A Multirobot System for 3-D Surface Reconstruction With Centralized and Distributed Architectures

Guillaume Hardouin, Julien Moras, Fabio Morbidi, Julien Marzat, El Mustapha Mouaddib

A Multi-Robot System for 3D Surface Reconstruction with Centralized and Distributed Architectures

Guillaume Hardouin[1,2,*], Julien Moras[1], Fabio Morbidi[2], Julien Marzat[1], El Mustapha Mouaddib[2]

***Abstract*— This paper proposes an original solution to the problem of surface reconstruction of large-scale unknown environments, with multiple cooperative robots. As they progress through the 3D environment, the robots rely on volumetric maps obtained via a TSDF representation to extract discrete Incomplete Surface Elements (ISEs), and a list of candidate viewpoints is generated to cover them. A Next-Best-View planning approach, which approximately solves a Traveling Salesman Problem (TSP) via greedy allocation, is then used to iteratively assign these viewpoints to the robots. Two multi-agent architectures, a centralized one (TSP-Greedy Allocation or TSGA) and a distributed one (dist-TSGA), in which the robots locally compute their maps and share them, are developed and compared. Extensive numerical and real-world experiments with multiple aerial and ground robots in challenging 3D environments, show the flexibility and effectiveness of our surface representation of a volumetric map. The experiments also shed light on the nexus between reconstruction accuracy and surface completeness, and between total distance traveled and execution time.**

***Index Terms*— Next-Best-View (NBV) planning, Multi-robot system, 3D reconstruction, Truncated Signed Distance Function (TSDF), Sampling-based motion planning.**

I. Introduction

As research in multi-agent systems is developing at a rapid pace, autonomous cooperative robots are being increasingly used in time-sensitive applications, such as ocean sampling [1], cinematography [2], wildlife survey [3], logistics automation [4], and mapping of mine tunnels [5], just to name a few relevant examples. However, in spite of major progress in the field, we are still a long way from coordinated online exploration and reconstruction of vast, complex, unknown 3D environments with teams of mobile robots (large industrial plants, archaeological sites, battlegrounds, etc.). For this challenging problem, distributed solutions offer distinctive advantages over centralized ones (resilience against failures, scalability with respect to the number of robots, etc.), but they are still relatively rare in the literature.

In this paper, we present a novel 3D reconstruction method for multiple cooperative robots that addresses the problem of surface inspection of unknown environments, via a Next-Best-View (NBV) frontier-based planner. A single objective function, which takes the *surface representation* explicitly into account, is used to plan collision-free paths for the robots within a *centralized* and a *distributed architecture*. Numerical simulations and real-world experiments show that the proposed architectures are robust against uncertain measurements, and provide accurate, complete and time-efficient 3D reconstructions.

This work is a significant outgrowth of our two previous conference papers [6], [7]. We propose here a new distributed multi-robot architecture and numerically show its pros and cons with respect to the centralized one, in [7]. We also compare the single-robot method in [6], with a relevant state-of-the-art algorithm [8]. Finally, the present article includes a more thorough description of the background, a more general mathematical formulation, and extensive hardware experiments with multiple ground robots.

In summary, the original contributions of this paper are the following:

- Differently from the conventional NBV exploration approaches, which select viewpoints among a large set of randomly-sampled poses [8], [9], we directly consider a 3D representation of the incompleteness of the surface to generate a roadmap of viewpoints and to carry out the reconstruction with a team of mobile robots,
- We introduce multi-agent NBV planners to route robots to viewpoint configurations, and achieve surface reconstruction. More specifically, clusters of configurations are greedily allocated to the robots by successively (approximately) solving a Traveling Salesman Problem (TSP),
- To validate our approach, which admits both a centralized and a distributed implementation, we undertook a large campaign of numerical simulations with Unmanned Aerial Vehicles (UAVs) and real-world experiments with ground robots.

The remainder of this paper is organized as follows. Sect. II reviews and catalogues the related work, while Sect. III formally introduces the problem studied in the paper. Sect. IV provides a general overview of the proposed approach, and Sect. V and Sect. VI deal with the perception and planning problems, respectively. The centralized and distributed multi-robot architectures are tested via numerical and hardware experiments in Sect. VII and Sect. VIII, respectively. Finally, Sect. IX concludes the paper, and identifies possible areas of improvement.

II. Related Work

Planning informative paths for real-time 3D modeling of unknown environments has been the subject of intense research in recent years. Volumetric-mapping methods represent the free and occupied space, and provide accurate surface estimations for reconstruction, both for single and multiple robots.

[1]DTIS, ONERA, Université Paris-Saclay, 91123 Palaiseau, France. Emails: firstname.lastname@onera.fr

[2]MIS laboratory, Université de Picardie Jules Verne, 80039 Amiens, France. Emails: {fabio.morbidi, mouaddib}@u-picardie.fr

This work was supported by ONERA DTIS and by the Hauts-de-France region, through the research project ScanBot, "*Robotic Scanners for Automatic 3D Modeling of Cultural Heritage*" (2018-2021). G. Hardouin also received support from the Interreg VA France (Channel) England ADAPT project (2016-2022).

*Corresponding author.

TABLE I
MAIN FEATURES OF SOME REPRESENTATIVE STATE-OF-THE-ART NBV RECONSTRUCTION METHODS, COMPARED TO OURS.

Method	Mapping	View planning	Criterion	Path computation	Multi-robot
Ours, 2020 [6], [7]	**TSDF**	**Roadmap**	**Surface**	**Lazy PRM⋆**	✔
[Bircher *et al.*, 2018] [10]	TSDF	Random sampling	Volumetric	RRT	✘
[Song & Jo, 2018] [11]	TSDF, Point cloud	Random sampling	Volumetric	RRT⋆	✘
[Schmid *et al.*, 2020] [8]	TSDF	Random sampling	Surface	RRT⋆	✘
[Kompis *et al.*, 2021] [12]	TSDF, ESDF	Roadmap	Surface	A⋆	✘
[Mannucci *et al.*, 2017] [13]	OctoMap	Random sampling	Volumetric	RRT⋆	✔
[Corah & Michael, 2021] [14]	General occupancy grid	Random sampling	Volumetric	MCTS	✔
[Lauri *et al.*, 2020] [15]	General occupancy grid	Offline sampling	Volumetric	Offline	Multi-sensor

In tandem with these emerging mapping methods, volume exploration planners have been developed to rapidly explore unknown volumes and provide coarse 3D reconstructions. The main focus of early work on surface-based reconstruction has been on accuracy, and the optimized cost function incorporates one or multiple surface criteria. Recently, the volumetric and surface-based approaches have been combined to take advantage of their unique properties, but only the single-robot case has been studied.

A. Volumetric mapping

Volumetric mapping consists in discretizing the 3D space into small cells: the cells can be unknown, empty or occupied, and they can be used to represent a 3D scene of interest. A popular online 3D modeling approach, *OctoMap*, was introduced in [16]. It relies on a 3D occupancy grid with an internal octree data structure. This representation adapts the level of detail of the map to the environment, which reduces memory usage, enabling real-time processing on CPU. In [17], the authors proposed *KinectFusion*, which uses measurements from an RGB-D camera to calculate the Truncated Signed Distance Function (TSDF) [18] over a grid, leading to an implicit surface representation. Successive improvements of the method resulted in reduced memory usage [19], and recently the open source library CHISEL [20] made it available for 3D reconstruction on mobile devices. In [21], this mapping method has been revisited by considering Euclidean Signed Distance Fields (ESDF), which improves the accuracy of the distance map. The concept of manifold mapping [22] has been incorporated within a TSDF framework, to mitigate the mapping error due to localization drift [23], and the traditional monolithic map has been replaced by a collection of multiple local sub-maps (or patches). In [24], the authors have extended this concept to distributed mapping, for multiple ground robots.

