Unmanned Systems, Vol. 10, No. 2 (2022) 175–186

DOI: 10.1142/S2301385022500108

UNMANNED SYSTEMS
https://www.worldscientific.com/worldscinet/us

FAST RRT* 3D-Sliced Planner for Autonomous Exploration Using MAVs

Á. Martínez Novo*, Liang Lu†, Pascual Campoy

Computer Vision and Aerial Robotics (CVAR) Group,
Centre for Automation and Robotics (CAR),
Universidad Politécnica de Madrid (UPM-CSIC)
Calle José Gutiérrez Abascal 2, 28006 Madrid, Spain

This paper addresses the challenge to build an autonomous exploration system using Micro-Aerial Vehicles (MAVs). MAVs are capable of flying autonomously, generating collision-free paths to navigate in unknown areas and also reconstructing the environment at which they are deployed. One of the contributions of our system is the "*3D-Sliced Planner*" for exploration. The main innovation is the low computational resources needed. This is because Optimal-Frontier-Points (OFP) to explore are computed in 2D slices of the 3D environment using a global Rapidly-exploring Random Tree (RRT) frontier detector. Then, the MAV can plan path routes to these points to explore the surroundings with our new proposed local "*FAST RRT* Planner*" that uses a tree reconnection algorithm based on cost, and a collision checking algorithm based on Signed Distance Field (SDF). The results show the proposed explorer takes 43.95% less time to compute exploration points and paths when compared with the State-of-the-Art represented by the Receding Horizon Next Best View Planner (RH-NBVP) in Gazebo simulations.

Keywords: Autonomous exploration; Signed Distance Field; Rapidly-exploring Random Tree; Fast RRT*.

US

1. Introduction

Micro-Aerial Vehicles (MAVs) have turned into one of the most powerful robotic tools for human tasks. They can move in critical environments and have the ability to see in a wider range than humans at the ground with the proper sensors [1–4]. The flying path can be found using the sampling-based planners such as Probabilistic RoadMap (PRM) [5] and Rapidly-exploring Random Tree (RRT) [6] or heuristic planners like the A* algorithm [7]. However, it is not easy to perform safe movements in a three-dimensional (3D) environment when a collision-free path must be generated in unknown scenarios with unexpected obstacles.

The future of MAVs leads to a fully-autonomous system which can work even if the map is not available at the beginning. This brings a wide range of possibilities in exploration tasks such as: scene reconstruction, industrial inspection, transportation, search and rescue missions and many other applications that humans are not able to achieve.

There are different techniques for building the map when performing autonomous exploration. For example, Fedorenko *et al.* [8] used a group of MAVs designed to generate a point-cloud map of the scene and sent it to other Unmanned Ground Vehicles (UGVs). Dang *et al.* [9] presented a novel strategy, optimizing the exploration of unknown space by focusing the robot to the most salient objects. In order to solve the problem of MAVs autonomous exploration using on-board sensors such as Visual-Inertial (VI) sensors [10] and Hokuyo laser, a "*FAST RRT* 3D-Sliced Planner*" is presented. It consists of a sequential two-dimensional (2D) slices exploration of the 3D environment. For this 2D exploration, Optimal-Frontiers-Points (OFP) are computed at every slice using a global RRT frontier detector. Collision-free paths to these points are generated using the

Received 12 December 2020; Revised 16 July 2021; Accepted 16 July 2021; Published 15 September 2021. This paper was recommended for publication in its revised form by editorial board member, Yunfeng Zhang.
Email Address: *alvaromartineznovo@gmail.com, †liang.lu@upm.es

local FAST RRT* planner with a Signed Distance Field (SDF) collision checking algorithm. Thus, the computation is reduced and low resources are needed to complete the environment exploration in an optimal way. The proposed algorithm is validated using RotorS [11] in Gazebo.

There are three key innovations in this work: (a) The dimensionality of 3D exploration is reduced by slicing the environment in 2D independent explorations. (b) Exploration points are computed in 2D using the OFP algorithm. This shows that the computing time needed to explore is reduced when compared with other planners such as the RH-NBVP [12]. (c) The proposed FAST RRT* planner can find paths that are more optimal than the ones found by the conventional RRT algorithm but in less time than the original RRT* version.

The remainder of this paper is organized as follows: In Sec. 2, a brief overview of related works is given. In Sec. 3, the SDF Local FAST RRT* Planner is introduced. The 3D-Sliced Exploration algorithm is presented in Sec. 4 and the results are discussed in Sec. 5. Finally, Sec. 6 concludes the paper and summarizes future directions.

2. Related Work

A lot of methods have been proposed during the last decades addressing the problem of autonomous exploration in unknown environments. The frontier-based exploration was first proposed by Yamauchi [13]. This work explains that frontiers consist on regions on the boundary between mapped and unmapped space. By moving to successive frontiers, the map knowledge increases and full exploration is covered. Based on this concept, several innovations have been proposed. Umari and Mukhopadhyay [14] presented a global/local RRT-based frontier detection method for autonomous exploration. It uses a RRT-based algorithm to find frontiers, instead of conventional image processing tools that are limited to 2D. In this work, a point reached by the tree is considered as a frontier point if it, or at least any point at the edge, lies in the unmapped space. The RRT algorithm is perfect for this type of exploration because it is biased to unexplored regions and makes this process scalable to higher-dimensional spaces.

An efficient MAV 3D exploration using road maps was presented in [15]. The innovation of it lays in the incremental building of the road map that generates a topological structure of the 3D area during the exploration process. Thus, Next-Best-View (NBV) evaluation is computed by using this road map information and prompting the efficiency for NBV determination. Furthermore, a local planner based on the Artificial Potential Field (APF) [16] method is proposed for leading the position of the MAV.

As one of the most famous sampling-based exploration method, Receding Horizon "Next-Best-View" Planner (RH-NBVP) was first proposed by Bircher *et al.* RH-NBVP repeatedly expanded a Rapidly-exploring Random Tree and selected the best next node to explore using an objective function that considered the set of visible and unmapped voxels from robot configuration. This approach had two main disadvantages: (a) Keeping repeatedly expanding RRT is computing expensive. (b) Selecting the optimal next step in the field of view might lead to a sub-optimal solution. In 2018, Mahdoui *et al.* [17] presented another approach using a cooperative frontier-based exploration multi-MAV system. In this work, the MAVs are intended to explore specific regions of the environment in an optimized manner. In order to save communication bandwidth, the MAVs exchange map frontier points between them instead of the whole grid map. Thus, feasible fast cooperative mapping with low energy consumption is achieved.

Most of these methods presented work in a 3D grid computed using the Octomap framework [18]. However, the proposed approach achieves the 3D exploration computing OFP in 2D slices of the environment using the RRT-based frontier exploration. Map reconstruction and collision checking are computed using Voxblox [19] which has demonstrated to be more efficient than Octomap.

