manner, the region explored first is the one closest to the UAV's starting coordinate. In Frames 2 and 3 of Fig. 13, the first region is partially explored. Between the Frames 2 and 3, the evolution of the medial axis is based on the amount of exploration that can be seen. Once the closest region is fully explored, the algorithm switches to the next closest target point (Frame 4). Finally in Frames 7 and 8, the UAV is navigating to approach the final set of target coordinates. Even though the initially provided coordinates are outside the mapped region due to map drift, the UAV is able to navigate towards the target and deliver the payload.

7 Conclusions

The computation of a hybrid path for a UAV flying in an unknown environment with the goal of detecting and eliminating a fire, is considered in this article. A safe trajectory segment for the explored space, followed by a shortest path in the Chebyshev-sense for the undiscovered space, is considered. The UAV is localized using the RTAB-Map SLAM and on-board sensors. Prior vague knowledge of target locations can be provided to the UAV to shorten exploration time. Upon target detection, the UAV is capable of executing a motion such that a fire-extinguishing ball can be released and ballistically hit the target, while maintaining a certain distance between the UAV and emulated fire.

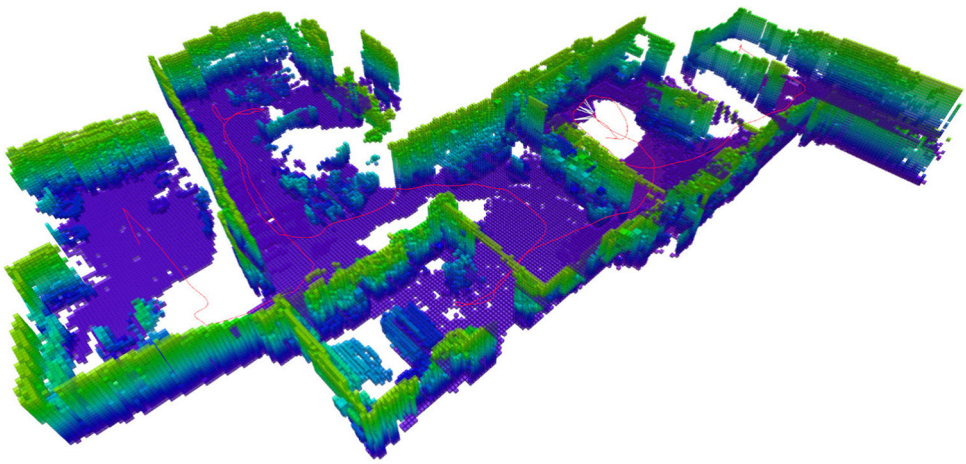

Fig. 12 Resultant 3D occupancy map from RTAB-Map for the multiple disjoint target regions scenario

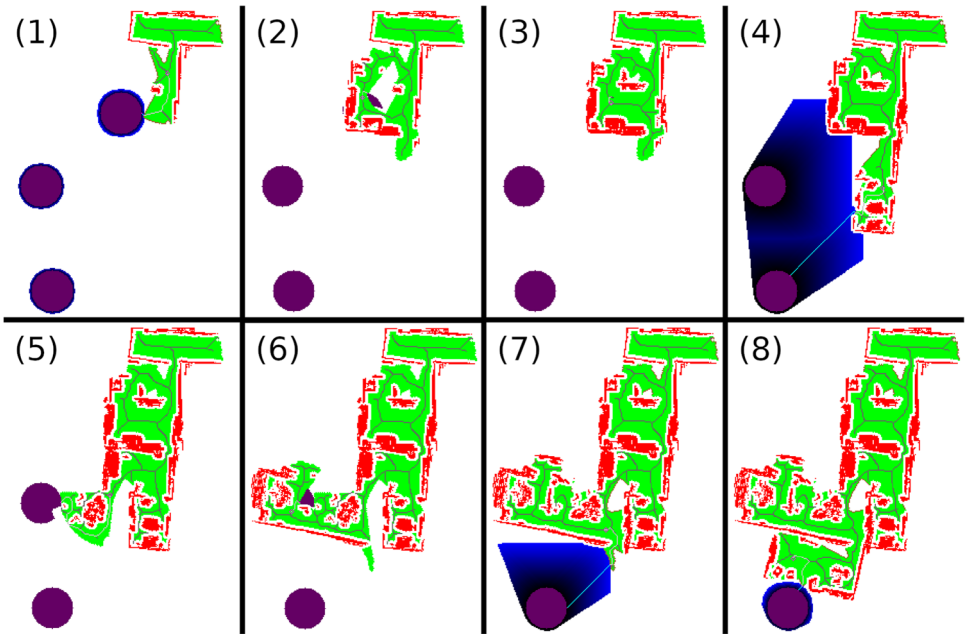

Fig. 13 Evolution of the tomographic occupancy map as the UAV is exploring the environment guided by the target regions

Author Contributions The following paragraph denotes how all authors contributed to this work. Conceptualization, ATz, HU, DC; initial development, HU, DC, ATz; simulator software, HU, DC, ATs; SLAM, HU; path planning, HU, ATs; UAV controls, DC; ballistic delivery, DC; writing-original draft, HU, DC, ATz; writing-review and editing, ATz; visualizations, HU, DC; supervision, ATz; funding acquisition ATz. HU and DC contributed equally to this work. All authors read and approved the final manuscript.

