2021 International Conference on Unmanned Aircraft Systems (ICUAS)
Athens, Greece. June 15-18, 2021

An Efficient Object-Oriented Exploration Algorithm for Unmanned Aerial Vehicles

Edwin Paúl Herrera Alarcón[§], Davide Bagheri Ghavifekr[§], Gabriele Baris[§], Michael Mugnai,
Massimo Satler and Carlo Alberto Avizzano
PERCRO Laboratory @ IIM Institute
Department of Excellence in Robotics and A.I.
Scuola Superiore Sant'Anna
Pisa, Italy

2021 International Conference on Unmanned Aircraft Systems (ICUAS) | 978-1-6654-1535-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICUAS51884.2021.9476764

***Abstract*— Autonomous exploration of unknown environments usually focuses on maximizing the volumetric exploration of the surroundings. Object-oriented exploration, on the other hand, tries to minimize the time spent on the localization of some given objects of interest. While the former problem equally considers map growths in any free direction, the latter fosters exploration towards objects of interest partially seen and not yet accurately identified.**

The proposed work relates to a novel algorithm that focuses on an object-oriented exploration of unknown environments for aerial robots, able to generate volumetric representations of surroundings, semantically enhanced by labels for each object of interest.

As a case study, this method is applied both in a simulated environment and in real-life experiments on a small aerial platform.

I. INTRODUCTION

The latest advancement in aerial robotics research combined with the inherent characteristics of *Micro Aerial Vehicles* (MAV) systems, i.e. flexibility, and low cost, has led to their extensive usage in a variety of civil tasks. MAV platforms are valuable since they allow to gather information quickly, with high precision, and not less important, the use of such a technology in dangerous environments may also reduce the risks to human operators. Some of the best known tasks performed by drones are infrastructure inspection [1], [2], surveillance and exploration [3], [4], 3D building reconstruction [5], [6], precision agriculture [7], industrial inspection [8], deliver of goods [9] and, recently, SARS-CoV-2 disinfection of indoor facilities [10].

An area of study of UAV that is constantly increasing interest in academic research, as well as international funding agencies and organisations, is *Search and Rescue* (SAR). For example, since 2018 DARPA has promoted the Subterranean Challenge* (SubT) *Search and Rescue competition* and the EU has founded many projects like *SHERPA* in 2013 (FP7-ICT). Both of these projects are oriented to autonomous searching for objects in hazardous scenarios.

In a SAR mission, localizing survivors is a challenging problem as building plans cannot be used as a reference for navigation due to structural changes that may have occurred. Rescue workers face challenging situations in post-disaster scenarios while carrying out their missions. They have to be aware of possible threats that could endanger the exploration, like gas leaks, weak building structures or suspicious packages; as well as to any possible clues that may lead to a victim faster or explain the disaster's cause. However, even if rescue workers should take their time to meticulously explore the environment, time is also a critical factor that can impact a victim's survival. In this context MAVs can provide a fast response due to their capability to sneak through rubble-obstructed paths and reach full exploration of the place while minimizing the exposure and risk of human workers' injury. Nevertheless, to be helpful in real-life application, MAVs must rely on autonomous features to be able to either assist the remote operator or perform autonomously the task.

The work presented in this paper focuses on the searching of objects belonging to certain classes of interest during the exploration of an unknown environment by a MAV platform. The contribution of this work is a vision-based autonomous exploration algorithm oriented towards object localization. The main novelty of the proposed algorithm is a new approach for finding objects in a dense map by labeling their bounding boxes. Based on a *next-best-view* algorithm, it is paired with a strategy to steer the exploration towards points of interest detected with deep learning techniques.

The proposed algorithm utilizes an RGB-D camera to obtain information from the objects of interest, which is later included in the exploration map to obtain a semantically enhanced knowledge of the environment. Furthermore, if the object in question is beyond the limits of the currently explored map, it is projected to the known space's frontier, as shown in Fig. 1. In this way, the autonomous system keeps a record of what it sees and increases its attention in the object's direction. The efficiency of the algorithm is evaluated by measuring the average time of object localization.

The remaining of this paper is structured as follows: section II provides a brief description of the state-of-the-art solutions for autonomous exploration algorithms. In section III, the problem is stated and the proposed approach is described, while experiments are depicted and analyzed in section IV. Finally, section V summarizes the main novelties, results and future project developments.