B. Single-robot planning

Given a volumetric map, a robot can try to explore an unknown volume containing objects of interest. *Volumetric-exploration methods* usually leverage a volumetric representation (e.g. an OctoMap), to identify known, unknown, or occupied areas. *Sampling-based planners*, such as Rapidly-exploring Random Trees (RRT [25], RRT⋆ [26]), are used for trajectory generation by incrementally expanding a tree constituted of randomly-sampled sensor poses in the free space. On the other hand, Probabilistic Roadmap planners (PRM [27], Lazy PRM⋆ [28]), extract the trajectory/path from a graph formed by randomly-sampled poses between a start and a goal configuration, according to a given objective function. In an NBV-planning framework, volumetric exploration methods are usually related to *informative path planning*, which consists in expanding trees and in selecting the NBV trajectory that guarantees the maximum coverage of the volume (see [29]–[33] for RRT-based methods, and [34] for a PRM-based method). Pose coverage along the trajectory is evaluated via ray tracing [35], and as the robot follows the assigned path, the volume is automatically explored. These fast and efficient methods rely on a coarse volumetric map of the environment for navigation purposes, but they do not explicitly account for the completeness and accuracy of the reconstructed surface.

Surface-inspection methods use the current surface to generate a roadmap of candidate viewpoints which ensure a complete and accurate 3D reconstruction. NBV methods determine the next best viewpoints to visit, depending on the mission of the robots. Among them, *frontier-based methods* yield viewpoint configurations pointing towards the frontier of the known surface, represented as a mesh, according to a given orientation, position, or sensing constraint. By visiting these configurations, the robot-sensor gathers new surface information with some overlap, for ensuring continuity [36]–[38]. Most of these approaches deal with small objects reconstructed by cameras mounted on the end-effector of robot manipulators, and make strong assumptions on the navigable free space. In the last five years, NBV inspection methods have been extended to mobile robots by using volumetric representations [9], [37] and the TSDF [6], [8], [39].

Recently, *hybrid* or *mixed methods*, which benefit from the main advantages of surface-inspection and volumetric-exploration approaches, have emerged for improved reconstruction accuracy and faster coverage (see Table I). In [10], the reconstruction process includes two steps: first, a coarse TSDF map of the environment is generated, and then, in a second modeling step, the surface is refined. Reasoning on the occupancy map, the algorithm in [40] allows the robot to cover the whole surface model. The authors have extended their approach in [11], where the RRT⋆ path is further refined by taking the completeness of the environment surface, into account. More recently, in [41], the method has been improved and validated via numerical and real-world experiments. These approaches rely on both volumetric and surface representations, which are costly to generate, and only

a few of them solve the surface-inspection problem directly. A RRT-based planning method with a volumetric representation of the surface, has been presented in [8]. However, the poses are still sampled randomly rather than planned in advance, which may lead to unnecessary maneuvers. In [6], we proposed to extract the viewpoints directly from the knowledge of the map. A planning roadmap is created and used for 3D reconstruction, and the magnitude of unknown surface determines the stopping criterion. Finally, in [12], the authors introduced a method reminiscent of ours, based on ESDF mapping and A* path search.

C. Multi-robot planning

As allused to in the previous sections, the existing research on incremental online reconstruction generally targets a single robot, and multi-robot cooperative systems are still relatively rare in the literature [42]. By considering robots equipped with laser scanners in a 2D environment, the authors in [43] were the first to propose a method for computing frontier cells, and to define the trade-off between utility and distance traveled in a multi-agent exploration task. In [44], the authors compared multiple volumetric objective functions in terms of reconstruction time and volume completeness, by considering pose measurement errors in 2D environments. In [13], the authors considered a team of aerial robots in an uncluttered outdoor environment, and proposed one of the first cooperative frontier-based methods, which relies on 3D space modeling for multi-robot exploration. The centralized OctoMap and the (RRT*-based) coordinated motion planning of the aerial vehicles are computed on a base station, and the algorithm is evaluated via realistic numerical experiments with emulated stereo sensors in ROS/Gazebo. Exploration methods based on probabilistic occupancy maps with entropy reduction, such as decMCTS [45] or SGA [46], stem from the notion of mutual information of range sensors [47]. The authors in [48], [49] have proposed a finite-horizon decentralized planner, called DSGA (Distributed Sequential Greedy Assignment), which relies on sampling-based Monte-Carlo Tree Search (MCTS) [50]. The paths are allocated to the robots by solving a submodular maximization problem over matroid constraints with greedy assignment heuristics. More recently, in [14], the authors have established connections between information-theoretic and volumetric coverage objectives in terms of expected coverage, for teams of mobile robots. Finally, a similar matroid-constrained submodular maximization problem has been considered in [15] for multi-sensor NBV planning, and real-world experiments have been conducted with two KUKA robot arms. One NBV per sensor and per iteration is computed, and viewpoint sampling and trajectory generation are performed offline, by assuming a partial prior knowledge of the environment (location of the target object).

Based on this literature review, we can notice that a large body of research has leveraged volumetric exploration for 3D reconstruction, but without taking the problems of surface completeness and occlusion explicitly into account. Moreover, even though the surface inspection problem with mobile robots is becoming increasingly popular, to the best of our knowledge, no multi-robot formulation exists.

In this work, we propose a generic NBV planning strategy inspired by the mixed approaches, to solve the surface inspection problem and cooperatively reconstruct large-scale environments with a team of mobile robots. Our frontier-based method relies on a volumetric representation of the surface, which allows to identify areas of interest to be scanned. Candidate viewpoint configurations are generated from these areas in compliance with the sensing and dynamic constraints of the robots, and they are clustered according to their location in space. In order to find the best path for each robot, we evaluate the interest of visiting a specific cluster. By successively solving this assignment problem, we find the paths which allow to explore the unknown environment and maximize the completeness of reconstructed surface, while ensuring short travel distances and execution times. The proposed strategy, called TSGA (TSP-Greedy Allocation), has been validated via extensive experiments with multiple quadrotor UAVs and wheeled robots, by using a centralized architecture. A distributed variant, referred to as dist-TSGA, is also introduced and tested via numerical and real-world experiments. With dist-TSGA, it is possible to perform cluster assignment in a decentralized fashion, and to keep track of the map under construction.

III. Problem formulation

In this paper, we consider a team of N cooperative mobile agents[1]. Let $\mathbf{q}^i \in \mathrm{SE}(m)$ be the pose of agent i, and $\mathbf{q}_0^i$ its initial configuration, $i \in \{1, 2, \dots, N\}$: $m = 2$ in the case of ground vehicles, and $m = 3$ in the case of aerial vehicles. We assume that all agents are equipped with an accurate localization system which allows to estimate their pose with respect to a global reference frame, and that a robust low-level trajectory tracking algorithm is available. Each agent is equipped with a forward-facing depth sensor with limited field-of-view (FOV) and sensing range, extrinsically calibrated with respect to its body frame. The agents should cooperatively scan an unknown 3D environment (for instance, a building), characterized by its surface. A mapping algorithm is required to build a representation of the reconstructed surface and identify the free space for navigation. We consider a volumetric mapping, which allows to build a map M as a collection of discretized 3D space elements. These elements, referred to as *voxels* $\mathbf{v} \in M$, represent *unknown*, *occupied*, or *empty* space. Let $X \subset M$ be the set of unknown voxels, and $A \subset M$

[1]We will use the terms "agent" and "robot", and "path" and "trajectory" interchangeably, throughout the paper.

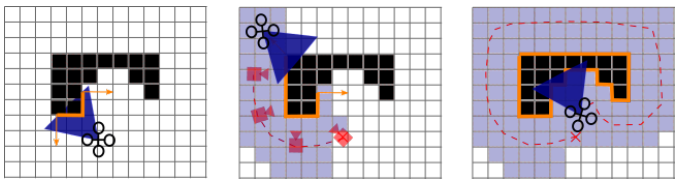


Fig. 1. 2D illustration (from left to right) of NBV planning for surface inspection via volumetric mapping. White, blue, and black voxels represent unknown, empty, and occupied voxels, respectively. The surface is depicted in orange.

