2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Benchmarking and Experimental Validation of a Real-Time Multi-Robot Hybrid Coverage Algorithm for Known Environments

Lucas Wälti and Alcherio Martinoli

***Abstract*— In the context of asset inspection, the size of the environment to be covered can be large. Mobile robotic systems are capable of acquiring more extensive data than static sensors, but the capacity of the robotic platform used can be limited by its autonomy and sensing capabilities. This is why multi-robot systems are interesting in such applications. However, scaling up to larger robot team sizes requires coordination among robots to be carried out efficiently. In this work, we investigate a hybrid coordination strategy with a team of micro-aerial vehicles, where a ground station centrally assigns tasks in real time to the robots, and the robots distributively coordinate their trajectories to carry out the coverage of a known asset. In particular, we perform the benchmarking and experimental validation of such a strategy. Several variants of the strategy are implemented by adapting existing state-of-the-art solutions to this context. Extensive simulation experiments are carried out in various environments to benchmark each variant and evaluate how their performance scales with the robot team size. The results show that the strategy scales well for larger robot teams, thanks to its efficient task generation process. Notably, despite its relatively simple but efficient task generation technique, it outperforms or is comparable to other methods employing more complex schemes (such as information gain or frontiers). Finally, we validated the proposed strategy with teams of up to three robots in physical experiments.**

I. Introduction

Multi-robot systems, composed in particular of rotary-wing Micro Aerial Vehicles (MAVs), present exciting qualities for inspection applications, where large-scale objects are often encountered, and the coverage requirements go beyond the capabilities of a single robot (due for instance to limited battery life or embedded sensors limitations). A common approach to deploy a team of MAVs for the coverage of a given asset is to use Coverage Path Planning (CPP) algorithms [1], where CPP is the process of covering an asset or environment by determining a set of locations to visit. In the literature, several works leverage the knowledge of the environment to generate viewpoints and solve some version of the Traveling Salesperson Problem (TSP) to generate paths offline for a single or multiple robots to follow [2], [3], [4]. Thus, these fall into the category of model-based approaches. However, model-free approaches remove this assumption, which requires the environment to be explored and discovered. They can be further subdivided into several subcategories, namely, *frontier-based* (where the interface between known and unknown space is identified and explored [5], [6]), *volume-based* (where unknown volumes in the environment are used to drive the exploration, most commonly implemented as Next-Best-View (NBV) methods [7], [8]) and *surface-based* (can be seen as a subset of the volume-based approaches, where surfaces are explicitly represented, e.g., as Truncated Signed Distance Field [9], [10]). Applying such approaches to multi-robot systems is non-trivial, as sharing map information can represent large amount of data and ensuring synchronization among the robot team can be challenging. More specifically on multi-robot solutions, most of them require offline computation and optimization of the coverage paths and do not accommodate online replanning in case of unforeseen events [4], [11], [12]. Some distributed online coverage approaches have been proposed [13], but they require sharing dense map information between robots and require extensive onboard computations. More generally, in the context of inspection tasks, it is reasonable to assume knowledge about the environment is available, as such tasks are often carried out repetitively throughout the lifetime of the asset. For this reason, model-based methods can be employed to leverage this knowledge and drive coverage more efficiently.

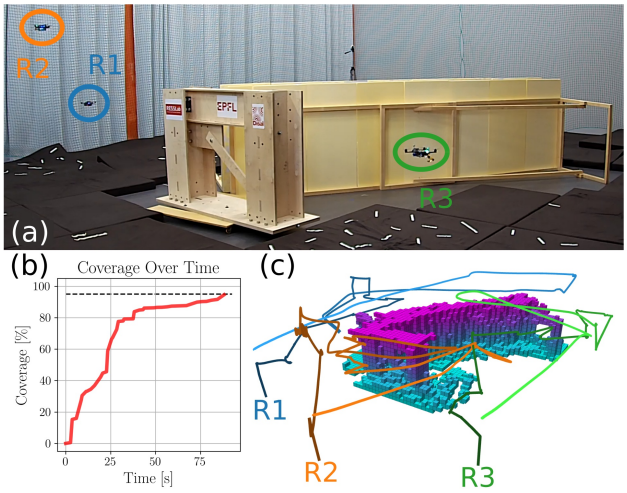


Fig. 1: **(a)** Physical experiment where a team of three MAVs perform the coverage of the asset. **(b)** Asset coverage over time (with a targeted coverage completion of 95%). **(c)** Map built from the asset at 10 cm resolution, along with the resulting trajectories of the three robots with time encoded by a darker to brighter color gradient.

In our previous work [14], we proposed a real-time model-based multi-robot hybrid coverage scheme that distributes responsibilities between a ground station serving as centralized task assigner, and the robot team, responsible for coordinat-

Both authors are with the Distributed Intelligent Systems and Algorithms Laboratory (DISAL), Swiss Federal Institute of Technology (EPFL), 1015 Lausanne, Switzerland. Corresponding e-mail: lucas.walti@epfl.ch

Additional information about the research and accompanying material are available here: https://www.epfl.ch/labs/disal/research/aerialdistributedinspection/

ing individual trajectories in a distributed fashion. Although model-based approaches often precompute the paths that robots will execute [2], [3], [4], our previous approach lets the robots compute their trajectories in real time, with the advantage of offering robustness against individual robot failures and unforeseen events, as well as allowing dynamic robot team sizes. Furthermore, leveraging the knowledge of the environment, as in model-based methods, allows the robots to carry out coverage more efficiently without requiring stochastic exploration of the volume, as opposed to some model-free methods. However, task generation must be fast to compute, as it is performed sequentially for each robot in a centralized way. Slow task generation will result in scalability challenges as the number of robots increases, which we addressed using the PH-tree data structure [15].