[§]Authors contributed equally.
(Corresponding author: Edwin Paúl Herrera Alarcón, e-mail: e.herreraalarcon@santannapisa.it)
*www.subtchallenge.com

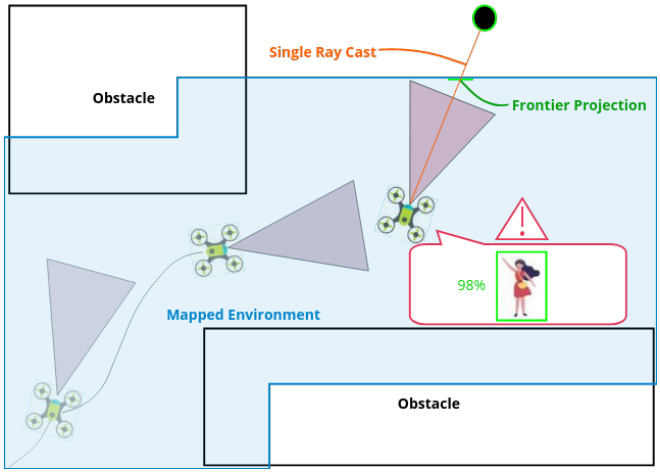


Fig. 1: Proposed algorithm core idea. Projection of object detection to the currently explored map frontier. Object localization uses a single ray cast, later expanded using a wave-alike propagation.

II. RELATED WORK

Autonomous exploration in unknown environments is a problem that has been addressed with many methods, which can be summarized in two main categories: Frontier-based and Sampled-based. The frontier-based approach continuously explores the boundaries between known and unknown space, maximizing the exploration of the scenario. On the other hand, sample-based methods explore new paths in the already explored map and select the most valuable ones. An example of this kind of algorithms is the *Next-Best-View* (NBV) formulation[11], which expands a *Rapidly-exploring Random Tree* (RRT) of new sampled viewpoints in the map and looks for the best node, i.e. the one that maximizes an exploration-related objective function.

Juliá *et al.* [12] highlight the difference in exploration time between autonomous exploration methods, pointing out that frontier-based approaches take more time to explore an unknown environment because their criteria chooses low-cost solutions that are usually near the current position. Whereas, sample-based approaches search for areas with high value which may postpone small areas to later stages in the exploration task. Lu *et al.* [13] presented a work for an efficient and optimal strategy to generate points in frontier-based approaches using information gains aiming to save exploration time. Moreover, sample-based approaches easily allow adding gain formulations to the overall exploration gain, such as object visualization or unexplored volume [14], [15], [16].

Selin *et al.* [17] proposed a hybrid exploring strategy by mixing both frontier-based and sample-based algorithms. The former was used to select the goal for the global exploration and the latter was adopted for local exploration of the environment. On the other hand, Schmid *et al.* [18] proposed an online informative path planning algorithm to overcome costly RRT computation and local minimum problems. Basically, the proposed solution keeps alive non-executed parts of new generated sampled viewpoints expanding a growing tree between algorithm iterations and moreover it uses a single objective function to perform global searching. The proposed solution has been compared with the standard NBV formulation [11] and the previously mentioned hybrid approach [17]. The evaluation was carried on comparing the ability to explore and reconstruct volumetric indoor environments and the results showed that the hybrid approach as well as the informative path planning, i.e. [17] and [18] respectively, outperform the standard NBV planner. Furthermore, Schmid *et al.* [18] algorithm performs slightly better regarding path planning generation thanks to their continuous improvement of the unique grown tree.

Dang *et al.* in [14] and [15] have proposed methods for autonomous exploration and simultaneous object search that enhanced an NBV planner with a specific gain for objects detection. Their exploration also produces a semantic map, based on Octomap [19], which gives more information about the environment's occupancy. Also, Ashour *et al.* [20] suggests a semantically-aware exploration for object detection using semantic segmentation techniques, using as well an NBV formulation. While highly effective, these methods have limitations related to discarding large parts of the tree on each iteration and the possibility of selecting locally optimal solutions that may lead to sub-optimal paths [18].

Some exploration algorithms have already been tested in real-life hazardous scenarios as the ones used by the competitors of DARPA Subterranean (SubT) Challenge. This competition tests exploration algorithms in a challenging scenario as SAR missions. Among the participant teams, of particular interest are the work of Petrlik *et al.* [21] and Dang *et al.* [22] which decided to use YOLO Convolutional Neural Network (CNN) for the object detection of artifacts. However, considering the low visibility of this typology of SAR environments, the exploration algorithms employed are fully focused on volumetric gain rather than an object-oriented search.

III. PROPOSED APPROACH

The ideal behaviour for a robot designed for an autonomous exploration task is the capability to explore as fast as possible the unknown environment while focusing the attention on specific artefacts valuable for the mission accomplishment. In term of requirements, this behaviour translates to an object-oriented exploration where the system must perform a fast but selective inspection of the environment. To accomplish such a desired behavior, we design a system that localizes specific items in the scene and either maps them in a 3D occupancy map or steer the mapping process towards the detected object if it relies on outside the mapped space. The proposed method is based on Schmid *et al.* [18] path planning where we introduce: (i) a new gain formulation to balance between exploring unknown parts of the scene and move towards objects of interest; (ii) an enhanced map representation based on Voxblox, and finally, (iii) a new method to retrieve the voxels representing an object inside a bounding box.

A. Problem Description

The problem addressed in this work refers to the searching problem of objects belonging to certain classes of interest (e.g. civilians or vehicles) while exploring an unknown bounded volume $V \subset \mathbb{R}^3$.