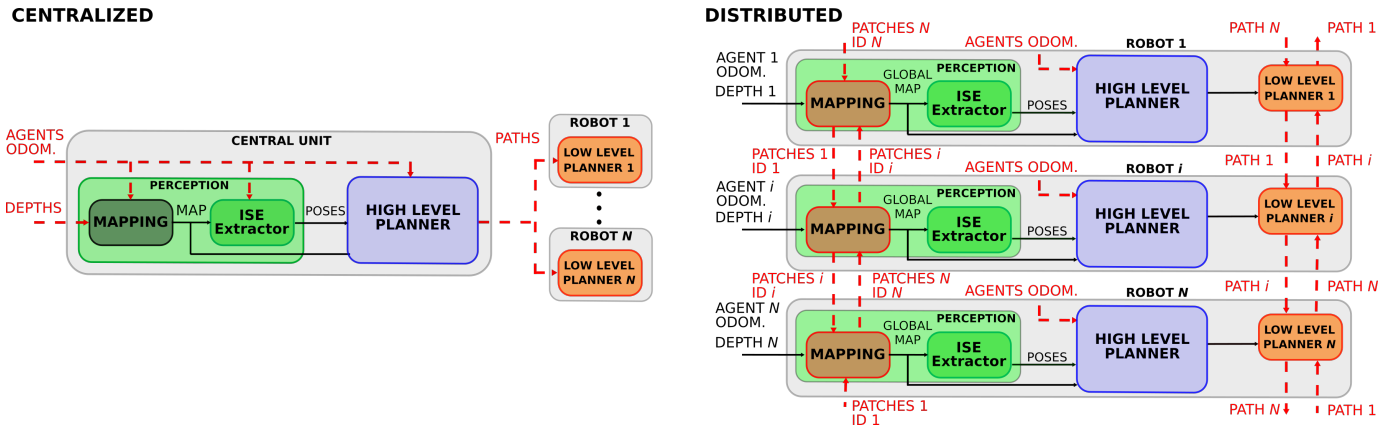


Fig. 2. General flow chart of our multi-agent surface-reconstruction architectures: [left] centralized, [right] distributed. The internal structure of the perception module is shown inside the green shaded box. The intra- and inter-block connections are represented with black solid and red dashed lines, respectively.

the set of known voxels such that $X \cap A = \emptyset$. Moreover, let $O \subset A$ be the set of occupied voxels, and $E \subset A$ the set of empty voxels. The goal of the scanning process is to discover unknown voxels: in particular, a voxel is said to be *scanned* when it becomes known, once at least one agent has detected it. Similarly to [9], [12], [39], we aim at identifying the incomplete surface within a given volume. Based on [39, Equ. (2)], we propose the following general definition of incompleteness of a surface model:

Definition 1 (Incomplete surface element): We call *Incomplete Surface Element*, or ISE, for short, a voxel $\mathbf{v} \in M$ lying on the surface at a frontier, near both the unknown and empty space. Let C be the set of all ISEs. A voxel $\mathbf{v} \in C$ if and only if

$$\begin{aligned} &a)\ \mathbf{v} \in E, && \text{(empty)} \\ &b)\ \exists\, \mathbf{u} \in \mathcal{N}_{\mathbf{v}}^{6} \text{ s.t. } \mathbf{u} \in X, && \text{(unknown)} \\ &c)\ \exists\, \mathbf{o} \in \mathcal{N}_{\mathbf{v}}^{18} \text{ s.t. } \mathbf{o} \in O, && \text{(occupied)} \end{aligned}$$

where $\mathcal{N}_{\mathbf{v}}^{6}$ and $\mathcal{N}_{\mathbf{v}}^{18}$ denote the 6- and 18-connected voxel neighborhoods of $\mathbf{v}$, respectively.

Definition 2 (Remaining incomplete surface): Let Q be the set of all collision-free configurations of an agent, and let $Q_c \subseteq Q$ be the set of all configurations from which an ISE $\mathbf{v} \in C$ can be scanned. The *remaining incomplete surface* is then defined as

$$C_{\text{rem}} = \bigcup_{\mathbf{v} \in C} \{\mathbf{v} \mid Q_c = \emptyset\}.$$

We will use the function $\mathbf{p}_{j,k}^{i}(s) : [0,\, 1] \rightarrow \text{SE}(m)$, $m \in \{2,\, 3\}$, to define the path of agent i from configuration j to configuration k, where $\mathbf{p}_{j,k}^{i}(0) = \mathbf{q}_{j}^{i}$ and $\mathbf{p}_{j,k}^{i}(1) = \mathbf{q}_{k}^{i}$, $i \in \{1, 2, \ldots, N\}$. We assume that $\mathbf{p}_{j,k}^{i}(s)$ is collision-free and feasible for agent i (i.e. the kinematic/dynamic constraints of the robot are satisfied along the path). The problem studied in this paper can then be formally stated as follows.

Problem 1 (Multi-agent inspection problem): Consider a team of N agents with initial configurations $\mathbf{q}_{0}^{i} \in Q,\ i \in \{1, 2, \ldots, N\}$. The multi-agent inspection problem asks to find collision-free paths $\mathbf{p}_{0,f}^{i}(s)$ visiting the poses $\mathbf{q}_{k}^{i}$, $k \in \{0, 1, \ldots, f\}$, which allow the agents to scan the set $C_{\text{ins}} = C \setminus C_{\text{rem}}$ of all ISEs contained in the current reconstructed map M.

By progressing along their paths $\mathbf{p}_{0,f}^{i}(s),\ i \in \{1, 2, \ldots, N\}$, the agents are able to disclose the unknown space, discover new ISEs, and iteratively solve the inspection problem until $C_{\text{ins}} = \emptyset$. Fig. 1 graphically illustrates this idea for a planar quadrotor.

Based on these premises, in the next section, we will provide a general overview of the approach proposed in this work to solve Problem 1.

IV. Overview of the Proposed Approach

In this paper, we introduce a generic multi-agent system for 3D surface reconstruction of unknown environments, which admits both a centralized and a distributed implementation. The centralized and distributed architectures depicted in Fig. 2, are well suited to accommodate multiple ground or aerial robots, or a combination thereof (i.e. heterogeneous fleets). The architectures include two distinct modules: a *perception module* (green block in Fig. 2), which extracts the ISEs (cf. Sect. III) from a volumetric map estimated online, and a *planning module* (blue and orange blocks in Fig. 2), that is in charge of computing the paths of the robots.

A mapping algorithm acts as the front-end of the perception module, which takes as input the depth maps generated by the on-board sensors (RGB-D cameras, stereo-rig, etc.) together with their associated poses, and integrates them into a 3D volumetric map used for reconstruction (i.e. extraction of ISEs) and for navigation (i.e. collision-free path planning and tracking in the free space). The map is then processed for the duration of the overall mission in order to extract new ISEs, which are the centerpieces of the planning module. We chose the TSDF representation for its attractive properties, and in particular for its ability to implicitly represent surfaces (see Sect. V-A). In fact, it allows to generate configurations in the free space that efficiently cover the ISEs, while taking the specificities of the environment and depth sensors (range, resolution, etc.) explicitly into account (see Sect. V-B). However, note that other volumetric mapping methods could

be used as well, with minor modifications. The *centralized architecture* incrementally integrates all input data (depth maps and poses from all agents) into a unique map on a single base station, where all the ISEs are generated. In the *distributed architecture*, instead, each agent computes its own local map, henceforth referred to as "*patch*" and identified by a unique ID, based on its own sensing and localization information. The patches are exchanged via a distributed algorithm based on manifold mapping [22].

The planning module guarantees that the agents complete the surface reconstruction. The scanning process stops when no ISEs are left. Given the list of poses provided by the perception module, one can schedule the visit of each configuration via an appropriate TSP-based path finder. To this end, the TSGA planner (TSP-Greedy Allocation) clusters sets of configurations according to their location in space, in order to identify and rank areas of interest in the incomplete map. It then generates a directed graph which represents the travel utility of visiting a cluster, depending on the capabilities of each robot (terrestrial or aerial). In order to maximize the cumulative utility function at the team level, collision-free paths are extracted from the digraph and broadcast to the robots (see Sect. VI-A). The high-level paths are sent to the low-level planners, which generate sampling-based trajectories for the agents and gather the odometric and path-allocation information for collision avoidance (see Sect. VI-B). In the distributed architecture, global-map inconsistencies, due, e.g., to patch losses or communication delays, may result in clusters assigned to multiple robots. To overcome this problem, the low-level planners exchange their current paths with the robots, check the consistency between the individual and team-wise allocation, and wait for a re-assignment, if needed. Finally, trajectory tracking is performed with standard controllers (e.g. PID or Model Predictive Control).