Another planner using Voxblox based on history-aware NBV [20] showed substantial improvements in the exploration time. To obtain a better sampling efficiency, this work used a history of previously visited places. These positions were used as seeds of the RRT to find informative regions when the mapped area increases far away from the robot and no gain is found in the vicinity. It reduces the sampling time as the RRT no longer needs to sample all the map from the current position. Schmid *et al.* [21] presented an informative path planning algorithm based on a continuously expanded single tree. Then, using an objective function, the trajectories are refined to maximize their utility. More recently, Lu also proposed an optimal frontier-based autonomous exploration algorithm for MAVs [22]. This algorithm uses Kmean++ to cluster the frontiers and information gain to select the best candidate to explore. The path to reach the selected frontier is generated using a corridor A* algorithm.

3. Local FAST RRT* Planner Using Signed Distance Field

In this section, the proposed FAST RRT* Planner is described. This planner is used to move the MAV to the OFP computed during the exploration process. The "*SDF-based Collision Checking Algorithm*", "*FAST RRT* Planner*" and three exploration strategies: "*Fixed Tree Resolution*", "*Sequential Local-Planning*" and "*Collision-Avoidance Flight Maneuver*" are discussed.

3.1. *SDF-based collision checking algorithm*

When applied to the mapping problem, SDF [23] is a type of representation generated from voxel meshes or 2D bitmaps. For better understanding, SDF is a function that takes a position as input and outputs the distance from that input to the nearest obstacle as shown in Fig. 1.

Another key concept about SDF is the sign of the distance stored in each voxel. Computing the subtraction of the inverted Euclidean Distance Transform (EDT) from the original, results into the SDF. In the SDF, negative values are inside the represented obstacle's contour while positives values are outside of it. This last feature can be used to decide whether the robot is inside the contour or outside of it and build a collision-free path. There are two well-known SDF representations.

One of them is the Truncated Signed Distance Field (TSDF), commonly used in computer graphics and surface reconstruction. The other one is the Euclidean Signed Distance Field (ESDF) which is accurate for collision checking in planning. The difference between TSDF and ESDF map representation is the process to compute the distance to the obstacle. In ESDF, each voxel stores the Euclidean distance to the nearest occupied voxel. On the other hand, the TSDF computes the projective distance from the voxel to the next occupied voxel along the ray direction of the sensor at that moment. This distance is truncated to only have values that are close to the surface. Thus, ESDF is used as map representation for collision checking. Because the MAV can be considered as an sphere with a fixed radius, it is assumed as R_{MAV}, then the collision status in the ESDF at any point can be computed as follows:

$$\text{checkCollision}(X_{\text{MAV}}) = \begin{cases} \text{FREE}, & d_{\text{ESDF}} - R_{\text{MAV}} > 0 \\ \text{OCCUPIED}, & d_{\text{ESDF}} - R_{\text{MAV}} < 0 \\ \text{UNEXPLORED}, & d_{\text{ESDF}} = \text{Infinite} \end{cases} \quad (1)$$

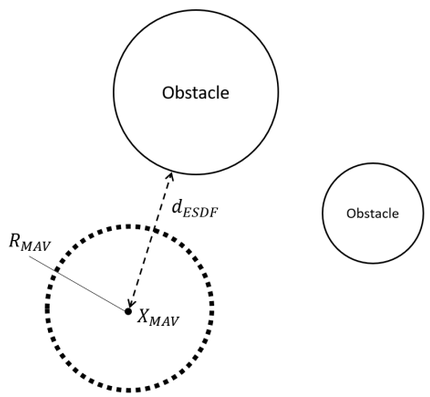


Fig. 1. Euclidean SDF.

Remark 3.1. UNEXPLORED voxels are those where ESDF is not computed.

3.2. *FAST RRT* planner*

RRT is a well-known sampling-based algorithm that can be used to solve path planning with low computational resources. It does not increase much complexity when applied to high-dimensional spaces. RRT has probability completeness, which means that, if initial and target positions are in the same connected area, then a path from the initial position to the goal position will be found after enough tree nodes extensions. There are many other implementations studied to improve the RRT performance. One of them is the Bidirectional RRT [24] where two trees grow from the initial position and the goal until they reach each other. Recently, the RRT* version that computes a low cost optimal path was also proposed [25]. In our work, FAST RRT* is presented as an hybridization between RRT and RRT* algorithms to solve the autonomous exploration problem in an optimal way.

The process of the proposed FAST RRT* algorithm is introduced here. The tree growth is exactly the same as the conventional RRT algorithm. First, the current position of the robot is set as the seed of the tree and then a random node is sampled. A new node will be added to the tree if it is collision free between it and the nearest node in the tree. This process is iterated until a new node in the tree is close enough to the desired goal. From here, RRT would search the path easily because the tree has a hierarchical structure, where each node has a parent node. Thus, the path is found instantly using a recursive process from the goal node to the seed of the tree.

However, in our proposed algorithm, this process is different and similar to the original RRT* reconnection

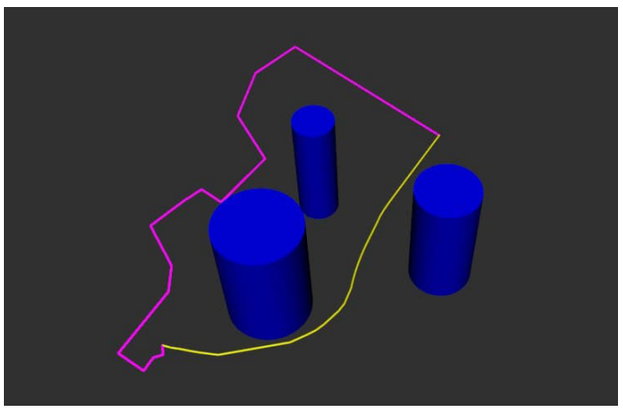


Fig. 2. FAST RRT* and RRT paths comparison in yellow and pink colors, respectively.

Algorithm 3.2 (FAST RRT* Planner).
Require: map_{ESDF}
Ensure: $path$

1: **function** FAST_RRT_STAR(X_{init}, X_{goal})
2: $Goal = onSearch$
3: **while** $Goal \neq Found$ **do**
4: $X_{rand} \leftarrow generateRandomPoint()$
5: $X_{near} \leftarrow findNearestNodeOnTree(X_{rand})$
6: $X_{inter} \leftarrow increaseResolution(X_{near}, X_{rand})$
7: **for all** $X_i \in X_{inter}$ **do**
8: **if** $checkCollision(X_i) \neq Occupied$ **then**
9: **if** $i > 1$ **then**
10: $X_{i-1} \leftarrow adoptNode(X_i)$
11: **else**
12: $X_{near} \leftarrow adoptNode(X_i)$
13: **end if**
14: **if** $checkGoal(X_i, X_{goal}) = True$ **then**
15: $Goal = Found$
16: $break$
17: **end if**
18: **else**
19: $break$
20: **end if**
21: **end for**
22: **if** $rrtTime() = True$ **then**
23: $Goal = NotFound$
24: $X_{ngoal} \leftarrow findNearestNodeOnTree(X_{goal})$
25: $reconnectNodes(X_{ngoal})$
26: **return** $generatePath(X_{ngoal})$
27: **end if**
28: **end while**
29: $reconnectNodes(X_{goal})$
30: **return** $generatePath(X_{goal})$
31: **end function**
32:
33: **function** RECONNECTNODES(X_{node})
34: **if** $X_{node} \neq X_{init}$ **then**
35: $X_{neigh} \leftarrow findNeighbours(X_{node})$
36: $sortByCost(X_{neigh})$
37: **for all** $X_i \in X_{neigh}$ **do**
38: **if** $collisionFree(X_i, X_{node})$ **then**
39: $X_i \leftarrow adoptNode(X_{node})$
40: $reconnectNodes(X_i)$
41: **return**
42: **end if**
43: **end for**
44: **else**
45: **return**
46: **end if**
47: **end function**