The robot configuration ξ_i is defined as the position $\vec{x}_i$ and orientation $\vec{q}_i$ of the robot at instant i with respect to a fixed *world frame* W. The exploration problem is defined as to find a safe and collision-free path $\{\xi_0, \xi_1, \ldots, \xi_n\}$ to run across; during the exploration, the previously unknown volume can be identified as either free (V_{free}) or occupied (V_{occ}) and discretized in a map M composed by voxels, a representation of the continuous space with cubical volumes. Thanks to an object detection algorithm we are also able to tell which object the voxel represents.

B. Object Detection

The most widely used method for environmental awareness considers using a computer vision algorithm on frames captured from a 2D color camera. Therefore, we assume that the robot is equipped with an on-board camera rigidly attached to it, so that a static transformation between the robot frame and the camera frame exists and can be defined at the design stage.

The object detection module takes color camera frames as input and gives, for each frame, a set of bounding boxes around the objects of interest as output. A bounding box is defined as (x, y, w, h), where (x, y) is its position in the image and w and h is its width and height, respectively. All coordinates are expressed in pixels.

The actual logic implemented to find objects in the image may vary and depends on the specific application domain. For simple tasks, it may be sufficient to apply legacy computer vision techniques (e.g. color segmentation, contour detection, shape matching) whereas, for detecting objects (e.g. vehicles, people, or animals) in a complex scenario, it may be better to choose more complex approaches such as deep neural networks.

C. Volumetric Mapping

The environment is represented by Voxblox [23] map, which incrementally builds *Euclidean Signed Distance Fields* (ESDFs) from *Truncated Signed Distance Fields* (TSDFs) in a voxel-hashing approach, allowing insertions and lookups of voxels in *O(1)* time complexity. We extend the common Voxblox data structure to include semantic knowledge, hence every voxel contains robot traversability information as well as a label corresponding to the detected class as output of the object detection module, and the associated confidence score.

In the literature, in order to map a detected object, the commonly used approach consists in projecting the bounding box edges (as pixels) through the map by ray-casting each pixel and marking the occupied voxels that the rays reach with the same label of the corresponding box. However, depending either on the size of the mapped space or on the geometric characteristics of the detected object, this kind of solution is prone to errors. More in detail, if an object is detected beyond the mapped space, then the traced rays will not find any occupied voxels and hence the information related to the detected object will be lost. Furthermore, depending on the object geometry, i.e. either if the object is concave or the bounding box doesn't fit the object accurately, some traced rays may not hit the voxels occupied by the object, and end up beyond the mapped space or, even worse, in other parts of the map mismatching the semantic annotation.

To overcome these problems, we propose what we called the *Wavefront Raycast* method which allows to mark both the occupied voxels and the projected voxels belonging to the frontier of the map with the bounding box label. In particular, we defined as *surface voxels* the occupied voxels within the mapped space and as *map-frontier voxels* the projected voxels belonging to the frontier of the map. Formally, a *surface voxel* is an occupied voxel with a neighbouring unoccupied voxel, and a *map-frontier voxel* is an unknown voxel with a neighbor unoccupied voxel located on the boundary of the actual map. The former allows locating the detected object in the space, the latter gives information about the direction along which an object outside the current map boundaries should be located.

Instead of tracing rays through all the bounding box, the method projects a single ray in the center of the bounding box. The idea that led to the definition of such a method, arises from the observation that usually the central pixel of a bounding box has a higher probability than peripheral pixels to belong to the item class which the bounding box refers to. Hence, a ray that is cast from the central pixel of the bounding box has a high chance to cross either the occupied voxel belonging to that object or a frontier voxel that points towards the distant object. Therefore, a wavefront mechanism is started and propagated with a *breadth-first* search from the projected box center. The complete method is shown in Algorithm 1.

It starts with the initialization of the queue of voxels to visit $queue_V$, inserting the projected box center v_{center}. For all the voxels popped from that queue, the voxels in the 26-connectivity are marked as visited and, if a non-visited voxel of the same kind of v_{center} (surface voxel or map frontier voxel) is found, if the re-projection of its center in the focal plane (using the camera matrix formula) falls into the bounding box, it is added to $queue_V$. Then, given the voxel center position vector $\vec{v}_W$, in world frame, the transformation T_{CW} from camera frame to world frame and the camera matrix K, the re-projection finds the pixel (u,v) corresponding to a voxel using (1).

$$\begin{cases} \vec{v}_C = T_{CW}\ \vec{v}_W \\ \vec{p} = K\ \vec{v}_C \\ u = {}^{p_x}/_{p_z} \\ v = {}^{p_y}/_{p_z} \end{cases} \tag{1}$$

In the case that v_{center} is a surface voxel, to add the voxel into the queue, an additional condition needs to be met, i.e. the surface gradient must be similar to the one computed