V. Perception

A. Surface-based mapping

A volumetric map M, here based on a TSDF representation, is used to detect non-reconstructed areas, as defined by the extraction of ISEs (see Sect. V-B). The TSDF map [18] consists of a voxel grid, where each voxel contains a truncated signed distance value $\phi \in \mathbb{R}$ and a positive weight w. It implicitly represents surfaces, which correspond to the zero level set of the distance field: hence, the TSDF volume is a *volumetric representation of a surface*. Algorithms such as MarchingCubes [51] can be used to extract a triangular mesh, which is an explicit representation of those surfaces, e.g. for visualization. The map is built in an incremental fashion by sequentially integrating depth measurements. In order to keep the map consistent, the pose of the sensor on-board the robot must be used to relocate depth measurements with respect to the map frame. We assume that this pose is provided by a localization system which relies, for example, on a visual SLAM algorithm (cf. [52], [53]), and that the pose estimates are sufficiently accurate. At each time step and for each voxel $\mathbf{v}$, the integration is performed by recursively computing a weighted mean of the distance. In order to take into account uncertainty due to sensors [54], [55], the new measurements are weighted by an inverse-squared distance increment $1/z^2_{\mathbf{q}}(\mathbf{v})$, where $z_{\mathbf{q}}(\mathbf{v})$ is the distance between voxel $\mathbf{v}$ and the current pose $\mathbf{q}$ of an agent[2]. The state of a voxel $\mathbf{v}$ is set to *known* (either *occupied* or *empty*), if $w(\mathbf{v}) \geq W_{\text{th}}$ and to *unknown* if $w(\mathbf{v}) < W_{\text{th}}$, where the positive threshold W_{th} depends on the sensing range of the depth sensor.

B. ISE extractor and viewpoint generation

Similarly to [39], a voxel $\mathbf{v} \in M$ is considered as an ISE, i.e. $\mathbf{v} \in C$, as stated in Definition 1, if it verifies the following conditions:

$$
\begin{aligned}
&a)\ w(\mathbf{v}) \geq W_{\text{th}} \wedge \phi(\mathbf{v}) > 0, && \text{(empty)}\\
&b)\ \exists\, \mathbf{u} \in \mathcal{N}^6_{\mathbf{v}} \text{ s.t. } w(\mathbf{u}) < W_{\text{th}}, && \text{(unknown)}\\
&c)\ \exists\, \mathbf{o} \in \mathcal{N}^{18}_{\mathbf{v}} \text{ s.t. } w(\mathbf{o}) \geq W_{\text{th}} \wedge \phi(\mathbf{o}) \leq 0. && \text{(occupied)}
\end{aligned}
$$

Definition 3 (Scanned element): A voxel $\mathbf{v} \in M$ which satisfies, $w(\mathbf{u}) \geq W_{\text{th}},\ \forall\, \mathbf{u} \in \mathcal{N}^6_{\mathbf{v}}$, is called a *scanned element*.

The direction $\overline{\mathbf{n}}_{\mathbf{v}}$ to observe the ISE $\mathbf{v}$, is determined from the gradient of the weight function $\nabla w(x, y, z)$, which can be computed as

$$
\mathbf{n}_{\mathbf{v}} = \sum_{\mathbf{c} \in \mathcal{N}^{26}_{\mathbf{v}}} w'(\mathbf{c})\, \frac{\mathbf{c} - \mathbf{v}}{\|\mathbf{c} - \mathbf{v}\|}, \qquad \overline{\mathbf{n}}_{\mathbf{v}} = \frac{\mathbf{n}_{\mathbf{v}}}{\|\mathbf{n}_{\mathbf{v}}\|},
$$

where $\mathcal{N}^{26}_{\mathbf{v}}$ is the 26-connected neighborhood of $\mathbf{v}$ and the weight function

$$
w'(\mathbf{c}) = \begin{cases} -W_{\text{th}} & \text{if voxel } \mathbf{c} \text{ is occupied,} \\ W_{\text{th}} & \text{otherwise.} \end{cases}
$$

Note that the last definition differs from [39, Equ. (6)], since $w'(\mathbf{c})$ takes the value W_{th} if the voxel is unknown or empty (in our experiments, we empirically observed that this variant is more robust against noisy data). A sensor configuration is generated along the direction $\overline{\mathbf{n}}_{\mathbf{v}}$ at a distance δ_{pose} from the corresponding voxel $\mathbf{v}$ (see Fig. 3). The sensor is oriented towards $\mathbf{v}$ along $-\overline{\mathbf{n}}_{\mathbf{v}}$, and the value of δ_{pose} depends on the sensing range of the depth sensor.

Note that in some recent works [12], [33], [56], the authors sample multiple viewpoints within a cone whose principal axis

[2]In the interest of clarity, in the remainder of this section, we will drop the superscript i, and we will simple write $\mathbf{q}$ instead of $\mathbf{q}^i$.

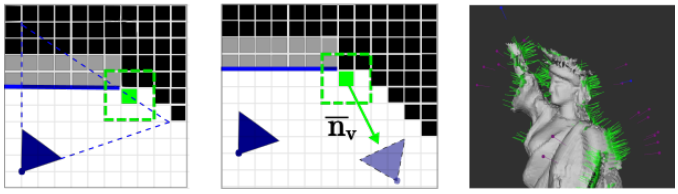


Fig. 3. [left] Two-dimensional example of ISE $\mathbf{v}$ (filled green square). Its 2D neighborhood is represented by a dashed green square. Unknown voxels are black, occupied are gray, and empty voxels are white. The reconstructed surface is depicted as a blue segment, and the sensor configuration and its frustum as a dark blue triangle. [center] Direction from the contour, $\overline{\mathbf{n}}_{\mathbf{v}}$, and corresponding viewpoint configuration at distance δ_{pose} from $\mathbf{v}$ (light blue triangle). [right] Snapshot of a simulated reconstruction in progress, with ISEs and their directions from the contour (green arrows).

is aligned with the viewing direction from the surface, and then select the most promising one by solving an optimization problem. However, this approach incurs extra computational costs, and to keep our formulation simple, we preferred not to include it here.

If two poses $\mathbf{q}_j, \mathbf{q}_k \in Q$ generated from the ISEs $\mathbf{v}_j, \mathbf{v}_k \in C$, are located within a short distance (i.e. $\text{dist}(\mathbf{q}_j, \mathbf{q}_k) < \epsilon$, for a small $\epsilon > 0$), and their viewing directions $\overline{\mathbf{n}}_{\mathbf{v}_j}, \overline{\mathbf{n}}_{\mathbf{v}_k}$ are almost parallel (i.e. $|\overline{\mathbf{n}}_{\mathbf{v}_j} \cdot \overline{\mathbf{n}}_{\mathbf{v}_k}| \simeq 1$, where "$\cdot$" denotes the dot product), then these configurations are merged into a single viewpoint, by averaging their positions and orientations. This allows to reduce the overall number of poses, without missing key information. Even after the merging step, a large number of candidate poses pointing towards the ISEs is typically generated in large-scale environments, which is not compatible with the planning objective. To avoid this problem, neighboring viewpoints are grouped into N_c *clusters* U_j, $j \in \{1, 2, \ldots, N_c\}$ (the idea, as will become apparent later, is to assign each cluster to an agent, in order to improve efficiency). The set of all clusters is denoted by $\mathcal{U} = \{U_1, U_2, \ldots, U_{N_c}\}$. A configuration $\mathbf{q}_l$ belongs to a generic cluster U if $\exists\, \mathbf{q}_j \in U$ such that $\text{d}(\tau_l^j) < d_\nu$, where $\text{d}(\tau_l^j)$ denotes the length of the path τ_l^j between $\mathbf{q}_l$ and $\mathbf{q}_j$ on a directed graph (to be defined in Sect. VI-A), and d_ν is an upper bound on the distance. If no neighbors are found, d_ν is increased up to a maximum value $d_\nu^{\max}$. Once the clusters have been defined, their respective level of informativeness needs to be quantified, since each cluster does not necessarily contain the same number of viewpoints. To evaluate a configuration, we use the ray-tracing method [35] from a frontier-based perspective, i.e. the ISEs that can be seen, are counted. Let $C_{\mathbf{q}}$ be the set of all ISEs seen from viewpoint $\mathbf{q}$ and let $C_U = \bigcup_{\mathbf{q} \in U} C_{\mathbf{q}}$. The *gain* $g(U)$ of cluster U is then defined as

$$g(U) = \frac{|C_U|}{|C|}, \tag{1}$$

where $|C_U|$ denotes the cardinality of the set C_U.