We introduce several technical improvements to our original pipeline in [14] to make it applicable to physical real-time multi-robot systems:

- We bring some adjustments to the pipeline to improve coordination among robots and overall reliability, including adjustments to the task generation, more efficient ray-tracing, trajectory collision checks with captured point cloud, and a deadlock handling mechanism.
- We generalize the observability check introduced in [14], allowing us to apply the method to more varied geometries (see simulated experiments).

We extend the experimental validation of the method:

- We adapt several state-of-the-art approaches to the framework in [14] for comparison.
- We perform extensive simulation tests and compare the performance of the method with other state-of-the-art approaches, focusing in particular on the scalability with respect to larger robot team sizes.
- We perform physical experiments with a team of up to three robots to validate the applicability of the method to real robots.

The remainder of this manuscript is structured as follows. We recall our previous method [14] and review the improvements introduced in this work, as well as the state-of-the-art methods implemented, in Section II, describe the experimental setup in Section III, report on the obtained results in Section IV, and discuss them in Section V.

II. Methods

We first recall our original method proposed in [14] and then introduce several technical improvements that enable a safer deployment of the method to real robots. We then introduce several variants of the method, adapted from various state-of-the-art approaches, which we will use for comparison.

A. Original Method

A ground station generates observation tasks centrally and assigns them to a team of MAVs. The environment is represented as a fixed resolution discrete voxel grid map. An avoidance map is derived from it by expanding the obstacles

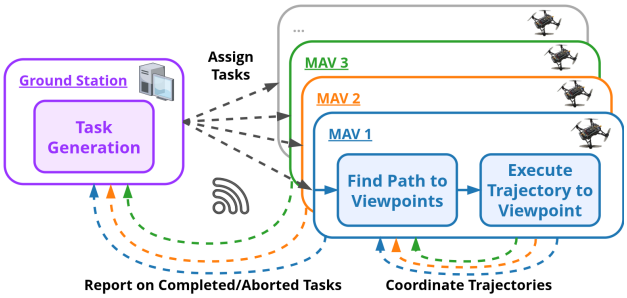


Fig. 2: Pipeline overview with centralized task generation and assignment to each robot, and distributed coordination for the execution of the tasks by the robot team.

and is used by the ground station and the robots to enforce a minimal distance between the robots and the obstacles in the environment. An important design element is that the map is shared with each robots before the mission start and is no longer exchanged afterwards. Regarding sensing data, they can be either broadcasted in real-time, or stored onboard each robot during the mission, depending on the needs of the mission and available communication bandwidth. In fact, the ground station only needs to know the position and heading at which an observation was carried out to deduce which regions were observed, given known intrinsic and extrinsic parameters of the sensors onboard the MAVs. A PH-tree [15] is used to reference uncovered voxels in the Region Of Interest (ROI) and query them efficiently by the ground station, offering efficient k-Nearest-Neighbors (KNN) and box query functionalities. Each voxel in the ROI stores information about its observation status, as well as the status of its individual faces, which are used to select voxel and face pairs to be observed during the task generation process. More concretely, tasks are constructed by performing a KNN query centered at the current robot position on the voxels within the ROI that have not been observed so far. For each voxel candidate returned, a face is selected based on its orientation relative to the robot's and observation status (i.e., whether it is observed or obstructed), and a set of viewpoints with a line of sight to the face is generated. The resulting voxel, face and viewpoints triplet, to be sent as a task to the robot, is selected based on simple distance heuristics relative to the robot's position and heading. This process is summarized in Alg. 1. The robots coordinate their trajectories to perform their assigned task and report to the ground station when a task is completed or aborted, communicating as well their position and heading at the time of completion. With this information, the ground station systematically updates the observation status of the map and generates a new task for each robot reporting a completed or aborted task. Robots continuously share with each other their current position and planned path, which prevents the generation of potentially unsafe intersecting trajectories. The overall architecture is illustrated in Fig. 2.

B. Pipeline Improvements

We propose the following improvements to the pipeline in [14]. While these changes do not fundamentally change the pipeline's behavior, they significantly improve its reliability.

Algorithm 1 Task generation algorithm from [14]. We add in this work the condition at line 7.