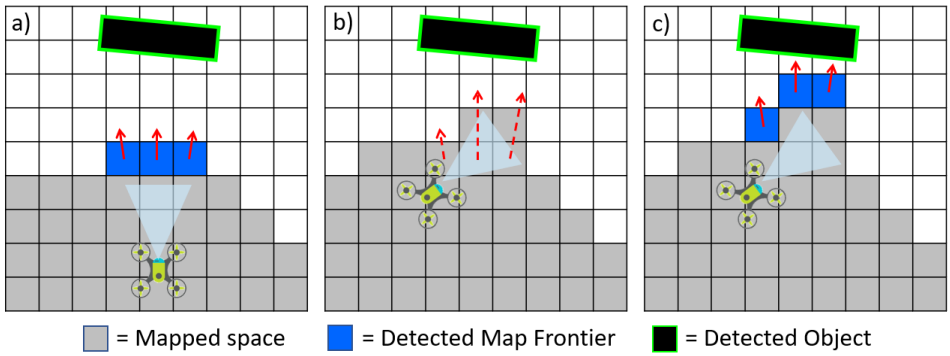


Fig. 2: *Frontier Map Voxels update. In (a) the object is detected and, since the object volume is not mapped, only the corresponding map frontier voxels are labeled, and the robot-voxel center vectors are stored. Then (b) the map frontier voxels get observed, and the projection toward the new map frontier happens and the voxels found are marked with the same data (label, confidence and direction vector) of the previously labeled map frontier voxels.*

in the center of v_{center}. This avoids propagating the search through discontinuities, for example, the floor on which the object stands or on the backside of the object itself. The gradient of a voxel is computed as an interpolation of the ESDF values of its neighboring voxels. Given the gradient $\vec{g}_w$, computed on voxel w and $\vec{g}_{center}$, computed on v_{center}, the gradient angular similarity condition is met if

$$\vec{g}_{center} \cdot \vec{g}_w < g_{TH} \tag{2}$$

is verified, where g_{TH} is a threshold constant.

Once the set of voxels resulting from the ray cast have been identified, the detected label is assigned to them, and the confidences are updated according to the current detection confidence conf_{det}. In particular, if that voxel has not already a label, the detection confidence is assigned to it, otherwise it is updated by the factor $\alpha(\text{conf}_{det} - \text{conf}_{voxel})$, where α is a fixed constant, chosen by design.

Moreover, defining as $\vec{c}_m$ the center of the voxel m found by the raycasting operation, the vector $\vec{c}_m$ - $\vec{x}(t_{det})$ is stored in the voxel itself. This vector represents the direction that the robot should keep in order to reach the detected object.

As the exploration evolves, the map changes and gets refined. This may produce a labeled voxel to become unoccupied, and, in that case, it is reset to unlabeled. In the case of a map frontier voxel, this would cause the loss of the direction along which the previously detected object is. To solve this issue, all the map frontier voxels, before being reset, are projected along the direction stored in them, at the previous detection, into a new map frontier voxel, in which the label and confidence of the precedent one are copied, as shown in Fig. 2. This method prevents losing track of the direction of the detected object respecting to the robot, and is useful for the planner to find objects even if they have not been mapped.

Algorithm 1: Wavefront Raycast

Require: $bbox$ detection bounding box
Require: v_{center} voxel resulting from raycast to the $bbox$ center
Require: $queue_V$ used for detecting voxels to label
$queue_V \leftarrow \emptyset$
$result \leftarrow \emptyset$
ENQUEUE($queue_m$, v_{center})
mark v_{center} as $visited$
$\vec{g}_{center} \leftarrow v_{center}$.gradient
while $queue_V$ *is not empty* **do**
  Set $p \leftarrow$ DEQUEUE($queue_V$)
  Add p to $result$
  foreach w *in neighborhood of* p **do**
    if w *is marked as visited* **then**
      continue
    end
    if $bbox$ *contains PROJECTION*(w) **then**
      if *isSURFACEVOXEL*(v_{center}) **then**
        if *isSURFACEVOXEL*(w) **then**
          $\vec{g}_w \leftarrow w$.gradient
          if $\vec{g}_{center} \cdot \vec{g}_w < g_{TH}$ **then**
            ENQUEUE($queue_V$, w)
          end
        end
      else
        if *isMAPFRONTIERVOXEL*(w) **then**
          ENQUEUE($queue_V$, w)
        end
      end
    end
    mark w as visited
  end
end
return $result$

D. Planner Algorithm

As previously mentioned, we used the RRT* RH-NBV planner presented in [18], introducing modifications to the computation of the gains to accomplish a fast object-oriented mission. It is worth pointing out that a different definition of cost and gain influences the robot's behavior, making this planner versatile for many applications.