process performed during its tree growth to compute an optimal path. In the FAST RRT*, once the goal is reached, the path is found by reconnecting the tree at local level. First, the goal node searches for neighbors nodes at a fixed distance radio. Then, it reconnects to the lower cost node if possible. The cost is based on the distance traveled from the seed to this node. This process is repeated for every reconnected node until the seed of the tree is reached, generating the path. Fig. 2 shows a comparison between the path generated by RRT and the proposed FAST RRT* algorithm. Both are using the same tree until the goal is found and then each one looks for the path.

The FAST RRT* path is more optimal than the one found by the RRT, but also its execution time is faster than the RRT*. This is because the "*reconnectNodes*" method, defined in line 33 of Algorithm 3.2, is only executed when computing the path and not during the tree building as the original RRT* does. This reconnection process is only done for the minimum cost nodes during the path finding and not during the whole tree expansion process as explained before. Thus, the number of collision checking for nodes reconnection and computing resources are reduced. This property can speed up the optimal path finding, which make this planner suitable for path planning in big environments. Once the tree has been reconnected, the generated path is smoothed by using a polynomial trajectory optimization algorithm [26].

3.3. *Planning strategies*

In this section, to improve the performance during the exploration, three strategies are discussed. First, the *"Fixed Tree Resolution"* that can minimize the number of collision checking. Second, the *"Sequential Local-Planning"*, that helps to achieve points that are out of the planner range. Lastly, the *"Collision-Avoidance Flight Maneuver"* to ensure safe path tracking when unexpected obstacles appear.

3.3.1. *Fixed tree resolution*

To ensure a collision-free path from the nearest node X_{near} to the random one X_{rand}, the collision status between them must be checked. In the proposed planner, the resolution of the tree's edge is fixed by creating interpolated points and the collision is computed between them. This strategy comes out with two benefits:

(i) The times of collisions checking depend on the tree resolution. A proper minimum distance resolution should be equal to the MAV radius R_{MAV} to guarantee there is no collision between nodes from the same tree edge as it is shown in Fig. 3.
(ii) The segments consist of two nodes that are X_n and $X_{(n+m)}$. By using this approach, they are divided into finite nodes at a fixed resolution. Thus, the tree can grow at a higher precision, even in narrow areas.

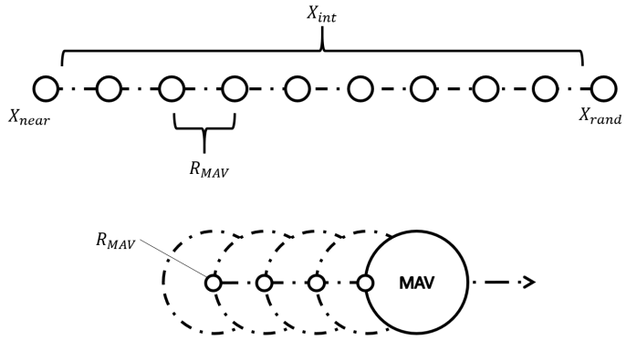


Fig. 3. Node-to-node interpolation.

Those interpolated nodes are described as X_{int}. The *i*th node is computed at each dimension, *j*, as it is shown in Eq. (2).

$$\begin{cases} X_{\text{int}(i)}(j) = X_{\text{near}}(j) + \dfrac{X_{\text{rand}}(j) - X_{\text{near}}(j)}{\lambda} \cdot i \\ \lambda = \dfrac{d}{\sigma} \end{cases} \tag{2}$$

Remark 3.3. λ is the interpolator value, d is the distance between nodes and σ is the resolution.

3.3.2. *Sequential local-planning*

To reduce the computation of the planner, it is executed in a limited local area. Thus, there is a high possibility that new exploration points cannot be achieved with just one plan. Sequential planning is used to solve this problem. When a new OFP is sent, the planner executes itself during a limited time. If no feasible path is found, the closest point to it is set as the inter-goal.

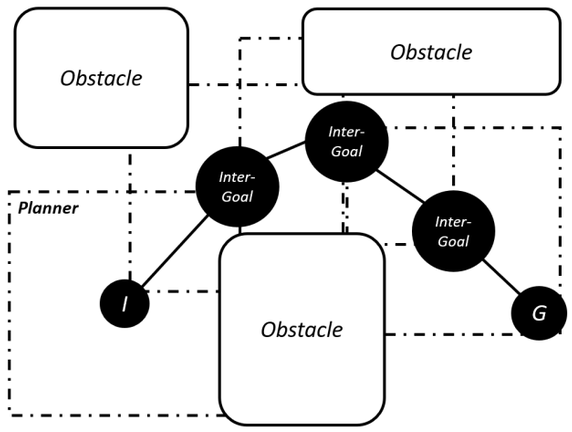


Fig. 4. Sequential planning.

Then, when it is reached, the planner executes itself again and retries to find a path to the original goal as the map knowledge increases (see Fig. 4). In the exploration, the MAV performs π rad/m spins during the path tracking to analyze its surroundings and cover much more space. This feature can improve the mapping performance.

3.3.3. *Collision-avoidance flight maneuver*

During the flight, the path is supposed to be safe, this means maintaining a secure distance from the obstacles. However, unexpected obstacles can be found during the exploration. Thus, the MAV can use the laser information to check the environment around it. When an unexpected obstacle is found, a collision-avoidance flight maneuver is executed and replanning is performed. With the new environment information of its surroundings, another path is computed.

The maneuver works as follows. As it is shown in Fig. 5, there are three frames: The *World* frame, which is fixed; the *Drone* frame, which is out of phase an angle of α_{yaw} from *World*; and the *Laser_range* frame, which is also out of phase π *rad* from the *Drone* frame. The maneuver consists of computing a new point (X_S, Y_S, Z_S) at a secure distance d_{safe} in the opposite direction to the nearest detected obstacle. Because the laser view is like a circle, when the nearest unexpected obstacle touches tangent, the normal vector to it can be computed using the θ_{obs} angle. This angle can be obtained from the minimum distance detected. When one of the measurements from *Laser_range*$[n]$ surpasses the secure distance threshold, its laser index *i* is stored. Then using this index and the resolution of the sensor, the θ_{obs} angle is computed.