Require: $\mathbf{p}_r \in \mathbb{R}^3$ *(Robot position)* , $\theta_r \in \mathbb{R}$ *(Robot heading)* , PH-tree *(Initialized PH-tree)* , $\mathbf{C}$ *(cache of viewpoints for {voxel, face} pairs)*

▷ ***Voxel and face selection***

1: $\mathbf{V} \leftarrow [\,]$ ▷ *empty {voxel, face} pairs priority queue*
2: **for all** $v \in$ PH-tree.kNN($\mathbf{p}_r$) **do** ▷ *voxel KNN query*
3: $f \leftarrow$ SELECT_FACE(v, $\mathbf{p}_r$)
4: $\mathbf{V}$.push($\{v,f\}$) ▷ *Add {voxel, face} pair to priority queue*

▷ ***Sorted task list generation***

5: $\mathbf{T} \leftarrow [\,]$ ▷ *Empty list of tasks*
6: **for all** $\{v,f\} \in \mathbf{V}$ **do** ▷ *Generate viewpoints for each pair*
7: **if** $\{v,f\}$ is currently assigned **then**
8: **continue**
9: **if** $\mathbf{C}$.contains($\{v,f\}$) **then**
10: $\mathbf{W} \leftarrow \mathbf{C}$.get($\{v,f\}$) ▷ *Use cached viewpoints*
11: **else**
12: $\mathbf{W} \leftarrow$ VIEWPOINTS_GENERATION(v,f)
13: $\mathbf{C}$.push($\{v,f\}$, $\mathbf{W}$) ▷ *Cache the viewpoints* $\mathbf{W}$
14: $\mathbf{T}$.append($\tau(v,f,\mathbf{W})$) ▷ *Append the $\{v,f\}$ pair and viewpoints* $\mathbf{W}$ *as a new task* τ *to* $\mathbf{T}$
15: $\mathbf{T} \leftarrow$ sort($\mathbf{T}$) ▷ *Sort tasks according to effective viewpoints* $\mathbf{W}$*, robot position* $\mathbf{p}_r$ *and heading* θ_r

▷ ***Task assignment***

16: **for all** $\tau \in \mathbf{T}$ **do**
17: **if** $\tau \notin \mathbf{B}_r$ **then** ▷ *Task* τ *is not in the robot's task buffer* $\mathbf{B}_r$
18: $\mathbf{B}_r$.push(τ) ▷ *Add the task to the robot's task buffer*
19: Send task τ to robot r **and** exit
20: $\mathbf{B}_r$.clear() ▷ *Empty the robot's task buffer*
21: Send first task τ in $\mathbf{T}$ to robot r **and** exit

1) Task Generation: When originally performing the KNN query, the {voxel, face} pairs are sorted in the priority queue in Alg. 1, line 4, according to the distance from the voxel to the current position of the robot, and the alignment of the current robot heading with the normal direction of the face. However, this overlooks the fact that a voxel located further away could yield viewpoints closer to the current position of the robot, compared to the closest voxels. For this reason, we compute the theoretical viewpoint at the desired observation distance directly facing the voxel face and apply the same heuristic to the viewpoint. To keep the evaluation efficient, we do not perform any collision or line-of-sight check on the viewpoint and assume that it is valid.

Finally, when compiling tasks from the {voxel, face} pairs, we only create a task if the pair is currently not assigned to any other robot. The corresponding check is reported in Alg. 1, line 7. This simple yet effective check allows us to minimize redundant task assignments and reduce the risk of assigning conflicting tasks to robots.

2) Efficient Ray Tracing: Instead of the naive ray tracing employed in [14] that traces a predetermined series of rays, we leverage the fact that the environment is known and use the PH-tree's box query to only trace the necessary rays from the unobserved voxels within the robot's FoV. Concretely, we perform a box query centered on the robot's position within an axis-aligned cube with edge length equal to twice the sensor range, discard voxels within this box that do not fall within the sensor's FoV, and trace rays from the remaining voxels to the robot's position to check for line of sight. This allows us to efficiently update the map upon task completion before reassigning a new task to the robot.

3) Trajectory Collision Check: The robots use the point cloud data captured by their ToF camera to assess whether their planned trajectory is in collision with the environment (due to inaccurate environment modeling, communication failure between robots, or drift from the intended path), requiring to recompute a safe trajectory. We filter the raw point cloud of the sensor using a voxel grid filter to enforce an upper bound on the point density and filter out sensor noise, sample the trajectory from the current position at five evenly spaced locations in time over a three-second horizon, and fit a second-order polynomial. Locally fitting the trajectory in this way allows us to efficiently compute the distance between any point and the polynomial in closed form and efficiently detect collisions.

4) Deadlock Handling: Some MAVs may end up in a deadlock, where they fail to complete a task for the same reason several times in a row (typically due to another robot blocking the assigned goal). We implement a simple deadlock resolution method, where if a given robot repeatedly reports aborted tasks for the same reason (we set the threshold to five), it is assigned a random task in the environment.

5) Observability Check: We extend the offline observability check proposed in [14], used to preprocess the ROI and identify observable voxels within it. In the original approach, the observability check consists in sampling poses along the surfaces of the environment's bounding box, verifying which voxels are visible by performing KNN queries on the PH-tree from these locations, and checking for a line of sight to the returned voxels. In our new version, we grow an RRT* tree within the avoidance map from a feasible seed position to ensure the validity of the sampled poses, but also their connectivity, thus ensuring that they are reachable. This approach allows us to identify observable voxels for any environment geometry. During the line-of-sight check, we use the camera model of the sensor to assess which voxels can fall within its FoV. We assume that the MAVs can rotate at 360° and thus we only check the sensor's vertical FoV and range. We illustrate the end result of this process in Fig. 3 in the apartment environment in Fig. 5c.