The gain is computed as the combination of an exploration gain g_{expl} and an object recognition gain g_{obj}. The first gain is responsible for incorporating new information into the occupancy map, whereas the second is responsible for making the robot focus on the detected objects. We call $Vis(\xi)$ the set of voxels visible from the robot configuration ξ. The gain g_{expl} is computed as the number of unobserved voxels visible from the configuration ξ, and g_{obj} is the number of labeled map frontier voxels visible from ξ.

$$g_{expl}(\xi) = \sum_{m \in \text{Vis}(\xi)} \begin{cases} 1 & \text{if } \mathbf{m} \text{ is unknown} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

$$g_{obj}(\xi) = \sum_{m \in \text{Vis}(\xi)} \begin{cases} 1 & \text{if } \mathbf{m} \text{ is map frontier voxel} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

It is worth pointing out that, in (4), only the labeled map frontier voxels contribute to the gain calculation since they are responsible of leading to objects that have not been already localized on the map. In fact, we do not consider the labeled surface voxels because they represent interesting objects that have already been mapped and, for a fast object search operation, it is not worth spending too much time on them.

In (5), the object detection gain and exploration gain are combined through a weighted summation.

$$g(\xi) = g_{expl}(\xi) + \beta g_{obj}(\xi) \tag{5}$$

The parameter β is set to a high value to prioritize the search of objects over the exploration of new parts of the environment.

IV. EXPERIMENTS

We tested the effectiveness of the algorithm by applying it on a Search and Rescue (SAR) mission, both in simulation and real case scenarios. The objective of the mission was to localize people inside a previously unknown environment.

Since the main objective was to look for people, we decided to use a Deep Neural Network (DNN) as the object detection module. Reviewing the state-of-the-art to seek for suitable solutions, YOLO4 [24] appeared to be a reasonable trade-off between detection accuracy and inference speed. It comes pre-trained with MS COCO dataset which has 80 classes, including *people*.

Moreover, since both the PC running the simulations and the MAV's companion computer were based on NVIDIA architectures, tkDNN [25] has been exploited for hardware acceleration. More in detail, tkDNN is a library built on top of cuDNN and TensorRT, allowing it to easily transform a DNN model written in any popular framework (e.g. Darknet, PyTorch, Tensorflow, etc.) into a smaller model specifically optimized for the underlying hardware. Then, we implemented a ROS node using the tkDNN inference library, subscribing to the camera color frames and publishing, as a result, custom messages with bounding boxes information.

A. Algorithm evaluation on simulated environment

To validate the features of our algorithm, we proceed to test its behavior and tune its parameters in a controlled-simulated environment. Moreover, we used the simulated environment as a test-bed to compare our approach with the planner proposed by Dang *et al.* [14] for Autonomous Exploration and simultaneous Object Search (AEOS). We implemented the exploration algorithm they described in their paper and ran it in our simulation stack.

The synthetic environment consists of a $20.0\,\text{m} \times 20.0\,\text{m} \times 2.5\,\text{m}$ *maze*, without roof, with 4 people in it. The position of the people is known and it is used to establish when the algorithm actually localizes a person during the exploration phase. The nominal bounding volume of a person is $0.5\,\text{m} \times 0.33\,\text{m} \times 2.0\,\text{m}$, however, it is enlarged to $1.0\,\text{m} \times 1.0\,\text{m} \times 2.0\,\text{m}$ to take into account inaccuracies coming from the UAV odometry (especially on the xy plane) as well as from the detection module.

It is worth mentioning that this *enlarged* bounding volume does not compromise the effectiveness of people localization for a SAR task. Finally, this *enlarged* bounding volume defines the position ground truth for each person in the map that has been used in the offline analysis to assess when a person was actually detected by the two evaluated algorithms. More in detail, a person is considered to be localized if the number of labeled voxels inside the ground truth bounding volume is above a given threshold n_{vox}.

The UAV has no prior information of the environment and its vertical displacement is limited to $2.25\,\text{m}$ to avoid finding people by overflying the maze's walls. Simulated experiments duration was limited to $15\,\text{min}$ to resemble the average autonomy of a standard UAV system.

We performed 30 simulations for each exploration algorithm due to the stochastic nature of the planners. For each experiment, we saved the map every $30\,\text{s}$ to track the progress of the exploration. And, for each of these *checkpoints*, we counted the number of voxels representing people contained in the map. We used an $n_{\text{vox}} = 20$ because this number of voxels covers more than $50\,\%$ of the volume of a person.

PX4 Software in the Loop (PX4-SITL) is used to test the proposed algorithm. PX4-SITL is based on RotorS simulator [26], a Gazebo-based simulation environment, that provides reliable multirotor models. However, the advantage of using PX4-SITL is that it is designed to be used with the drone's autopilot so that the algorithm can be tested directly with the same topics and wrappers that the real prototype uses.

The drone is equipped with a stereo camera that provides real-time feedback of the simulated environment. This camera is in line with the hardware integrated on the real prototype. In simulation, the depth images are taken directly from the camera. Also, the drone's position is obtained from the odometry of the PX4 simulator instead of using a position estimation algorithm. Depth images and drone's position are not computed during the simulations because, on our prototype, they are directly calculated on the camera hardware.

The simulated experiment parameters are summarized in Table I, while mapper and planner standard parameters are presented in Table II and III, respectively.