The orientation of the MAV is kept during the maneuver. When the maneuver starts, the last safe orientation,

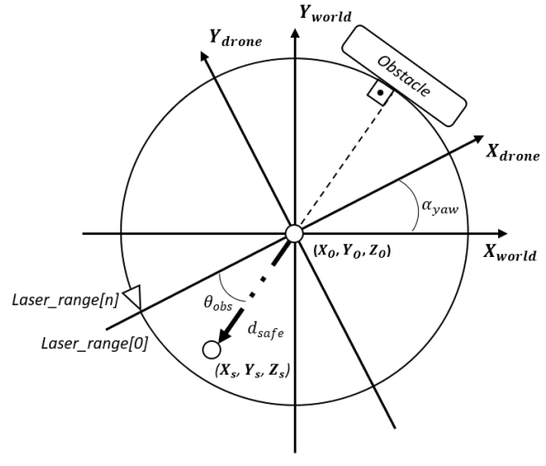


Fig. 5. Collision avoidance safe flight maneuver.

represented as q_O, is used as the new secure one, q_S. Then, a new safe point can be obtained using Eq. (3).

$$\begin{cases} \begin{bmatrix} X_S \\ Y_S \\ Z_S \end{bmatrix} = \begin{bmatrix} \cos(\phi) & 0 & 0 \\ 0 & \sin(\phi) & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdot d_{\text{safe}} + \begin{bmatrix} X_O \\ Y_O \\ Z_O \end{bmatrix} \\ q_S = q_O \end{cases} \quad (3)$$

Remark 3.4. The value of ϕ is equal to $(\theta_{\text{obs}} + \alpha_{\text{yaw}})$.

4. 3D-Sliced Exploration

This section explains the *"3D-Sliced Planner"* exploration algorithm. It consists of a sequential 2D slices exploration of the 3D environment. These slices are built using a 2D SLAM system based on the information recovered from the Hokuyo laser [27].

For this 2D exploration, OFP to explore are computed at every slice using the global RRT frontier detector algorithm. Path to these points is generated by using the proposed FAST RRT* planner. Thus, the computation is reduced and low resources are needed to complete 3D exploration in an optimal way.

The proposed 3D exploration algorithm is introduced here and its illustration is shown in Fig. 6.

The OFP searching process is combined with three steps. First, the global RRT-based frontier detector is used to generate the frontier points candidates. Then, in order to increase the computing efficiency, the mean shift filter is applied to cluster the candidates. Finally, the results of the filter step are sent to the robot task allocator module. A cost function including navigation cost, information gain, and the revenue from a frontier point is used to select the final OFP. More details can be found in [14].

Algorithm 4.1 (3D-Sliced Exploration).
Require: map_{2D}, map_{ESDF}
Ensure: $Exploration$

```
function 3DS_Exploration(map_2D, map_ESDF)
    while Exploration ≠ Completed do
        newLevel ← Z_MAV + 2R_MAV
        if checkCollision(newLevel) = Free then
            MAVFlyUp(newLevel)
            2D_RRT_Exploration()
        else
            Exploration = Completed
        end if
    end while
end function

function 2D_RRT_Exploration(map_2D, map_ESDF)
    while True do
        frontiers ← frontierDetector()
        if frontiers > 0 then
            filteredFrontiers ← filter(frontiers)
            OFP ← assigner(filteredFrontiers)
            FAST_RRT_STAR(XYZ_MAV, OFP)
        else
            break
        end if
    end while
end function
```

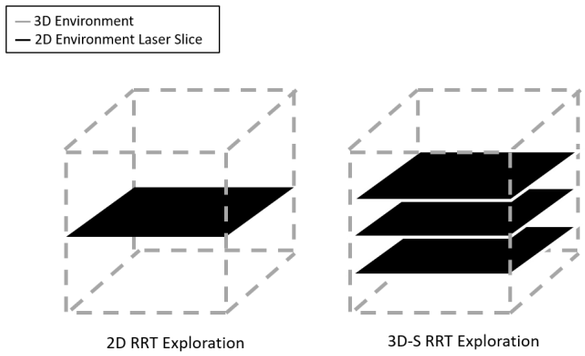


Fig. 6. 2D versus 3D-S Exploration.

An individual 2D slice exploration is considered completed when the global RRT frontier detector is not able to discover more frontier points. Then, the MAV checks if it has enough free space upwards at the moment. It checks the collision status of the point whose altitude is equal to the MAVs current position plus its sphere collision diameter. In case that it is collision-free, the MAV flies up and starts a new 2D slice exploration of the 3D environment using the 2D RRT exploration algorithm. Thus, when the 2D slice is completely explored, it starts again at a new height. When there is no more free space upwards, the MAV considers that there are no more slices to explore, and the 3D exploration is finished. The proposed approach has two advantages:

(i) OFP are computed at a 2D space. This reduces the complexity when compared with 3D exploration.
(ii) 3D environment space is divided into single 2D slices. The distance between every slice is much smaller than the VI-sensor Z-axis FoV used. This means that the same points are rediscovered from different views at each slice. This implies better 3D reconstruction.

Fig. 7 shows the whole connections between each subsystem and the MAV using the proposed 3D-S exploration system.

5. Experiments and Results

This section shows the whole exploration system results using the algorithms explained in the previous sections. There are two different scenarios in total and the experiments consist in reconstructing them. Furthermore, a comparison with RH-NBVP is discussed.

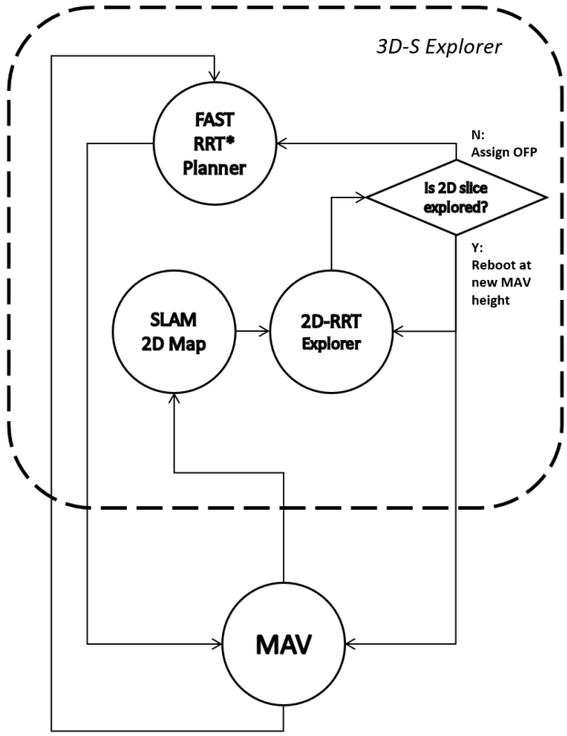


Fig. 7. 3D-S Explorer scheme.