C. Centralized vs distributed mapping

In the centralized architecture, all the computations are performed on the base station. The agents send their poses and depth maps to it, and a GPU-based algorithm[3] fuses them to create the global TSDF volume. The ISE extractor then computes the ISEs for the whole team. The correct operation of the base station is crucial to the centralized architecture: in fact, if it crashes, a system breakdown occurs. If the communication with a robot is temporary broken, the robot freezes, and a degradation of planning performances is experienced. On the other hand, in the distributed architecture, a CPU-based distributed manifold mapping founded upon [24], enables each agent to compute TSDF (sub-)maps on its own embedded computer. The general block diagram of the mapping module is depicted in Fig. 4. To synchronize the map among the agents, it is subdivided into different patches. Each patch is a local TSDF with a unique ID. It is associated with a local frame (i.e. the frame of the first depth map processed), which is used to integrate the depth maps until a certain user-defined event is triggered. Originally, in [24], a distance-traveled criterion was considered, for simultaneous localization and mapping (SLAM). Here, we assume that a new patch is created, when a certain number of depth maps has been integrated into the current map. Once this occurs, the current patch is locally stored into the agent's *private map* and shared with the others, and a new "current" patch is initialized. Moreover, for each new patch stored, the list of patch IDs is updated and broadcast to the other agents. Thanks to this ID list, a client-server protocol allows the other agents to request a missing patch which might have been lost during the first communication attempt. The request is processed by the closest agent which owns it. Finally, when a robot receives a patch sent by another agent, it stores it locally into its *public map*.

An approximation of the global map, denoted by $\widehat{M}_i$, can thus be constructed by agent i, as the union of its current patches, its private map, and its public map. We improved the method proposed in [24], in order to rebuild the TSDF map from the collection of patches. In fact, the volumes are aggregated, and the overlapping regions between the patches are fused together by summing up the weights and computing the weighted average of distance values for each TSDF voxel. To not overload the communication network, the current map of an agent is not accessible to the others until its completion. Therefore, even excluding the communication losses, the agents do not have access to the full map M, simultaneously. However, it is worth pointing out that our list-and-request mechanism to synchronize the older patches between the agents, ensures that the majority of the global map M is available to the agents, except for the most recent patches which are currently being built by each agent. Following the procedure described in Sect. V-B, the ISEs are extracted by agent i from $\widehat{M}_i$, and $\widehat{\mathcal{U}}_i = \{U_1^i, U_2^i, \ldots, U_{N_C}^i\}$ is used to denote the set of clusters that should be observed to complete the surface reconstruction process. Finally, the gains are computed with equation (1), as in the centralized case. Fault tolerance is an asset to our architecture: in fact, in the event of a communication failure, each agent can evolve independently using the last map exchanged. New patches are locally stored as the agent reconstructs the environment on its own, and if communication is re-established, they can be requested by the team, once the IDs list has been updated.

VI. Planning

A. Next-Best-View planning

The high-level planner allocates clusters to the agents and schedules their visit according to a given common TSDF map (or its best approximation, in the distributed architecture). To formulate our optimization problem, we introduce the weighted directed graph $\mathcal{G} = (\mathcal{U}, \mathcal{E}, \{a_{UV}\}_{(U,V) \in \mathcal{E}})$, where $\mathcal{U}$ is the set of clusters, $\mathcal{E}$ is the set of edges, and $\{a_{UV}\}_{(U,V) \in \mathcal{E}}$ is the collection of weights associated to the edges. Each directed edge $e_{UV} \in \mathcal{E}$ connects cluster U

[3]When the depth maps sent by all the agents are fused, a GPU implementation is needed to ensure real-time performance.

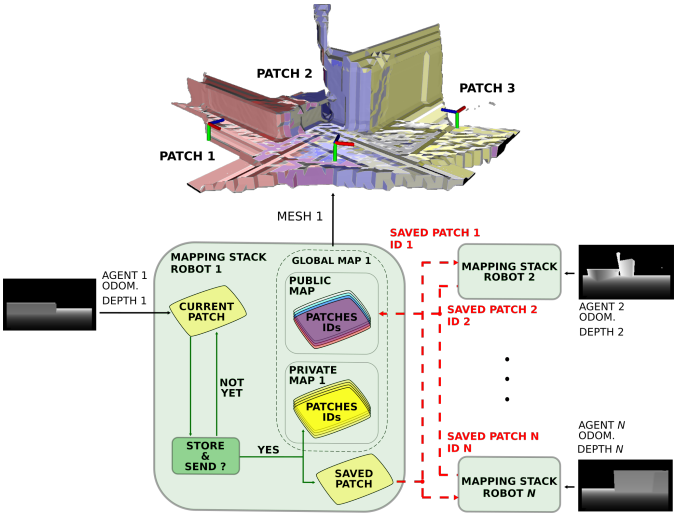


Fig. 4. Distributed mapping: inner structure of the module. The intra- and inter-block connections are represented with green solid and red dashed lines.

to cluster V, with $U, V \in \mathcal{U}$. It is assumed that the initial configuration of agent i belongs to one of the clusters of $\mathcal{G}$, i.e. $\mathbf{q}_0^i \in \mathcal{U}$. Let $\mathbf{q}_k^i$ be a configuration in cluster U, and $\mathbf{q}_l^i$, $\mathbf{q}_m^i$ two configurations in cluster V. Then, the weight a_{UV} between cluster U and V is the 6-tuple

$$a_{UV} = \left\{\tau_k^l,\, \tau_l^m,\, g(V),\, \mathrm{d}(\tau_k^l),\, \mathrm{d}(\tau_l^m),\, f_{UV}\right\}, \qquad (2)$$

where

- τ_k^l denotes the path from $\mathbf{q}_k^i \in U$ to $\mathbf{q}_l^i \in V$, i.e. the path between cluster U and cluster V,
- τ_l^m denotes the shortest Hamiltonian path [57] including configurations of V, which starts at $\mathbf{q}_l^i$ and ends at $\mathbf{q}_m^i$,
- $g(V)$ is the gain of cluster V, as defined in (1),
- $\mathrm{d}(\tau_k^l)$ is the cost associated with the inter-cluster path τ_k^l, i.e. the length of τ_k^l,
- $\mathrm{d}(\tau_l^m)$ is the cost associated with the intra-cluster path τ_l^m, i.e. the length of τ_l^m,
- f_{UV} is the *utility function* defined as

$$f_{UV} = g(V) \exp\left(-\lambda_{\mathrm{tc}}\, \mathrm{d}(\tau_k^l) - \lambda_{\mathrm{ic}}\, \mathrm{d}(\tau_l^m)\right), \qquad (3)$$

where λ_{tc} and λ_{ic} are positive penalty terms for the inter-cluster and intra-cluster costs, respectively, which can be used to promote the visit of clusters far apart or large clusters. Their value depends on the motion capabilities of the agents (i.e. ground vs aerial robots). A similar utility function was originally proposed in [58].

The weights on the directed graph $\mathcal{G}$ in (2), quantify the potential benefit of choosing a certain path, to pursue the 3D reconstruction: in fact, the higher the value of the function f_{UV}, the more convenient is the path. Note that $f_{UV} > 0$, since $g(V) > 0$.

Our inspection problem can be stated as a *maximum Asymmetric Traveling Salesman Problem* (maxATSP), i.e. as the problem of finding a maximum-utility Hamiltonian path $\mathbf{p}$ on $\mathcal{G}$, see [7]. In a graph, a Hamiltonian path is an undirected or directed path that visits each vertex exactly once. In what follows, we will denote by maxATSP($\mathcal{U}$), the set function that takes the set of clusters $\mathcal{U}$ as input and outputs its utility value p, from which the path $\mathbf{p}$ can be computed. A linear programming formulation of maxATSP is

$$\begin{aligned}
\max \quad & \sum_{U \in \mathcal{U}} \sum_{V \in \mathcal{U}} f_{UV}\, x_{UV} \\
\text{s.t.} \quad & \\
& x_{UV} \in \{0,1\},\ U, V \in \mathcal{U},\ U \neq V, \\
& \sum_{U \in \mathcal{U},\, U \neq V} x_{UV} = 1, && V \in \mathcal{U} \setminus \{\mathbf{q}_0\}, \\
& \sum_{V \in \mathcal{U} \setminus \{\mathbf{q}_0\},\, V \neq U} x_{UV} \leq 1, && U \in \mathcal{U}, \\
& \sum_{V \in \mathcal{U} \setminus \{\mathbf{q}_0\},} x_{\{\mathbf{q}_0\} V} = 1, && \{\mathbf{q}_0\} \in \mathcal{U}, \\
& \sum_{U \in \mathcal{S}} \sum_{V \in \mathcal{S},\, \mathcal{V} \neq \mathcal{U}} x_{UV} \leq |\mathcal{S}| - 1, && \forall\, \mathcal{S} \subsetneq \mathcal{U},\ |\mathcal{S}| > 2,
\end{aligned}$$

where $\{\mathbf{q}_0\}$ denotes the cluster which only contains the initial configuration, $\mathcal{S}$ is a proper subset of $\mathcal{U}$, and $x_{UV} = 1$, if the edge belongs to the optimal path, and $x_{UV} = 0$, otherwise. maxATSP is solved by converting it into a symmetric TSP (i.e. a standard TSP) and then by using the classical Lin-Kernighan heuristic [59].