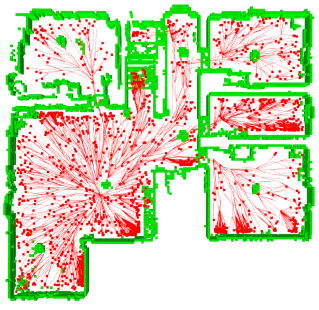


Fig. 3: Observability check example, where the RRT* algorithm is used to grow a tree (shown in red) from a feasible location in the environment to be inspected of Fig. 5c and discover observable voxels (shown in green) within the ROI, given obstacle and sensor characteristics constraints.

C. Benchmarking

We benchmark our coverage strategy with alternative state-of-the-art variants. We fully implement each alternative coverage method and do not rely on open-source implementations to ensure proper and fair integration within our framework, where tasks are generated centrally by the ground station. Thus, each method presented here alters how the task generation process is handled on the ground station, but does not alter the robots behavior. We selected the methods listed hereafter, including NBV and Frontiers approaches. Although these approaches were initially designed for the exploration of unknown environments, they offer relevant mechanisms for the coverage of known assets as well. Frontiers will enforce a more progressive coverage, while NBV should produce more optimal viewpoints since information gain is considered. For a fairer comparison, we introduce a version of NBV where the knowledge of the environment is leveraged to achieve more efficient viewpoint sampling (see Section II-C.4).

1) PH-Tree + Distance Heuristics (DH, this method): This constitutes the base method, as described in Section II-A, where the PH-tree is used to perform KNN voxel queries around the robots' location to generate tasks, using simple distance heuristics as per [14].

2) PH-tree + Information Gain (IG): The method is identical to the original one, but instead of using distance heuristics between the closest viewpoint and the robot's position and heading when sorting tasks in Alg. 1, line 15, we calculate and use the information gain a {voxel, face} pair would yield if selected, as the number of uncovered voxels [7] that would be visible from the corresponding viewpoint. The tasks are thus sorted according to their information gain, instead of the default distance heuristic.

3) Next-Best-View (NBV): This follows the algorithm proposed by Bircher et al. [7]. Here, an RRT* tree is grown from the current location of the robot. Up to 50 nodes are added to the tree. Once the tree is created, the headings are uniformly sampled for each node. Then, the information gain for each node is calculated (as the number of uncovered voxels visible from that node's position and heading), to which is added the node's parent's gain in a cumulative fashion. The branch with the highest gain is then selected, and the first node is sent as a target to the robot. Fig. 4 shows a resulting tree with the best branch highlighted in red. As in [7], we retain the best branch and use it to initialize the tree in the next task generation and recompute the gains of the preserved branch before regrowing the tree. In case of deadlock, all branches in the tree are discarded.

4) NBV + Assisted Heading (AH): The same approach is used as in the *NBV* method, but the heading is computed from the direction towards the weighted center of mass of nearby voxels, using weights inversely proportional to the distance between the robot and the voxels. The nearby voxels are queried using the PH-tree box query, similarly to the query proposed for the ray-tracing approach described in Section II-B.2. We apply the following exponential decay weighting scheme:

$$w_v = \exp\left(-\|\mathbf{c}_v - \mathbf{p}_r\|\right) \tag{1}$$

where the voxel v is assigned the weight w_v, given its center position $\mathbf{c}_v \in \mathbb{R}^3$ and the position $\mathbf{p}_r \in \mathbb{R}^3$ of robot r.

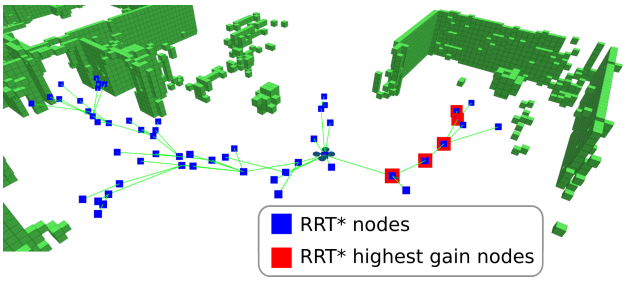


Fig. 4: RRT* tree in the *NBV* method. Edges (in green) and nodes (in blue) are constructed for each task generation and the branch with the highest gain spanning from the robot position is selected (red nodes). The remaining unobserved voxels in the ROI are shown in green.

5) Frontiers: We follow the frontier principle proposed in [5]. We keep track of the subset of unobserved voxels directly adjacent to observed voxels and restrict the PH-tree KNN queries in Alg. 1, line 2 to this set, performing task selection as in the *PH-tree+DH* method. If this set is empty (e.g., due to discontinuous regions), we default to the global KNN queries of the *PH-tree+DH* method until the frontier set is non-empty.

D. Metrics

To analyze the performance of the system, we rely on the following metrics.

- *Task Generation Duration* [s]: The total time elapsed while executing the task generation in Alg. 1.
- *Observation Efficiency* [voxels]: Number of newly observed voxels for each completed task.
- *Mission Duration* [min]: The total elapsed time between the first and last tasks performed.
- *Robot Trajectory Length* [m]: The length of the trajectory performed by a robot.
- *Activity Rate* [%]: The ratio of the time spent processing and performing tasks by a robot to the total duration of the mission.

Except for the *Task Generation Duration* and *Mission Duration* metrics, these metrics are computed individually for each robot and aggregated when presenting the results.