5.1. *Experimental setup*

In this section, the simulation environments in the experiments are described. For the simulation experiments, these are run on top of Ubuntu 18.04 using the Robot Operating System (ROS) [28]. The Gazebo simulator is used to provide the environment model and RotorS is employed to provide the physical model parameters of the robot. All the experiments run on a laptop with Intel Core i7-9750H at 2.6 GHz. The robot used in the simulation is the AscTec Firefly, which is equipped with a Hokuyo laser rangefinder with a FoV of 360° and ray length of 11 m. It also used the VI-sensor camera with an horizontal field of view of 98° and vertical field of view of 73°. The sensing range of the camera is set as 7 m. The camera has an angle of 6° looking down to the ground. In the simulation experiments, the ground truth is used to perform the robot localization.

5.1.1. *Scenario A*

To evaluate our exploration algorithm, it is first tested in the environment as shown in Fig. 8. The size of the environment is 20 m × 20 m × 3.5 m.

As can be seen from Fig. 8, the environment is covered by a blue transparent ceiling and has two stages separated

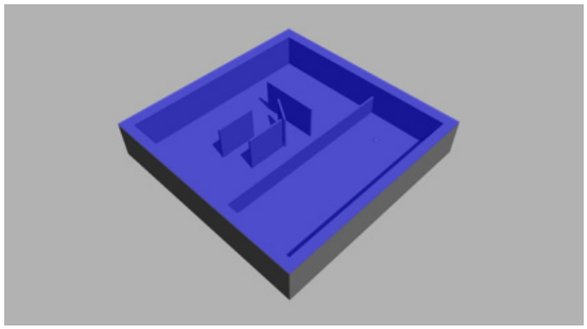


Fig. 8. Gazebo scenario A.

by a wall that does not reach the ceiling. The MAV takes off from area one, A_1, which is an empty room without obstacles, three walls that reach the blue-covered ceiling and one limited wall that does not reach it.

Compared with area A_1, area A_2 is more complicated. There are four more walls that reach the ceiling in area A_2. The only way to pass from A_1 to A_2 is by flying over the middle-limited wall. The dimensions of the testing environment lead the MAV to perform three slices of 2D RRT exploration. The experiments consist mainly of the reconstruction of the environment and the discovering of area A_2 behind the middle separating wall using the proposed 3D exploration strategy. The results are shown in the next section.

5.1.2. *Scenario B*

The second scenario is more complex than the first one. It consists on a house distributed in several areas, six in total. These are connected by windows, doors and narrow hallways. The environment is shown in Fig. 9 and its size is 20 m × 20 m × 2.5 m.

This is a challenging scenario as it requires the explorer to be able to find unexplored areas of different sizes across narrow accesses. Furthermore, there are lots of ways to explore it because of its multiple connections. This will be critical

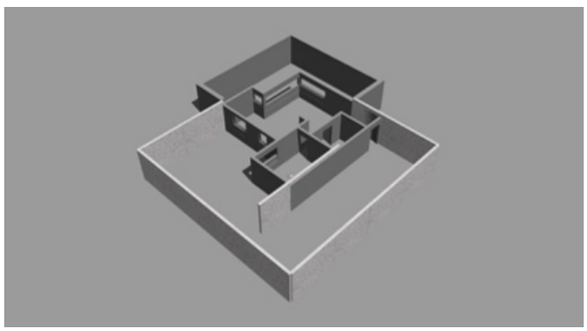
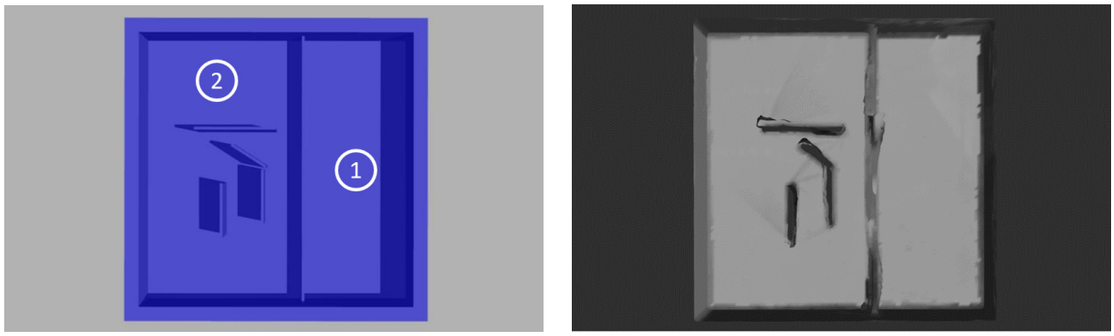


Fig. 9. Gazebo Scenario B.



(a) (b)

Fig. 10. Scenario A (top side view) 3D map.

when exploring to achieve a significant time reduction of the scene reconstruction. Also, there is one room that is only accessible by flying across its windows. The rest of the areas can be reached using the doors, but using the windows in a smart way can speed up the exploration time. The dimensions of this environment lead the MAV to perform two slices of 2D RRT exploration. The results are also discussed in the next section.

5.2. *Results and discussion*

In this section, the results of the *"FAST RRT* 3D-Sliced Planner"* are given. The MAV flies at a maximum velocity of 2 m/s and a maximum acceleration of 2 m/s^2. Also, as explained before, it performs π rad/m spins during the path tracking.

Fig. 10(a) is the Scenario A Gazebo environment with its two areas numbered, while Fig. 10(b) shows its reconstruction. The MAV is launched from the area A_1 at point (0,0,0) and it finishes the exploration in area A_2.

Furthermore, Fig. 11 shows how, during the first two 3D slices of exploration, the MAV flies only in the area A_1. When computing the third exploration slice, the MAV discovers the second area and continues the exploration crossing above the middle wall.

On the other hand, Fig. 12, shows the Scenario B Gazebo environment reconstruction during time, and the tracked exploration path. The last figure of the sequence shows the simulation environment with its six areas numbered. The MAV is launched at the central area A_1 in point (0,0,0) and it completes the whole scene reconstruction in area A_6. In this experiment, two slices of the 3D environment were explored.

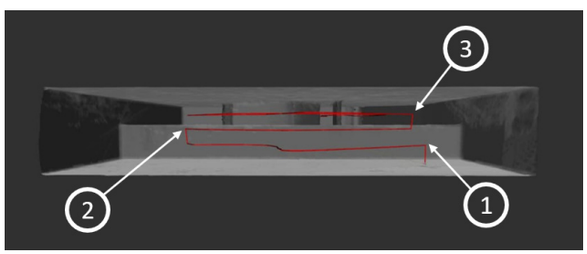


Fig. 11. Scenario A exploration path (three slices).