Let $\mathcal{U}^i$ be the set of clusters assigned to agent $i \in \{1, 2, \ldots, N\}$, such that $\bigcup_{i=1}^N \mathcal{U}^i = \mathcal{U}$. Then, the assignment problem can be stated as follows

$$\begin{aligned}
\max_{\mathcal{U}^1, \ldots, \mathcal{U}^N \subset \mathcal{U}} \Big\{ & \sum_{i=1}^N \mathrm{maxATSP}(\mathcal{U}^i) \,\Big|\, \mathcal{U}^i \cap \mathcal{U}^\ell = \emptyset, \\
& i \neq \ell,\ \bigcup_{i=1}^N \mathcal{U}^i = \mathcal{U} \Big\},
\end{aligned} \qquad (4)$$

where $\sum_{i=1}^N \mathrm{maxATSP}(\mathcal{U}^i)$ is a non-decreasing set function, and the space of feasible paths has the structure of a simple partition matroid [60]. Problem (4) can be approximately solved via local greedy heuristics (cf. [61]–[63]), which seek for the local maximum utility, based on an initial ranking of the items to assign. The centralized TSP-Greedy Allocation (TSGA) procedure [7] is reported in **Algorithm 1** and its distributed version (dist-TSGA) in **Algorithm 2** (see Sect. VI-C). Note that the single-agent algorithm is a special case of the centralized multi-agent algorithm with $N = 1$.

At each ISE extraction, the clusters are formed, and the shortest Euclidean distance to each agent is computed. These distances are then arranged in ascending order for the greedy heuristic. The TSGA planners greedily assign each cluster to an agent. More specifically, a cluster is assigned, when it locally maximizes the overall utility for the team. The path of agent i, e.g. the viewpoint sequence which results from the allocated clusters $\mathcal{U}^i$, is denoted by $\mathbf{p}_{\mathcal{U}^i}^i$, and the associated utility value by p^i. Once $\mathbf{p}_{\mathcal{U}^i}^i$ is computed, it is sent to the low-level planner. Unlike the classical insertion methods, in which a cluster is added to the path of a robot path [64], maxATSP is solved for the extended cluster set $\mathcal{U}^i \cup V$ with $V \in \mathcal{U} \setminus \mathcal{U}^i$. This strategy maximizes the individual utility of the agents over disjoint sets, so as to maximize, in turn, team-wise utility. Moreover, it is amenable to a distributed

implementation, since only local information is used (e.g. local free space, ISEs, $\mathcal{U}^i$ related to the map of agent i). On the long run, the maximization of the utility function pushes the agents towards the most valuable areas, in terms of completeness. For instance, this might prompt an agent to visit areas at the frontier between a known and an unknown surface containing multiple ISEs, and scan them all (cf. equation (1)).

B. Low-level planner

The low-level planner computes the path of an agent, using Lazy PRM* from the Open Motion Planning Library (OMPL) [65]. It leverages the path found by the TSGA planner (**Algorithm 1** or **2**) and the TSDF-map updates. Given a start and an end pose, it computes the path of a robot in the free space, given by the TSDF volume (on the plane for ground vehicles, and in the 3D space for aerial vehicles). By gathering all odometric information, each agent knows the position and orientation of the others, and it is then able to detect when another robot is near, when it faces it, or when it will cross its path. A *TrafficPolicy* function takes care of collision avoidance: for example, a robot might be asked to step aside to avoid a frontal impact, or to temporarily stop and wait until a teammate crossing its path, is outside its FOV. The function takes the robot's speed into account, and relies on a safety distance threshold for collision avoidance. On the other hand, the *ObstacleCheck* function triggers an emergency stop via a distance threshold to the surface (obtained from the TSDF), if a new obstacle is detected along the path.

Note that an ISE may be potentially scanned before the planned visit of an agent. To avoid unnecessary back-and-forth motions, the *UtilityCheck* function computes the remaining ISEs along the paths since the last planning iteration, and it waits for a possible update. This ensures a reactive visit of uncovered ISEs, as the map grows over time.

The high-level planner may generate paths of various length: hence, the agents may finish their tours at different time instants. When an agent has completed its current path, it continues the reconstruction asynchronously, with the latest path provided by the high-level planner.

C. Centralized vs distributed planning

In the centralized architecture, the map is directly generated from the depth maps sent by the agents. The centralized map is used for the extraction of the ISEs and the determination of the configuration clusters. A resolution of problem (4) on the base station allows to compute the paths $\mathbf{p}^1_{\mathcal{U}^1}, \dots, \mathbf{p}^N_{\mathcal{U}^N}$, which are broadcast to all the agents (cf. **Algorithm 1**).

Algorithm 1: TSP-Greedy Allocation (TSGA)

Set $\mathcal{U}^i = \emptyset$ and $p^i = 0$ for each agent $i \in \{1, 2, \dots, N\}$;
foreach *cluster* $V \in \mathcal{U}$ **do**
  $i \leftarrow \arg\max_{k \in \{1,2,\dots,N\}} \{\text{maxATSP}(\mathcal{U}^k \cup V) - p^k\}$;
  $\mathcal{U}^i \leftarrow \mathcal{U}^i \cup V$;
  $p^i \leftarrow \text{maxATSP}(\mathcal{U}^i)$;
  $\mathbf{p}^i_{\mathcal{U}^i} \leftarrow \{p^i, \mathcal{U}^i\}$;
Send paths $\mathbf{p}^1_{\mathcal{U}^1}, \dots, \mathbf{p}^N_{\mathcal{U}^N}$ to the agents (low-level planner);

Algorithm 2: dist-TSGA

Set $\widehat{\mathcal{U}}^i_i = \emptyset$ and $p^i_i = 0$ for agent $i \in \{1, 2, \dots, N\}$;
foreach *cluster* $V \in \widehat{\mathcal{U}}_i$ **do**
  $\ell \leftarrow \arg\max_{k \in \{1,2,\dots,N\}} \{\text{maxATSP}(\widehat{\mathcal{U}}^k_i \cup V) - p^k_i\}$;
  if $\ell = i$ **then**
    $\widehat{\mathcal{U}}^i_i \leftarrow \widehat{\mathcal{U}}^i_i \cup V$;
    $p^i_i \leftarrow \text{maxATSP}(\widehat{\mathcal{U}}^i_i)$;
    $\mathbf{p}^i_{\widehat{\mathcal{U}}^i_i} \leftarrow \{p^i_i, \widehat{\mathcal{U}}^i_i\}$;
  Send path $\mathbf{p}^i_{\widehat{\mathcal{U}}^i_i}$ to the low-level planner;

In the distributed architecture, due to material constraints (such as, saturation of communication network or packet losses), a newly-created patch may not be received by all the agents. Hence, in practice, the agents do not have the same knowledge of the full map. Nevertheless, provided that the patch request is frequent enough, agent i may have a rapid access to a full public map: it can thus update *its own local version of the global map* $\widehat{M}_i$, and synchronize it with the map of the other agents (see Sect. V-C and Fig. 4). Recalling that $\widehat{\mathcal{U}}_i$ denotes the set of all clusters generated from the ISEs extracted by agent i from its current global map $\widehat{M}_i$, let $\widehat{\mathcal{U}}^i_i \subset \widehat{\mathcal{U}}_i$ be the set of clusters assigned to agent i, such that $\bigcup_{j=1}^{N} \widehat{\mathcal{U}}^j_i = \widehat{\mathcal{U}}_i$. Based on the known set of clusters $\widehat{\mathcal{U}}_i$, each agent performs its own cluster allocation in a distributed way, to compute the best path. Once the assignment step is done successfully, agent i sends its path $\mathbf{p}^i_{\widehat{\mathcal{U}}^i_i}$ to the low-level planner (cf. **Algorithm 2**).