Funding This work is supported in part by the NYUAD Center for Artificial Intelligence and Robotics (CAIR), funded by Tamkeen under the NYUAD Research Institute Award CG010.

Declarations

Ethics Approval No ethical approval is required for publishing this research.

Competing Interest The authors have no relevant financial or non-financial interests to disclose.

References

1. Achtelik, M., Doth, K.M., Gurdan, D., Stumpf, J.: Design of a multi rotor MAV with regard to efficiency, dynamics and redundancy. In: AIAA Guidance, Navigation, and Control Conference, p. 4779 (2012)
2. Aggarwal, S., Kumar, N.: Path planning techniques for Unmanned Aerial Vehicles: A review, solutions, and challenges. Computer Communications **149**, 270–299 (2020)
3. Almeida, T., Santos, V., Mozos, O.M., Lourenço, B.: Comparative analysis of deep neural networks for the detection and decoding of data matrix landmarks in cluttered indoor environments. Journal of Intelligent & Robotic Systems **103**(1), 1–14 (2021)
4. Andersen, M.S., Dahl, J., Vandenberghe, L., et al.: CVXOPT: A Python package for convex optimization. abel. ee. ucla. edu/cvxopt **88** (2013)
5. Aqel, M.O., Marhaban, M.H., Saripan, M.I., Ismail, N.B.: Review of visual odometry: types, approaches, challenges, and applications. SpringerPlus **5**(1), 1–26 (2016)
6. Blum, H.: A transformation for extracting new descriptions of shape. Models for the perception of speech and visual form pp. 362–380 (1967)
7. Bouabdallah, S., Siegwart, R.: Backstepping and sliding-mode techniques applied to an indoor micro quadrotor. In: Proceedings of the 2005 IEEE International Conference on Robotics and Automation, pp. 2247–2252. IEEE (2005)
8. Delamer, J.A., Watanabe, Y., Chanel, C.P.: Safe path planning for UAV urban operation under GNSS signal occlusion risk. Robotics and Autonomous Systems **142**, 103800 (2021)
9. Erez, T., Tassa, Y., Todorov, E.: Simulation tools for model-based robotics: Comparison of Bullet, Havok, MuJoCo, ODE and PhysX. In: 2015 IEEE International Conference on Robotics and Automation (ICRA) (2015)
10. Furrer, F., Burri, M., Achtelik, M., Siegwart, R.: Rotors-a modular Gazebo MAV simulator framework. In: Robot operating system (ROS), pp. 595–625. Springer, Cham (2016)
11. Han, F., Wang, H., Huang, G., Zhang, H.: Sequence-based sparse optimization methods for long-term loop closure detection in visual SLAM. Autonomous Robots **42**(7), 1323–1335 (2018)
12. Hornung, A., Wurm, K.M., Bennewitz, M., Stachniss, C., Burgard, W.: OctoMap: An efficient probabilistic 3D mapping framework based on octrees. Autonomous robots **34**(3), 189–206 (2013)
13. Ivanovic, A., Car, M., Orsag, M., Bogdan, S.: Exploiting null space in aerial manipulation through model-in-the-loop motion planning. In: 2020 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 686–693. IEEE (2020)
14. Koenig, N., Howard, A.: Design and use paradigms for Gazebo, an open-source multi-robot simulator. In: 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566), vol. 3, pp. 2149–2154. IEEE
15. Labbé, M., Michaud, F.: RTAB-Map as an open-source LiDAR and visual simultaneous localization and mapping library for large-scale and long-term online operation. Journal of Field Robotics **36**(2), 416–446 (2019)