From area A_1 is it possible to access area A_5 by flying across its window. Also, area A_2 is reachable by using the door. However, during the experiments the MAV could not see the available window in the first slice, so it continued going to area A_2 and area A_3. As before, area A_6 can be reached from area A_2, but only using its windows. Because this windows were not visible from the first exploration slice, the MAV continued moving to area A_4 and area A_5. Then the MAV executed the second 3D slice and continued the exploration. At this new height, all the windows are visible to the MAV, so it continues exploring from area A_5 to area A_2 and then to area A_6 using the windows accesses. Thus, the MAV completes the exploration.

A video demonstration of the experiments of our exploration system is available in:

https://vimeo.com/479886339

In order to evaluate the proposed exploration algorithm, the RH-NBVP is used to compare with it. Table 1 shows the execution time of one run of the proposed algorithm and Table 2 gives the time consuming of one run of the NBV algorithm when exploring the environment.

As can be seen from Tables 1 and 2, computing the exploration points using the OFP algorithm, results into a 43.95% computation time reduction against the NBV. This is a critical improvement as it could lead the MAV to reach higher speeds to explore the surroundings in less time.

Furthermore, the RH-NBVP is also tested in both Scenarios A and B, to compare the path length and the exploration time with the proposed approach. The path length and the exploration time comparison between both algorithms are shown in Figs. 14(a) and 14(b), respectively. The results are also shown in Table 3.

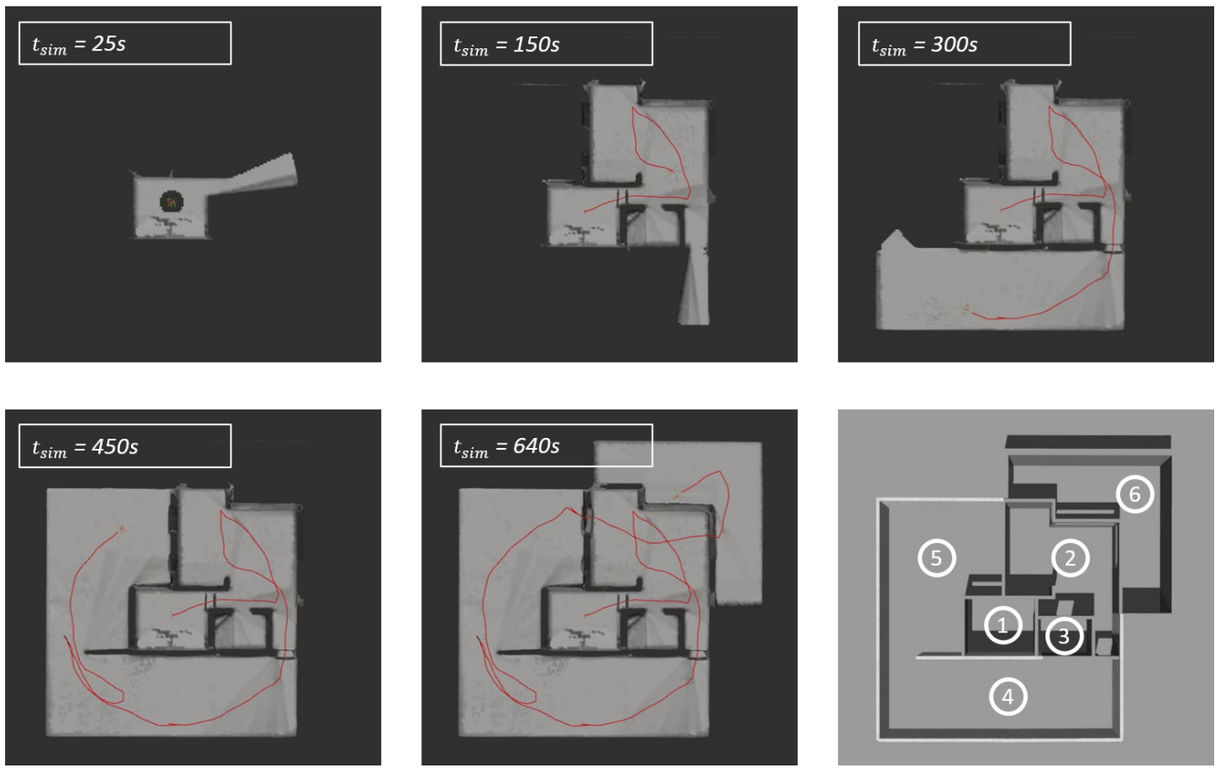


Fig. 12. Scenario B (top side view) exploration process with time.

Table 1. OFP time consuming.

ID	Time cost (ms)
G-RRT Detector	2.7 ± 1.5
Filter	80.6 ± 21.9
Assigner	71.8 ± 43.8
OFP	155.1 ± 45.5
L-FAST RRT* Path	53.3 ± 44.1
Total OFP planning	$\mathbf{208.4 \pm 62.7}$

Table 2. NBV time consuming.

ID	Time cost (ms)
Total NBV planning	371.8 ± 110.9

Table 3. Experiments results comparison.

Simulation environment	Map resolution (m)	Exploration algorithm	Path length (m)	Exploration time (s)
Scenario A	0.1	3D-S	$\mathbf{93.6 \pm 9}$	$\mathbf{674.38 \pm 44.2}$
	0.2	NBV	120.67 ± 6.36	787.22 ± 100.7
Scenario B	0.1	3D-S	$\mathbf{96.53 \pm 9.87}$	$\mathbf{748.57 \pm 66.7}$
	0.2	NBV	116.93 ± 11.39	958.13 ± 62.22

During the experiments, in Scenario A, the MAV took more time than expected to explore the area A_1. This is because the MAV could not discover the area A_2 until the third exploration slice. In this last slice, the middle

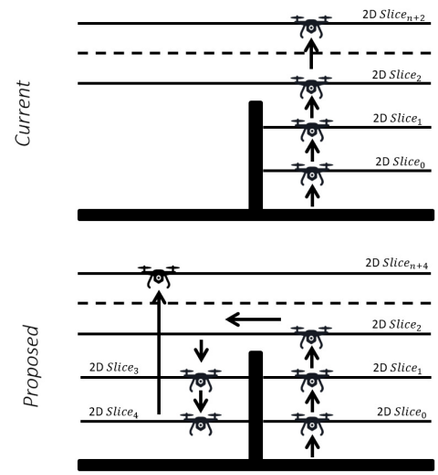
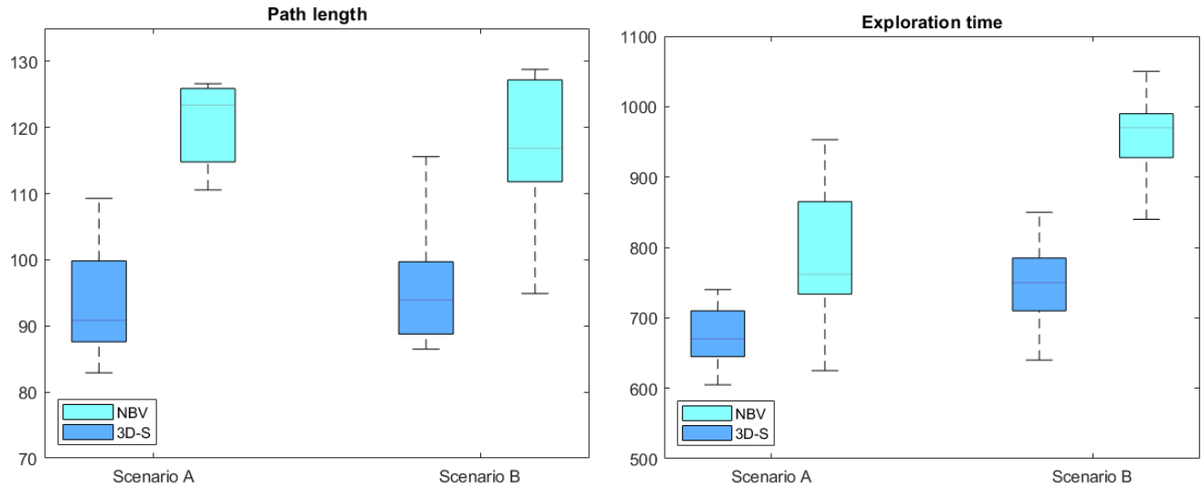