Because of possible inconsistencies in the global maps, multiple agents might be assigned to the same viewpoints of a cluster. To avoid such a scenario, the low-level planner broadcasts the path that is currently followed by an agent, to the others. In case of redundancy, the *RedundancyCheck* function of each low-level planner evaluates each agent's progress along the current path. The priority is given to the first agent which can reach the viewpoint, by taking its ranking in the list of visits and its current location, into account. The agents which were not granted priority, keep on moving until their last maneuver before the redundant cluster, and then wait for a new high-level path.

VII. Numerical Experiments

In this section, the centralized and distributed multi-robot systems are validated via extensive simulations with synthetic data. As a complement to our preliminary results in [6], the single-robot architecture is also compared with [8], using the simulation environment of the authors. Our baseline multi-agent planner, to test TSGA and dist-TSGA, is the Nearest Neighbor (NNB) greedy algorithm. In NNB, only one cluster is allocated to each robot by locally computing $\max_{V \in \mathcal{U}} f_{UV}$ for the updated map (cf. equation (3)). Re-planning is thus very fast compared to TSGA's, but only one cluster at a time is set to be visited. The different methods are evaluated in terms of map accuracy, surface completeness, total path length, and execution time.

1) Robots: RotorS simulator [66] has been used to model quadrotor UAVs equipped with a stereo camera, in the ROS-Gazebo environment. We considered 3 and 5 UAVs in our tests, and report the single-robot case previously studied in [6], for the sake of completeness. Each UAV has 4 degrees of freedom: its 3D position $[x,\, y,\, z]^T$ and its yaw angle ψ.

2) Simulation setup: We chose an industrial plant benchmark, which is well-known in the volumetric exploration literature [11]. The **Powerplant** model[4] (see Fig. 5-[top-left]) was scaled to fit in a $65\times42\times15\,\mathrm{m}^3$ box (as a consequence, the five flues have the same height). To study the impact of the two penalty terms in the utility function (3), on the reconstruction accuracy/completeness, we also considered the monumental Statue of Liberty[5] (**SoL**), of size $20\times20\times60\,\mathrm{m}^3$, see Fig. 5-[top-left]. The simulation parameters used in the two scenarios are reported in the first and second column of Table II.

To represent the depth-map uncertainty, we considered a Gaussian noise model. The standard deviation associated with a pixel, corresponds to the depth-value sensing error of the corresponding point located at a distance z, i.e. $\sigma(z) = \frac{|e_d|}{fB} z^2$, where $|e_d|$ is the magnitude of the disparity error, f is the focal length in pixels, and B is the baseline of the stereo camera on the UAVs, in meters. Following [54], [67], the raw depth map was blurred out by using a 3×3 kernel. In the single-UAV and centralized architecture, the TSDF volume was generated with the algorithm proposed in [68], that we adapted to multi-robot case. Our GPU-based algorithm allows to rapidly build and update (at about 1 Hz) the TSDF volume, as new depth maps sensed by robots are sent to the base station. Surface mesh reconstruction is performed with MarchingCubes [51], and the TSDF weight increment has been modified to be the inverse of squared distance, as reported in Sect. V-A. Distributed mapping is performed with the method proposed in [24], which allows each robot to compute its own local volume, and send it to other robots, so that a global map can be obtained. The event that triggers the integration and broadcast of a new patch to the other robots, is that 5 depth maps have been processed (cf. Sect. V-C). Unlike centralized mapping, distributed mapping is CPU-based, and it can be run on an embedded computer with limited resources. Lazy PRM* from OMPL [65] is used by the low-level planner (see Sect. VI-B) to compute collision-free paths for the UAVs (the collision radius is set to 1 m). The UAVs track the generated paths using Model Predictive Control [69], with a reference translational velocity fixed to 0.5 m/s.

The UAVs are initially located in the same area, around the base station (magenta dots in Fig. 5-[Columns 1 through 3]). The quantitative results of our numerical experiments are reported in Table III. The single-UAV architecture with perfect and noisy depth measurements (denoted by [6] and [6]*, respectively), has been used as a baseline, and compared with the centralized multi-robot architectures (with NNB and TSGA planners), and the distributed architecture (with the dist-TSGA planner), for a fleet of 3 and 5 UAVs. The last architecture has been only tested with **Powerplant**. To obtain statistically-significant values, 10 trials per architecture and per team of robots, were performed. For more details about the hardware platforms used in the simulations, the reader is referred to [70].

3) Metrics: The architectures have been evaluated in terms of cumulative path length and completion time (to fully cover the 3D environments). This includes travel time and sensing time (e.g., one depth map integration and map update). The reconstructed 3D surface has been evaluated with CloudCompare[6] using the M3C2 (Multiscale Model to Model Cloud Comparison) algorithm [71]. To assess how well the surface is covered, the reconstructed mesh is compared with a dense point cloud sampled on the ground truth (GT) mesh. The deviation is quantified via a cloud-to-mesh comparison (see Fig. 5-[4th column]). For a fair evaluation, all the invisible surfaces of the GT mesh were pruned beforehand (e.g., the interior floor and walls), and the analysis was restricted to the exterior surface mesh only. A point belonging to the GT point cloud is considered to be covered, if the shortest distance to this point along a normal to a mesh facet, is less than the length of the half diagonal of a voxel, i.e. $e_{\max} = r_{\mathbf{v}}\sqrt{3}/2$, where $r_{\mathbf{v}}$ is the voxel resolution. As a result, the more points are accurate, the better the coverage is. The quality of the recovered surface is evaluated in Table III (average and standard deviation of the signed distance error with respect to the GT point cloud and Root-Mean-Square Error).

4) Choice of penalty terms: The selection of penalty terms λ_{tc} and λ_{ic} in the utility function (3), depends on the nature of the 3D environment to explore. To find the combination of parameters which guarantees the shortest distance traveled, multiple reconstructions of **Powerplant** and **SoL** have been carried out with a single robot and different values of λ_{tc} and λ_{ic}. The results are compiled in Table IV, and indicate that the shape of the environment has a clear impact on the tuning of the penalty terms. In particular, in wide box-like environments as **Powerplant**, the majority of ISEs are uncovered near sharp edges or occluded regions, and tend to appear in groups separated by large layers of known space. To minimize the total path length, inter-cluster utility should then take priority over intra-cluster utility, i.e. $\lambda_{\mathrm{tc}} \gg \lambda_{\mathrm{ic}}$. On the other hand, the pedestal of the statue excluded, **SoL** predominantly consists of round surfaces and the average distance between two clusters is much smaller than in **Powerplant**.

[4] http://models.gazebosim.org/
[5] https://free3D.com/

TABLE II
PARAMETERS USED IN THE NUMERICAL EXPERIMENTS.

Parameter	**Powerplant**	**SoL**	**CB**
Voxel resolution $r_{\mathbf{v}}$ [m]	0.3	0.15	0.2
Threshold W_{th}	0.3	0.3	0.3
$e_{\max}$ [m]	0.2598	0.1299	0.1732
Camera range [m]	[1.6, 8]	[1, 5]	[1.5, 6]
Camera FOV [deg.] (H, V)	90 × 60	90 × 60	90 × 60
e_d [pixels]	0.1	0.1	0.1
f [pixels]	376	376	376
B [m]	0.11	0.11	0.11
Collision radius [m]	1	1	1
UAV nominal speed [m/s]	0.5	0.5	0.5
δ_{pose} [m]	4.7	3.6	2.5
d_ν [m]	2.0	2.5	1.5
$d_\nu^{\max}$ [m]	5	5	5
Penalty term λ_{tc}	0.3	0.17	0.25
Penalty term λ_{ic}	0.03	0.15	0.08