III. Experimental Setup

We perform extensive simulation experiments to benchmark the methods mentioned above and validate our approach with up to three real robots in physical experiments. We use the ROS1 middleware and employ its capabilities for inter-robot and robot-ground station communications. We adopt a single-master approach, where the ground station serves as such.

A. Simulated Experiments

We run our simulated experiments on a desktop computer equipped with a sixteen cores 11th Gen Intel Core i9-11900

CPU. We use the open-source, high-fidelity simulator Webots [16] to perform our experiments, which provides realistic rigid-body physics, sensor emulation, and a simple propeller model. We run a custom autopilot controller for each robot (see the modeled robot in Fig. 5d), emulating the PX4 control stack and its MAVROS interface. Thus, the code running in the simulation is identical to the one running on board the real drones.

For our experiments, we used three different environments, shown in Fig. 5. The tank wagon in Fig. 5a is the simplest environment with a nearly convex shape. We define a $6.6 \times 3.3 \times 4.2$ m^3 ROI at a resolution of 10 cm with 12'081 voxels, corresponding to half of the tank. The transmission tower in Fig. 5b represents a slightly more challenging environment due to its more complex overall geometry. We define a $10.2 \times 1.9 \times 9.7$ m^3 ROI starting at a height of 12 m corresponding to the upper part of the tower at a resolution of 10 cm with 10'718 voxels. Finally, the apartment environment in Fig. 5c is the most challenging, with cluttered rooms and narrow passages. The ROI spans $12.7 \times 13.4 \times 1$ m^3 at a height of 1 m above the apartment floor. We used a resolution of 10 cm with 28'773 voxels.

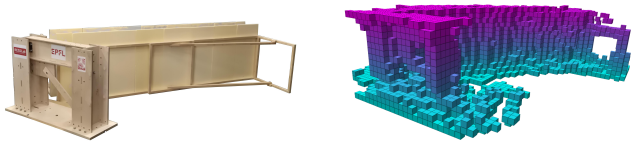
We tested the five methods introduced in Section II-C in each environment. Furthermore, we evaluated them with different robot team sizes, specifically, teams of 1, 3, 6 and 10 robots. Finally, we performed five trials for each combination of environment, method, and number of robots, resulting in a total of $5 \times 3 \times 5 \times 4 = 300$ simulation runs. We stop each simulation run when the coverage reaches 90%, as the last coverage percentages take significantly longer to execute due to the sparsity and spread of the remaining unobserved voxels in the ROI.

In all experiments, robots start grouped from the same location nearby or within the asset, and the *PH-tree+DH* method is used to generate the first task of each robot, so that all methods start with a comparable configuration in each run. Although we do not modify the initial conditions of the experiments, each experimental run yields slightly different results even for the same environment, team size, and method, as several independent processes run concurrently during a simulation, and thus no hard synchronicity is enforced.

B. Physical Experiments

We perform physical experiments in our flying arena of dimensions $12 \times 10 \times 5$ m^3, equipped with a Motion Capture System (MCS), produced by Motion Analysis, providing millimeter-level tracking accuracy. We use a laptop as ground station, with an eight-core Intel Core i7-10510U CPU, serving as ROS master. The ground station uses a wired connection to the modem that delivers the local network, and the robots use a 5GHz Wi-Fi connection. We perform experiments with teams of one, two, and three robots, using our method, where the robot team must cover an asset located in the middle of the flying arena and reach a coverage of 95%. We perform three experiments for each robot team size, for a total of nine experiments (see Fig. 1 showing one of the experiments performed).

1) Real Asset: We use wooden structures, displayed in Fig. 6a, in our flying arena to emulate an asset, occupying a $5 \times 4.9 \times 1.4$ m^3 volume. We map the asset by manually flying one of the MAVs shown in Fig. 7 and capturing point cloud data generated by the onboard ToF camera. We accurately track the pose of the robot with our MCS to transform the measured point clouds into the frame of the MCS. We model occupancy following the log-odds occupancy formulation of [17] and perform offline a denoising step by removing any voxel with less than two direct neighbors, resulting in the map shown in Fig. 6b.



(a) Real asset, constituted of various wooden structures.

(b) Captured map at 10 cm resolution.

Fig. 6: Structure used for real world experiments and the resulting map.

2) Micro Aerial Vehicles: We use the Starling1 platform produced by ModalAI Inc., equipped with a VOXL-Flight board fitted with a Snapdragon 821 quad-core CPU and a Flight Core PX4 flight controller [18]. It spans 190 mm diagonally and weighs slightly more than 300 g. The MAV runs the PX4 autopilot and is interfaced over ROS1 using MAVROS. State estimation is performed by onboard monocular VIO, using a 45° downward looking global shutter wide-angle camera, running Qualcomm's proprietary VIO solution mvVISLAM. Alternatively, the VIO estimate can be substituted with the pose measured by our MCS for better state stability. The MAV is equipped with a forward-facing ToF camera with horizontal and vertical FoV of, respectively, 85.1° and 106.5°, able to measure distances in the range $[0.5, 4]$ m.

Fig. 7: Team of three Starling1 MAVs used for the physical experiments.