Fig. 13. Current 3D-Sliced exploration and new future strategy proposed, respectively.



(a) (b)

Fig. 14. Exploration path and time comparison.

separating wall does not appear in the Hokuyo laser range. In the last slice, the rest of the environment can be discovered by the 2D SLAM and the OFP of A_2 are included in the exploration. A_1 is explored using three 2D slices of the environment. On the other hand, the area A_2 is explored using just the last slice, but the reconstruction is still good because of the vertical FoV of the sensor that is large enough in this experiment to cover all the height. However, the MAV could not explore the new area using only one slice exploration if the current altitude is bigger than the vertical FoV of the perception sensor.

The same happened in Scenario B, where area A_6 was discovered at the second slice and the MAV was then capable to see the accesses to it. As can be seen in Fig. 13, in order to solve this problem, a planning strategy that can compute the new 2D slices at lower altitude when new areas are discovered is proposed as future work. This strategy could lead to a complete and dense exploration in really complex scenarios.

As it is shown in the last figures, there is a significant path length and exploration time reduction when using the proposed *"FAST RRT* 3D-Sliced Planner"*. This is because the MAV flies directly to the frontiers of the environment in an optimal way, exploring the environment using a global explorer and not a local one, as it is the RH-NBVP.

6. Conclusions and Future Work

In this paper, a new 3D exploration strategy for MAVs is presented. The innovation of this work consists in the slicing of the 3D environment and its optimal exploration with low resources. Thus, the exploration is performed in 2D instead of 3D, so the computation can be reduced. The exploration system is based on the global RRT frontier detector to compute the OFP, and the new proposed local FAST RRT* planner to lead the position of the MAV to these points in an optimal way. Efficient collision checking for path planning is achieved by using ESDF representation from Voxblox.

The slicing strategy of the environment explained in the section before and shown in Fig. 13 will be proposed as future work. This will lead the explorer to reconstruct properly the whole new areas discovered during the exploration. Future work will also consider the post-processing of the scene reconstructed as it is done in [29]. Real test will also be proposed to ensure the performance of our system. It will be also proposed to use the new Voxblox++ framework [30] to build maps that contain information about the individual object instances observed in the scene.

Acknowledgment

This research was funded by the Spanish Ministry of Science, Innovation and Universities (COMCISE RTI2018-100847-B-C21, MCIU/AEI/FEDER, UE) and the Chinese Scholarship Council (CSC) for the Scholarship of the second author.

References

[1] A. A. Ravankar, A. Ravankar, Y. Kobayashi and T. Emaru, Autonomous mapping and exploration with unmanned aerial vehicles using low cost sensors, *Proceedings* **4**(1) (2019) 44.

[2] M. Faria, A. S. Ferreira, H. Pérez-Leon, I. Maza and A. Viguria, Autonomous 3D exploration of large structures using an UAV equipped with a 2D lidar, *Sensors* **19**(22) (2019) 4849.

[3] M. Alnuaimi and M. G. Perhinschi, Investigation of alternative parameters for immunity-based UAV navigation in GNSS-denied environment, *Unmanned Syst.* **09**(01) (2021) 65–72.

[4] S. Khan, M. Tufail, M. T. Khan, Z. A. Khan, J. Iqbal and A. Wasim, A novel framework for multiple ground target detection, recognition and inspection in precision agriculture applications using a UAV, *Unmanned Syst.* **0** (2021) 1–12.

[5] L. E. Kavraki, P. Svestka, J. Latombe and M. H. Overmars, Probabilistic roadmaps for path planning in high-dimensional configuration spaces, *IEEE Trans. Robot. Autom.* **12** (1996) 566–580.

[6] S. M. Lavalle, Rapidly-exploring random trees: A new tool for path planning, in *Algorithmic and Computational Robotics New Directions* (CRC Press, 1998), pp. 293–308.

[7] X. Liu and D. Gong, A comparative study of a-star algorithms for search and rescue in perfect maze, *2011 Int. Conf. Electric Information and Control Engineering* (2011), pp. 24–27.

[8] R. Fedorenko, A. Gabdullin and A. Fedorenko, Global UGV path planning on point cloud maps created by UAV, *2018 3rd IEEE Int. Conf. Intelligent Transportation Engineering (ICITE)* (2018), pp. 253–258.

[9] T. Dang, C. Papachristos and K. Alexis, Visual saliency-aware receding horizon autonomous exploration with application to aerial robotics, *2018 IEEE Int. Conf. Robotics and Automation (ICRA)* (2018), pp. 2526–2533.

[10] J. Nikolic, J. Rehder, M. Burri, P. Gohl, S. Leutenegger, P. T. Furgale and R. Siegwart, A synchronized visual-inertial sensor system with FPGA pre-processing for accurate real-time SLAM, *2014 IEEE Int. Conf. Robotics and Automation (ICRA)* (2014), pp. 431–437.

[11] F. Furrer, M. Burri, M. Achtelik and R. Siegwart, *RotorS "A Modular Gazebo MAV Simulator Framework* Vol. 01 (Springer, 2016), pp. 595–625.

[12] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova and R. Siegwart, Receding horizon "next-best-view" planner for 3D exploration, *2016 IEEE Int. Conf. Robotics and Automation (ICRA)* (2016), pp. 1462–1468.

[13] B. Yamauchi, A frontier-based approach for autonomous exploration, in *Proc. 1997 IEEE Int. Sympos. Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation'* (1997), pp. 146–151.

[14] H. Umari and S. Mukhopadhyay, Autonomous robotic exploration based on multiple rapidly-exploring randomized trees, *2017 IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)* (2017), pp. 1396–1402.

[15] C. Wang, H. Ma, W. Chen, L. Liu and M. Meng, Efficient autonomous exploration with incrementally built topological map in 3D environments, *IEEE Trans. Instrum. Meas.* **PP** (2020) 1–1.