16. Liu, C., Lee, S., Varnhagen, S., Tseng, H.E.: Path planning for autonomous vehicles using model predictive control. In: 2017 IEEE Intelligent Vehicles Symposium (IV), pp. 174–179. IEEE (2017)
17. Maini, P., Sujit, P.: Path planning for a UAV with kinematic constraints in the presence of polygonal obstacles. In: 2016 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 62–67. IEEE (2016)
18. Matos-Carvalho, J.P., Santos, R., Tomic, S., Beko, M.: GTRS-based algorithm for UAV navigation in indoor environments employing range measurements and odometry. IEEE Access **9**, 89120–89132 (2021)
19. de Melo, M.S.P., da Silva Neto, J.G., da Silva, P.J.L., Teixeira, J.M.X.N., Teichrieb, V.: Analysis and Comparison of Robotics 3D Simulators. In: 2019 21st Symposium on Virtual and Augmented Reality (SVR) (2019)
20. Nistér, D., Naroditsky, O., Bergen, J.: Visual odometry. In: Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004., vol. 1, pp. I–I. Ieee (2004)
21. Padhy, R.P., Verma, S., Ahmad, S., Choudhury, S.K., Sa, P.K.: Deep neural network for autonomous UAV navigation in indoor corridor environments. Procedia Comput. Sci. **133**, 643–650 (2018)
22. Platt, J.T.: Comparative Analysis of Ros-Unity3D and Ros-Gazebo for Mobile Ground Robot Simulation. Ph.D. thesis, University of Alabama Libraries (2022)
23. Rasekhipour, Y., Khajepour, A., Chen, S.K., Litkouhi, B.: A potential field-based model predictive path-planning controller for autonomous road vehicles. IEEE Trans. Intell. Trans. Syst. **18**(5), 1255–1267 (2016)
24. Santamaria-Navarro, A., Loianno, G., Sola, J., Kumar, V., Andrade-Cetto, J.: Autonomous navigation of micro aerial vehicles using high-rate and low-cost sensors. Autonomous robots **42**(6), 1263–1280 (2018)
25. Smith, R., et al.: Open dynamics engine (2007)
26. Sniedovich, M.: Dijkstra's algorithm revisited: the dynamic programming connexion. Control Cybern. **35**(3), 599–620 (2006)
27. Svacha, J., Mohta, K., Watterson, M., Loianno, G., Kumar, V.: Inertial velocity and attitude estimation for quadrotors. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1–9. IEEE (2018)
28. Tebbe, J., Gao, Y., Sastre-Rienietz, M., Zell, A.: A table tennis robot system using an industrial Kuka robot arm. In: German Conference on Pattern Recognition, pp. 33–45. Springer (2018)
29. Technologies, U.: Unity real-time development platform (2022). https://unity.com/
30. Unlu, H.U., Chaikalis, D., Tsoukalas, A., Tzes, A.: UAV-Navigation using Map Slicing and Safe Path Computation. In: 2023 9th International Conference on Automation, Robotics and Applications (ICARA) pp. 208-212. IEEE (2023)
31. Walker, O., Vanegas, F., Gonzalez, F., Koenig, S.: A deep reinforcement learning framework for UAV navigation in indoor environments. In: 2019 IEEE Aerospace Conference, pp. 1–14. IEEE (2019)
32. Wang, F., Wang, K., Lai, S., Phang, S.K., Chen, B.M., Lee, T.H.: An efficient UAV navigation solution for confined but partially known indoor environments. In: 11th IEEE International Conference on Control & Automation (ICCA), pp. 1351–1356. IEEE (2014)
33. Yang, L., Qi, J., Song, D., Xiao, J., Han, J., Xia, Y.: Survey of Robot 3D Path Planning Algorithms. Journal of Control Science and Engineering **2016**, 1–22 (2016)
34. Zhang, K., Cao, Z., Liu, J., Fang, Z., Tan, M.: Real-time visual measurement with opponent hitting behavior for table tennis robot. IEEE Transactions on Instrumentation and Measurement **67**(4), 811–820 (2018)
35. Zhang, Z., Xu, D., Tan, M.: Visual measurement and prediction of ball trajectory for table tennis robot. IEEE Transactions on Instrumentation and Measurement **59**(12), 3195–3205 (2010)
36. Zhao, W., Queralta, J.P., Westerlund, T.: Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In: 2020 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 737–744. IEEE (2020)

Halil Utku Unlu is a PhD candidate at New York University, with a Global PhD Fellowship from NYU Abu Dhabi as a member of the Robotics & Intelligent Systems Control Laboratory. His research focuses on 3D robot perception, collaborative robotics, and robot autonomy.

Dimitris Chaikalis is a PhD candidate at New York University and a graduate assistant at the Robotics and Intelligent Systems Control Lab of NYU Abu Dhabi. His research focuses on aerial vehicle control, autonomous transportation and cooperative control.

Athanasios Tsoukalas is a Research Engineer at New York University Abu Dhabi. His research focuses on UAV Tracking, GPU based algorithm optimization, autonomous transportation and city wide base station distribution and path planning.

Professor Anthony Tzes is the Head of the Electrical Engineering program at NYU Abu Dhabi. Concurrently he is the Lead Investigator of the NYUAD's Center for AI and Robotics. His research interests include distributed collaborative control of autonomous mobile agents.