3) State Estimation: We use two modalities for on-board state estimation. When performing experiments with one or two robots, we use the VIO estimate produced by the MAVs. This estimate has its origin set to where it is initialized and is used by the autopilot for stabilization. Thus, we use the MCS to localize this origin and express the MAV's pose in the arena, allowing us to transform poses between the odometry frame and the frame in which the environment is expressed. Alternatively, when using three robots, we disable the VIO pipeline and directly fuse the MCS-tracked pose into the onboard state estimate. This was necessary as the VIO estimate became too unreliable when running concurrently with the planning logic (due to the computational load

(a) Tank wagon environment, with roughly convex topology.

(b) Tower environment, with highly non-convex but easily traversable topology.

(c) Apartment environment, difficult to navigate due to clutter and narrow passages between rooms.

(d) Simulated Starling MAV in the Webots simulator [16].

Fig. 5: **(a)-(c)** Simulation environments in increasing order of complexity and **(d)** simulated MAV model.

induced by the increased complexity in navigation linked to more robots generating more clutter in the environment and requiring more complex navigation paths to be found).

IV. Results

In this section, we first present the results from the simulated experiments and then follow up with the results from the physical experiments.

A. Simulation Experiments

We first gather the *Task Generation Durations* from the 300 experiments and report them as violin plots in Fig. 8 for each method, with median, min, and max values. We see that the *PH-tree+DH* (ours) and *Frontiers* methods outperforms the other methods, with a median generation time of only 82 and 66 ms, respectively. Furthermore, the worst-case durations for these methods is comparable to the median generation time of the *NBV* methods.

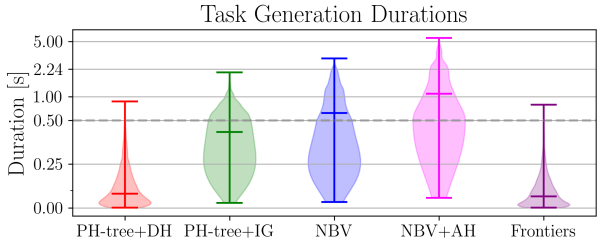


Fig. 8: Task generation durations for each method in seconds, with reported median, min and max values. A linear scale is applied within the range $[0, 0.5]$ s and is made logarithmic for durations above 0.5 s (interface marked by the dashed line).

We then examine the *Observation Efficiency* metric of the methods in the first row of Fig. 9. All methods show comparable performance. However, the only method that tends to outperform all others in each environment is the *PH-tree+IG* method. The median number of newly covered voxels per observation is on average $360 \pm 55\%$ higher than with any other method. Observing better efficiency here is not surprising, as this method leverages both the known environment and optimizes task selection based on information gain. However, due to information gain calculations, the *PH-tree+IG* method takes significantly longer to generate tasks than the *PH-tree+DH* and *Frontiers* methods (see Fig. 8). The *NBV* methods that also use information gain do not (fully) take advantage of the fact that the environment is known and explore stochastically, producing less efficient coverage. However, these methods outperform the *PH-tree+DH* and *Frontiers* for the tower environment (Fig. 5b). This can be explained by the topology of the structure, where more numerous distant voxels can be seen at once, favoring *NBV* approaches, while still being outperformed by the *PH-tree+IG* method.

Finally, we summarize the *Mission Duration*, *Robot Trajectory Length* and *Activity Rates* metrics in the remaining rows of Fig. 9. It is clear the *NBV*-based methods do not allow as efficient coverage of the environment, as can be seen in the *Mission Duration* plots (second row in Fig. 9). The slow task generations and stochastic exploration make them scale poorly with the number of robots, while remaining competitive with smaller robot team sizes. A distinct effect of the duration of task generation can be seen with the *Activity Rate* metric (bottom row in Fig. 9). Methods with slower task generation take longer to reassign tasks to robots, causing them to wait longer between tasks. The only methods that scale well with robot team size are the *PH-tree+DH* and *Frontiers* methods. Overall, individual robot path lengths are relatively consistent across methods. Notably, the simplest *PH-tree+DH* method performs at least as well as any other method and shows good behavior in all metrics with respect to the robot team size, with the exception of being outperformed only in the Tower environment by the *PH-tree+IG* method on the *Mission Duration* metric.

B. Physical Experiments

In Fig. 10, we report the coverage of the asset as a function of time for the various robot team sizes. Fig. 1 shows an experiment with three robots in more detail. It is clear that the multi-robot configurations outperform the single-robot one. However, increasing the robot team from two to three robots does not result in an equivalent enhancement in performance as observed when transitioning from a single- to a two-robot configuration. Regarding the rate at which coverage progresses, it is evident that the presence of a larger number of robots results in a more consistent coverage rate.