Parameter		Value
Camera FoV	$[a_h, a_v]$	80°, 50°
Voxel size	r	0.2 m
Collision Radius		0.4 m
Max. Acceleration		$1.0\,\mathrm{m\,s^{-2}}$
Max. Velocity		$1.0\,\mathrm{m\,s^{-1}}$
Max. Yaw Rate		$1.6\,\mathrm{rad\,s^{-1}}$
Max. Collision Radius		0.5 m
Camera Range	$d_{max}^{planner}$	4.0 m
Edge Length	l_{max}	1.5 m

TABLE I: Parameters used through all the experiments

Parameter		Value
Gradient angular threshold	g_{TH}	0.4
Confidence update factor	α	0.4

TABLE II: Parameters of the mapper

Parameter		Value
Local Sampling Count	n_{local}	10
Local Sampling Radius	r_{local}	1.5 m
Updating Radius	r_{update}	3.0 m
Ray Sub-sampling Factor	f_{sub}	3.0
Discount Factors (AEOS)	λ, γ, τ	0.5, 0.3, 10^5
Discount Factors (Ours)	λ	3.0
Objects Gain Factor	β	3000

TABLE III: Parameters of the planner

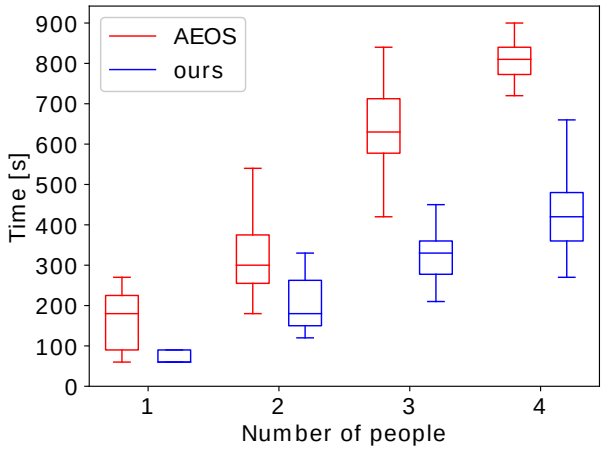
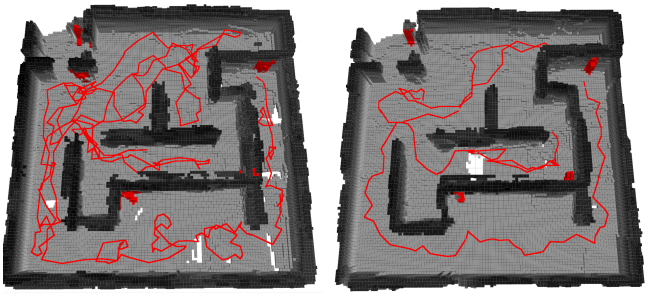


Fig. 3: Comparison of time required to find people in the environment by the two methods.



(a) AEOS (b) Ours

Fig. 4: Best exploration path. Comparison trajectory performed by both exploration algorithms to find the 4 persons hidden in a maze.

All experiments were done in a Docker container running Ubuntu 18.04, ROS Melodic built from source, OpenCV 4.4, CUDA 11.1, cuDNN 8.1 and TensorRT 7.2.2. The hardware platform was a 6 cores Intel(R) Core(TM) `i7-8700K` CPU with 32 GB of RAM and an Nvidia GeForce GTX 1080 GPU.

Fig. 3 depicts a box plot of the time required to find 1, 2, 3 or 4 people in each exploration approach. From this figure, it can be noticed a significant reduction in the average time needed to find a person. Moreover, as the number of people increases, the worst time of our experiments (maximum time) slightly becomes better than the best of AEOS's (minimum time). It is worth noticing that in 21 out of 30 experiments, the AEOS planner finds at most 3 out of 4 people in the maze during the 15 min long simulation.

In Fig. 4, we present a qualitative comparison of the trajectory followed by the drone to find all the people hidden in the maze. The drone's trajectory and detected people's voxels are colored in red, while obstacles (e.g. walls, floor) are colored according to a gray-scale palette depending on the height. In particular, this figure depicts the fastest experiment of each approach until they correctly locate the last person. Both experiments are represented, respectively, by the lowest point in the last couple of box plots shown in Fig. 3 (abscissa value equal to 4 people), where it can be noticed a significant reduction of time to accomplish the mission. From a quantitative point of view, the time spent by the drone using the AEOS algorithm was more than 10 min, while our algorithm took less than 5 min.

B. Algorithm testing on real platform

After testing on simulations, we performed functional validation of the entire software stack on a real drone platform. In this case no external systems were used, since all the computation related to environment perception, motion planning, and control run on-board the system.