[16] Q. Zhu, Y. Yan and Z. Xing, Robot path planning based on artificial potential field approach with simulated annealing, *Sixth Int. Conf. Intelligent Systems Design and Applications*, Vol. 2 (2006), pp. 622–627.

[17] N. Mahdoui, V. Frmont and E. Natalizio, Cooperative frontier-based exploration strategy for multi-robot system, *2018 13th Annual Conf. System of Systems Engineering (SoSE)* (2018), pp. 203–210.

[18] A. Hornung, K. Wurm, M. Bennewitz, C. Stachniss and W. Burgard, OctoMap: An efficient probabilistic 3d mapping framework based on octrees, *Autonomous Robots* **34** (2013) 189–206.

[19] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart and J. Nieto, Voxblox: Incremental 3D euclidean signed distance fields for on-board MAV planning, *2017 IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)* (2017), pp. 1366–1373.

[20] C. Witting, M. Fehr, R. Bhnemann, H. Oleynikova and R. Siegwart, History-aware autonomous exploration in confined environments using MAVs, *2018 IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)* (2018), pp. 1–9.

[21] L. Schmid, M. Pantic, R. Khanna, L. Ott, R. Siegwart and J. Nieto, An efficient sampling-based method for online informative path planning in unknown environments, *IEEE Robot. Autom. Lett.* **5**(2) (2020) 1500–1507.

[22] L. Lu, C. Redondo and P. Campoy, Optimal frontier-based autonomous exploration in unconstructed environment using RGB-D sensor, *Sensors* **20**(22) (2020), doi: 10.3390/s20226507.

[23] H. Oleynikova, A. Millane, Z. Taylor, E. Galceran, J. Nieto and R. Siegwart, Signed distance fields: A natural representation for both mapping and planning, *RSS 2016 Workshop: Geometry and Beyond - Representations, Physics, and Scene Understanding for Robotics* (University of Michigan, Ann Arbor, MI, 2016).

[24] W. Xinggang, G. Cong and L. Yibo, Variable probability based bidirectional RRT algorithm for UAV path planning, *The 26th Chinese Control and Decision Conf. (2014 CCDC)* (2014), pp. 2217–2222.

[25] L. Chen, Y. Shan, W. Tian, B. Li and D. Cao, A fast and efficient double-tree RRT*-like sampling-based planner applying on mobile robotic systems, *IEEE/ASME Trans. Mechatronics* **23** (2018) 2568–2578.

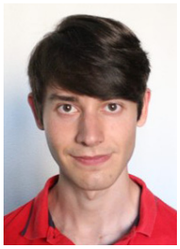
[26] C. Richter, A. Bry and N. Roy, Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments, *Robot. Res.* **114** (2016) 649–666.

[27] S. Kohlbrecher, O. von Stryk, J. Meyer and U. Klingauf, A flexible and scalable slam system with full 3D motion estimation, *2011 IEEE Int. Symp. Safety, Security, and Rescue Robotics* (2011), pp. 155–160.

[28] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler and A. Ng, ROS: An open-source robot operating system, *ICRA Workshop Open Source Software* Vol. 3 (2009).

[29] S. Omari, P. Gohl, M. Burri, M. Achtelik and R. Siegwart, Visual industrial inspection using aerial robots, in *Proc. 2014 3rd Int. Conf. Applied Robotics for the Power Industry* (2014), pp. 1–5.

[30] M. Grinvald, F. Furrer, T. Novkovic, J. J. Chung, C. Cadena, R. Siegwart and J. Nieto, Volumetric instance-aware semantic mapping and 3D object discovery, *IEEE Robot. Autom. Lett.* **4** (2019) 3037–3044.



Álvaro Martínez Novo was born in Granada, Spain, in 1997. He obtained his B.Sc. degree in Industrial Electronics Engineering from the Universidad de Granada (UGR), Spain, in 2019. In 2020, he received his M.Sc. degree in Automation and Robotics from the Universidad Politécnica de Madrid (UPM), Spain. He worked at the Research Group on "Computer Vision and Aerial Robotics" (CVAR) at UPM, within the Centre of Automatics and Robotics (CAR). He is currently a Ph.D. candidate in the Centro de Investigación en Tecnologías de la Información y de las Comunicaciones de la Universidad de Granada (CITIC-UGR). His main research interests include autonomous navigation, computer vision and bio-inspired solutions applied to aerial vehicles. Furthermore, he is currently a researcher and the main technical developer in several projects that aims to increase the autonomy of the Unmanned Aerial Vehicles (UAVs) using different technologies, including bio-inspired sensors.

Liang Lu was born in Zhejiang, China, in 1992. He received his B.S. degree in Forestry Engineering (Intelligent Equipment Engineering) in 2014 from Northeast Forestry University, Heilongjiang, China, and M.S. degree in Mechatronic Engineering in 2017 from Hefei Technology of University, Anhui, China. He is currently a Ph.D. candidate in Automatic and Robotics from the Research Group on "Computer Vision and Aerial Robotics" (CVAR), Centre for Automation and Robotics (CAR), Universidad Politécnica de Madrid (UPM), Madrid, Spain. His research interest is focused on autonomous navigation, computer vision and aerial robots. He has published several papers in the international conferences and journals. He has also participated in several international competitions, such as IROS ADR 2018, IMAV 19 and MBZIRC 2020. In MBZIRC 2020, he received the third place prize in the grand challenge. He was also a reviewer of several international journals and conferences, such as Robotica, Frontiers in Neurorobotics, IROS 2020, IROS 2021 and ICAR 2021. His personal homepage is: https://www.researchgate.net/profile/Liang˙Lu27.

Pascual Campoy is Full Professor on Automatics at the Universidad Politécnica de Madrid UPM (Spain) where he lectures on Control, Machine Learning and Computer Vision. He has been visiting professor at DCSC Department in TUDelft (The Netherlands) from 2014 to 2019, and previously visiting professor at Tong Ji University (Shanghai-China) in 2013 and Q.U.T. (Australia) 2011. He received his Ph.D. in Automatics and Robotics at Universidad Politécnica de Madrid in 1988, where he previously received his Master tittle in Automatics Engineering in 1983. He is leading the Research Group on "Computer Vision and Aerial Robotics" (CVAR) at UPM within the Centre of Automatics and Robotics (CAR), whose activities are aimed at increasing the autonomy of the Unmanned Aerial Vehicles (UAV) by exploiting the powerful sensor of Vision, using cutting-edge technologies in Image Processing, Control and Artificial Intelligence. He is the Heading Director of over 40 R&D projects, including R&D European projects, national R&D projects and over 25 technological transfer projects directly contracted with the industry. He is author of around 200 international scientific publications and nine patents, three of them registered internationally. He is awarded in the top international UAV competitions: IMAV12, IMAV13, IARC14, IMAV16 and IMAV17, General Chair for IMAV 2019 and he coordinated the international team that was awarded the third place in the Grand Challenge MBZIRC20.