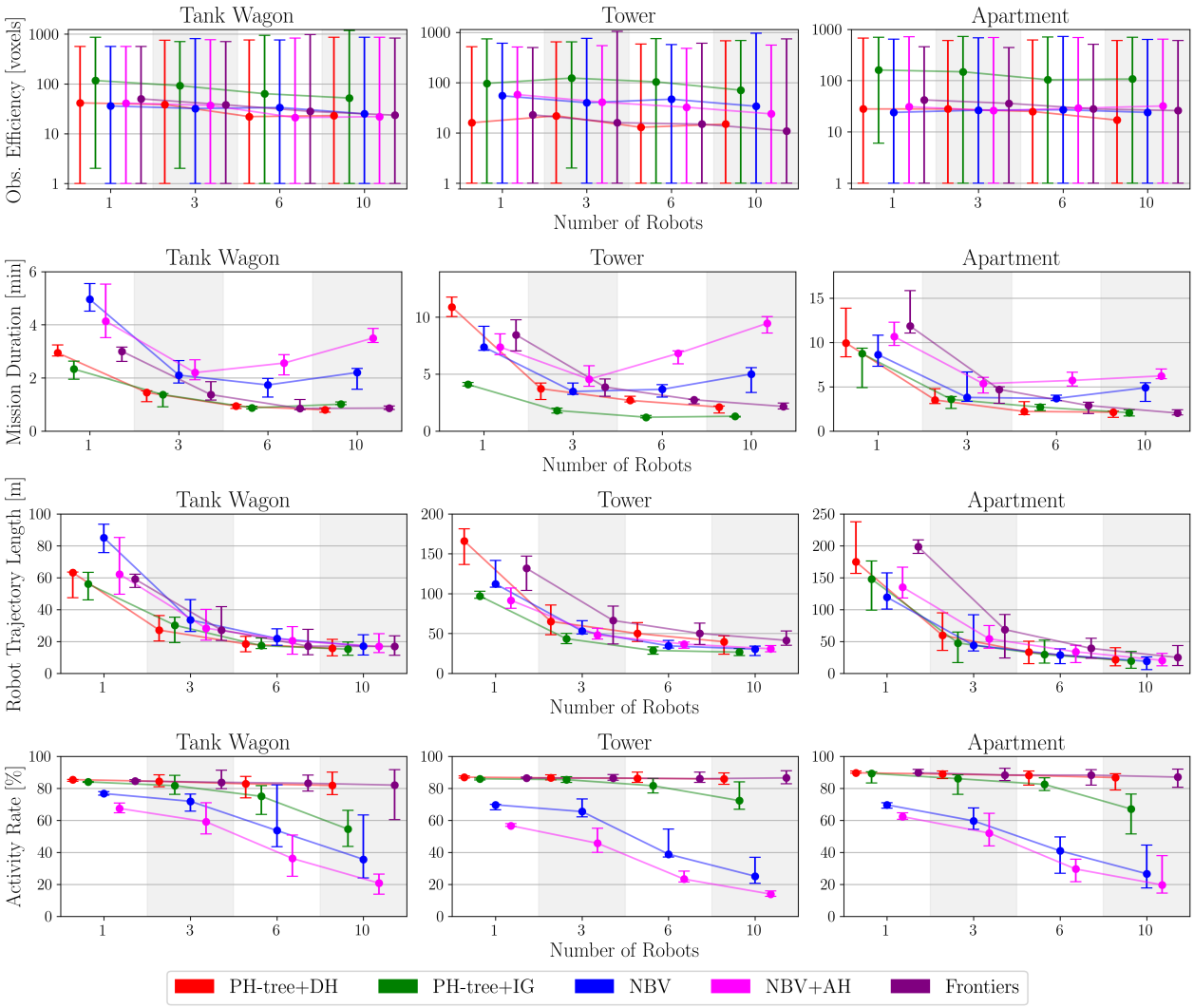


Fig. 9: Simulation results for the three environments, totaling 300 runs, reporting the median, min, and max values for each data point.

A slow down towards the end of the mission is inevitable, as remaining unobserved voxels are scarce and possibly spread between various locations. Furthermore, only a few (often only one) voxels are observed per observation towards the end of the mission.

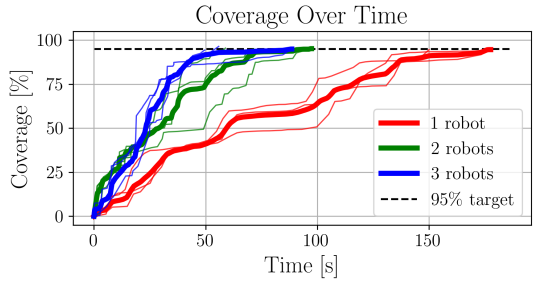


Fig. 10: Real experiment coverage over time of our method with teams of 1, 2 and 3 robots in our flying arena. Individual experiments (three per team size) are reported with thinner lines, while the thicker lines represent their average.

V. Conclusion

In this work, we conducted the benchmarking and validation of the various improvements of the method proposed in [14]. This method proposes a hybrid approach with centralized and distributed coordination schemes for the coverage of known assets or environments, where a ground station plans high-level observation tasks for the robots, and the robots coordinate their trajectories to carry out these tasks. We first proposed several improvements to the method introduced in [14] to extend its applicability and facilitate its deployment on real robots. We then implemented variants of the method by taking inspiration from or using state-of-the-art methods that leverage different paradigms for the coverage of an asset. We performed extensive experiments in simulations, allowing us to get insights on the performance and trade-offs of the tested variants compared to ours, as a function of the robot team size and complexity of the environment. Interestingly, our method, despite relying on simple distance heuristics but providing fast task generations, either outperforms or remains competitive with other methods leveraging

more complex principles such as information gain or frontier monitoring. We also show the scaling capabilities offered by the efficient task generation process for larger robot team sizes. We performed physical experiments with various robot team sizes with up to three robots, showing the applicability of the method to real MAVs, where the MAVs were running all computations fully onboard.