The hardware platform consists of a custom-built quadrotor that relies on Pixhawk 4 board for low-level attitude rate control and on NVIDIA Jetson XAVIER NX for high-level software modules, i.e. navigation, exploration, mapping and detection algorithms. To account for the perception task, the system integrates two front-facing cameras, and in particular, the Intel Realsense T265 camera to obtain the drone's local position and motion estimation, and the Intel Realsense D435i camera to obtain the depth and the RGB information. The height above the ground is measured by a Garmin Lidar Lite v3. The platform's weight is 1.5 kg and its dimensions are $0.45\,\mathrm{m} \times 0.45\,\mathrm{m} \times 0.20\,\mathrm{m}$.

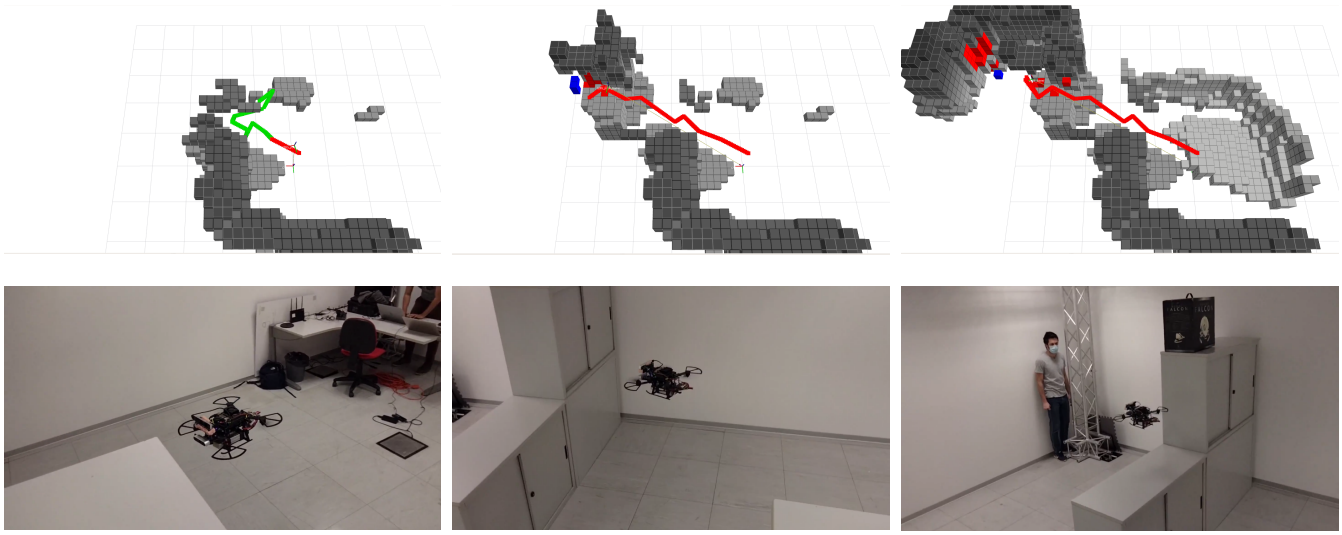


Fig. 5: Photo-sequence of a UAV using the proposed object-oriented exploration algorithm in a real-life experiment. The upper sequence shows, the system's trajectory and mapping, while the lower sequence shows the corresponding moment from an external point of view.

The experiment consisted in the exploration of a small room, where there were obstacles, with no prior information related to maps and/or geometry. We only set the exploration boundaries area to be in line with the actual room size, i.e. $5.0\,\mathrm{m} \times 3.0\,\mathrm{m} \times 2.0\,\mathrm{m}$. The mission goal was to look for a person lost in the room.

Considering the space to be explored is small, the camera range $d_{max}^{planner}$ is limited to $2.0\,\mathrm{m}$. In this way, it is possible to appreciate the algorithm's features instead of the camera's range capacity.

Fig. 5 depicts some snapshots of the experiment. The proposed method adjusts its planned trajectory (in green) to rapidly traverse the scene once the object of interest is distinguished. The picture follows the same color representation as Fig. 4. Moreover, in this photo-sequence it is possible to appreciate frontier-map voxels (in blue). It is clearly highlighted that the algorithm skips mapping a large part of the environment by giving a bigger priority to the object of interest.

V. CONCLUSIONS

In this paper, we present an efficient object-oriented exploration algorithm for UAVs in unknown environments. The algorithm exploits map annotations from a DNN object detector to recognize objects in the scene and steer the exploration towards them. The proposed algorithm has been extensively tested in synthetic environment based on ROS/Gazebo framework. In order to assess the effectiveness of the presented solution a comparison with the most advanced method at the state of the art has been performed in a SAR-like scenario. Moreover, tests on a real MAV have been done to prove both the feasibility of the proposed solution on a limited resources platform and the capability to map and localize objects of interest in a real-life scenario.

REFERENCES

[1] VH. Nguyen, R. Jenssen, and D. Roverso. Automatic autonomous vision-based power line inspection: A review of current status and the potential role of deep learning. *International Journal of Electrical Power & Energy Systems*, 99:107–120, 2018.