Several aspects could be explored in the future. For example, while this method can reach 100% coverage of an asset, it would be beneficial to study the effective quality of the coverage produced and its applicability to different fields, such as visual inspection, object reconstructions, or others, which induce different types of coverage requirements. Since this method directly stores information in the map voxels, altering its behavior can be done by adjusting the information that is stored in them. Tests on real large-scale assets should be performed. In GNSS-available settings, MAVs could be deployed around an asset with absolute localization. Challenges in terms of communication could then be explored, and multi-hop network strategies may need to be introduced to ensure connectivity with the ground station for certain environments. In GNSS-denied settings, the robots would need to localize themselves in an absolute fashion, which is a non-trivial problem and remains mostly an open question.

REFERENCES

[1] R. Almadhoun, T. Taha, L. Seneviratne, and Y. Zweiri, "A survey on multi-robot coverage path planning for model reconstruction and mapping," *SN Applied Sciences*, vol. 1, no. 8, p. 847, Jul. 2019. DOI: 10.1007/s42452-019-0872-y.

[2] C. Papachristos, K. Alexis, L. R. G. Carrillo, and A. Tzes, "Distributed infrastructure inspection path planning for aerial robotics subject to time constraints," in *2016 International Conference on Unmanned Aircraft Systems*, Jun. 2016, pp. 406–412. DOI: 10.1109/ICUAS.2016.7502523.

[3] M. D. Phung, C. H. Quach, T. H. Dinh, and Q. Ha, "Enhanced discrete particle swarm optimization path planning for UAV vision-based surface inspection," *Automation in Construction*, vol. 81, pp. 25–33, Sep. 2017. DOI: 10.1016/j.autcon.2017.04.013.

[4] M. Li, A. Richards, and M. Sooriyabandara, "Asynchronous Reliability-Aware Multi-UAV Coverage Path Planning," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, May 2021, pp. 10 023–10 029. DOI: 10.1109/ICRA48506.2021.9560770.

[5] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation. 'Towards New Computational Principles for Robotics and Automation'*, Monterey, CA, USA, 1997, pp. 146–151. DOI: 10.1109/CIRA.1997.613851.

[6] T. Cieslewski, E. Kaufmann, and D. Scaramuzza, "Rapid exploration with multi-rotors: A frontier selection method for high speed flight," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vancouver, BC, Sep. 2017, pp. 2135–2142. DOI: 10.1109/IROS.2017.8206030.

[7] 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*, Stockholm, Sweden: IEEE, May 2016, pp. 1462–1468. DOI: 10.1109/ICRA.2016.7487281.

[8] Z. Meng et al., "A Two-Stage Optimized Next-View Planning Framework for 3-D Unknown Environment Exploration, and Structural Reconstruction," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1680–1687, Jul. 2017. DOI: 10.1109/LRA.2017.2655144.

[9] S. Song and S. Jo, "Surface-Based Exploration for Autonomous 3D Modeling," in *2018 IEEE International Conference on Robotics and Automation*, Brisbane, QLD, May 2018, pp. 4319–4326. DOI: 10.1109/ICRA.2018.8460862.

[10] 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*, vol. 5, no. 2, pp. 1500–1507, Apr. 2020. DOI: 10.1109/LRA.2020.2969191.

[11] D. Morilla-Cabello, L. Bartolomei, L. Teixeira, E. Montijano, and M. Chli, "Sweep-Your-Map: Efficient Coverage Planning for Aerial Teams in Large-Scale Environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10 810–10 817, Oct. 2022. DOI: 10.1109/LRA.2022.3194686.

[12] W. Jing, D. Deng, Y. Wu, and K. Shimada, "Multi-UAV Coverage Path Planning for the Inspection of Large and Complex Structures," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2020, pp. 1480–1486. DOI: 10.1109/IROS45743.2020.9341089.

[13] R. Almadhoun, T. Taha, L. Seneviratne, and Y. Zweiri, "Multi-Robot Hybrid Coverage Path Planning for 3D Reconstruction of Large Structures," *IEEE Access*, vol. 10, pp. 2037–2050, 2022. DOI: 10.1109/ACCESS.2021.3139080.

[14] L. Wälti, I. K. Erunsal, and A. Martinoli, "Multi-robot Online Coverage with a Team of Resource-Constrained Micro Aerial Vehicles," in *Proc. of the 17th International Symposium on Distributed Autonomous Robotic Systems*, Springer Proceedings in Advanced Robotics, vol. 34, 2026, pp. 192–209. DOI: 10.1007/978-3-032-04584-3_14.

[15] T. Zäschke, C. Zimmerli, and M. C. Norrie, "The PH-tree: A space-efficient storage structure and multi-dimensional index," in *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, Snowbird Utah USA, Jun. 2014, pp. 397–408. DOI: 10.1145/2588555.2588564.

[16] O. Michel, "Cyberbotics Ltd. Webots ™ : Professional Mobile Robot Simulation," *International Journal of Advanced Robotic Systems*, vol. 1, no. 1, p. 5, Mar. 2004. DOI: 10.5772/5618.

[17] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, Apr. 2013. DOI: 10.1007/s10514-012-9321-0.

[18] ModalAI Inc. "VOXL Flight Datasheets," Accessed: Sep. 12, 2025. [Online]. Available: https://docs.modalai.com/voxl-flight-datasheet/.