[2] M. Satler, M. Unetti, N. Giordani, C. Avizzano, and P. Tripicchio. Towards an autonomous flying robot for inspections in open and constrained spaces. In *2014 IEEE 11th International Multi-Conference on Systems, Signals & Devices (SSD14)*, pages 1–6. IEEE, 2014.

[3] J. Gu, T. Su, Q. Wang, X. Du, and M. Guizani. Multiple moving targets surveillance based on a cooperative network for multi-uav. *IEEE Communications Magazine*, 56(4):82–89, 2018.

[4] A. Wada, T. Yamashita, M. Maruyama, T. Arai, H. Adachi, and H. Tsuji. A surveillance system using small unmanned aerial vehicle (uav) related technologies. *NEC Technical Journal*, 8(1):68–72, 2015.

[5] A. Murtiyoso and P. Grussenmeyer. Documentation of heritage buildings using close-range uav images: dense matching issues, comparison and case studies. *The Photogrammetric Record*, 32(159):206–229, 2017.

[6] S. Malihi, MJ. Valadan Zoej, and M. Hahn. Large-scale accurate reconstruction of buildings employing point clouds generated from uav imagery. *Remote Sensing*, 10(7):1148, 2018.

[7] P. Tripicchio, M. Satler, G. Dabisias, E. Ruffaldi, and C. Avizzano. Towards smart farming and sustainable agriculture with drones. In *2015 International Conference on Intelligent Environments*, pages 140–143. IEEE, 2015.

[8] P. Tripicchio, M. Satler, M. Unetti, and C. Avizzano. Confined spaces industrial inspection with micro aerial vehicles and laser range finder localization. *International Journal of Micro Air Vehicles*, 10(2):207–224, 2018.

[9] K. Yakushiji, H. Fujita, M. Murata, N. Hiroi, Y. Hamabe, and F. Yakushiji. Short-range transportation using unmanned aerial vehicles (uavs) during disasters in japan. *Drones*, 4(4):68, 2020.

[10] Y. Shen, D. Guo, F. Long, L. A. Mateos, H. Ding, Z. Xiu, R. Hellman, A. King, S. Chen, C. Zhang, and H. Tan. Robots under covid-19 pandemic: A comprehensive survey. *IEEE Access*, 9:1590–1615, 2021.

[11] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart. Receding horizon "next-best-view" planner for 3d exploration. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1462–1468, 2016.

[12] M. Juliá, A. Gil, and O. Reinoso. A comparison of path planning strategies for autonomous exploration and mapping of unknown environments. *Autonomous Robots*, 33(4):427–444, 2012.

[13] L. Lu, C. Redondo, and P. Campoy. Optimal Frontier-Based Autonomous Exploration in Unconstructed Environment Using RGB-D Sensor. *Sensors (Switzerland)*, 20(22):1–16, 2020.

[14] T. Dang, C. Papachristos, and K. Alexis. Autonomous exploration and simultaneous object search using aerial robots. In *2018 IEEE Aerospace Conference*, pages 1–7, 2018.

[15] T. Dang, C. Papachristos, and K. Alexis. Visual saliency-aware receding horizon autonomous exploration with application to aerial robotics. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2526–2533, 2018.

[16] C. Papachristos, S. Khattak, and K. Alexis. Uncertainty-aware receding horizon exploration and mapping using aerial robots. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4568–4575, 2017.

[17] M. Selin, M. Tiger, D. Duberg, F. Heintz, and P. Jensfelt. Efficient autonomous exploration planning of large-scale 3-d environments. *IEEE Robotics and Automation Letters*, 4(2):1699–1706, 2019.

[18] 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 Robotics and Automation Letters*, 5(2):1500–1507, 2020.

[19] A. Hornung, K. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 2013. Software available at `http://octomap.github.com`.

[20] R. Ashour, T. Taha, J. Manuel, M. Dias, L. Seneviratne, and N. Almoosa. Exploration for Object Mapping Guided by Environmental Semantics Using UAVs. 2020.

[21] M. Petrlík, T. Báča, D. Heřt, M. Vrba, T. Krajník, and M. Saska. A robust uav system for operations in a constrained environment. *IEEE Robotics and Automation Letters*, 5(2):2169–2176, 2020.

[22] T. Dang, F. Mascarich, S. Khattak, H. Nguyen, H. Nguyen, S. Hirsh, R. Reinhart, C. Papachristos, and K. Alexis. Autonomous search for underground mine rescue using aerial robots. In *2020 IEEE Aerospace Conference*, pages 1–8, 2020.

[23] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto. Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[24] A. Bochkovskiy, CY. Wang, and HYM. Liao. Yolov4: Optimal speed and accuracy of object detection, 2020.

[25] M. Verucchi, G. Brilli, D. Sapienza, M. Verasani, M. Arena, F. Gatti, A. Capotondi, R. Cavicchioli, M. Bertogna, and M. Solieri. A systematic assessment of embedded neural networks for object detection. In *25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, pages 937–944. IEEE, 2020.

[26] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart. Robot operating system (ros): The complete reference (volume 1). pages 595–625, Cham, 2016. Springer International Publishing.