[6] https://danielgm.net/cc/

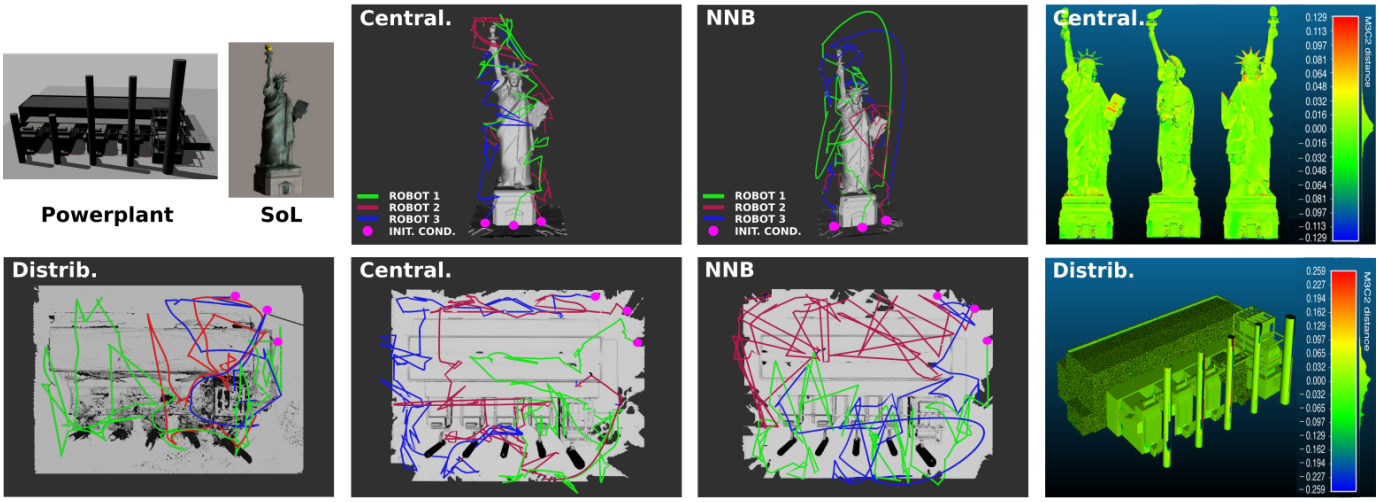


Fig. 5. *Numerical experiments*: [top-left] **Powerplant** and **SoL** models. Reconstructed meshes and 3D exploration paths $\mathbf{p}^1_{0,f}$, $\mathbf{p}^2_{0,f}$, $\mathbf{p}^3_{0,f}$ (green, red, blue) of the 3 UAVs for the two models, obtained with: [bottom-left] distributed architecture; [2nd column] centralized architecture; [3rd column] centralized architecture with NNB planner. The initial locations of the UAVs are marked with magenta dots. [4th column] Signed distance error: the color bar shows the error in meters with respect to the ground truth, computed with CloudCompare's M3C2 plugin.

TABLE III

RESULTS OF THE NUMERICAL EXPERIMENTS WITH **POWERPLANT**, **SOL** AND **CB** (STATISTICS OVER 10 TRIALS).

	Powerplant							
Number of UAVs	1		3			5		
Algorithm	[6]	[6]*	NNB	Central.	Distrib.	NNB	Central.	Distrib.
Path length [m]	780	785	1038	790	781	1113	879	872
Completion time [min.]	32	33′10″	11′09″	10′20″	9′56″	6′51″	6′22″	6′09″
Time gain [%] w.r.t. [6]*	–	–	66.4	68.8	70.0	79.4	80.8	81.5
Surface coverage [%]	91.5	90.4	90.0	91.0	95.9	90.5	90.6	95.6
M3C2 avg. error [cm]	0.14	−0.13	−0.15	−0.26	0.62	−0.11	−0.3	−0.73
M3C2 std. error [cm]	5.85	7.51	7.52	7.54	3.33	7.50	7.52	3.43
RMSE [cm]	5.86	7.51	7.52	7.55	3.39	7.50	7.52	3.51

	SoL						**CB**	
Number of UAVs	1		3		5		1	
Algorithm	[6]	[6]*	NNB	Central.	NNB	Central.	[8]	[6]
Path length [m]	547.0	550.0	733.0	721.0	580.0	574.2	641	632
Completion time [min.]	36′	37′	13′10″	10′18″	7′30″	6′45″	27′38″	25′20″
Time gain [%] w.r.t. [6]*	–	–	64.4	72.2	79.7	81.8	–	–
Surface coverage [%]	92.3	91.2	91.1	91.0	90.9	91.1	94.6	95.7
M3C2 avg. error [cm]	0.29	−0.02	−0.80	−0.03	0.06	−0.01	−1.06	0.28
M3C2 std. error [cm]	3.41	3.67	3.61	3.69	3.65	3.66	7.22	6.34
RMSE [cm]	3.43	3.67	3.69	3.69	3.65	3.66	7.29	6.34

Hence, similar penalty terms are preferable (i.e. $\lambda_{\text{tc}} \simeq \lambda_{\text{ic}}$).

5) Results: From an inspection of Table III, we can see that for a single UAV, the presence of noise has an impact on mesh accuracy, but that the navigation performance remains largely unaffected. We compared our single-robot algorithm with [8], by adapting our code to the simulator developed by the authors. We kept their default parameters and configurations, and selected the CityBuilding (**CB**) environment (see the third column of Table II). As we can notice in Table III, our method works slightly better than [8] in terms of completion time, path length, surface coverage, and accuracy. The algorithms described in [11], [40] exhibit a similar completion time with **Powerplant**. The deviation is more pronounced with **SoL**: in fact, the algorithm in [40] takes twice as long to finish the exploration. As the number of robots grows, the execution time decreases: in fact, with **Powerplant** (see the 4th row of Table III), the distributed (centralized) architecture with 3 UAVs is 70.0% (68.8%) faster, compared to the single-robot case. With 5 UAVs, the distributed (centralized) architecture is 81.5% (80.8%) faster, compared to a single quadrotor. Similarly, with **SoL**, TSGA achieves the task 72.2% (81.8%) faster with a fleet of 3 UAVs (5 UAVs), compared to the single-robot case. The distance traveled per UAV is shorter than that of a single UAV, but the total path length is larger, for any team of aerial vehicles.

TABLE IV

PENALTY TERMS AND PATH LENGTHS FOR **POWERPLANT** AND **SOL**.

	Powerplant				
λ_{tc}	0.35	0.3	0.15	0.03	0.01
λ_{ic}	0.01	0.03	0.15	0.3	0.35
Path length [m]	787	780	795	814	822
	SoL				
λ_{tc}	0.35	0.3	0.17	0.03	0.01
λ_{ic}	0.01	0.03	0.15	0.3	0.35
Path length [m]	578	559	550	587	601

TABLE V
COMPARISON BETWEEN THE CENTRALIZED TSGA AND NNB PLANNERS.

	Powerplant		**SoL**	
Number of UAVs	3	5	3	5
Path length gain [%]	23.90	21.00	1.64	1.00
Completion time gain [%]	7.32	7.06	21.80	10.00

The centralized TSGA also guarantees a shorter completion time and shorter distances compared to the classical NNB planner (see Table V), even with **SoL**. In fact, the profile of the statue and the presence of numerous contiguous ISEs should be more favorable, in principle, to fast local planners (for more details, the reader is referred to [7, Sect. V]). With **Powerplant**, the centralized architecture ensures that all the incomplete reachable regions are ultimately covered (surface coverage ranges between 90.4% and 91.5%). The overall reconstruction accuracy is better with the distributed mapping algorithm. Coverage improves as well, reaching 95.9% with 3 robots and 95.6% with 5 robots. Since the performance of MarchingCubes is dictated by the voxel size, the reconstructed mesh is more accurate, if the resolution is high. However, if the environment to explore is large, a high resolution entails resource-intensive mapping and ISE-extraction steps, which ultimately make the whole reconstruction process prohibitively expensive. Therefore, a balance between computational efficiency and reconstruction accuracy, should be found.

In summary, our numerical experiments show that the two multi-robot architectures are successful in scanning the 3D environments, covering upwards of 90% of their surface, even in the presence of noise in the depth measurements. In the next section, we will extend our analysis and study the accuracy and robustness of the centralized and distributed architectures deployed on mobile robots in real-life conditions.

VIII. REAL-WORLD EXPERIMENTS

In this section, which is organized as Sect. VII, the results of our hardware experiments are presented and discussed. To cope with variable environmental conditions (e.g. lighting change during the day), and slight differences in robot configuration (camera calibration, level of charge of the battery, etc.), a statistical analysis over multiple trials has been carried out.

1) Robots: The experiments have been conducted with a team of 4 identical Wifibots[7]. Each robot is equipped with an Intel NUC7i7BNH computer, a stereo rig with two IDS UI-1241-LE cameras (baseline $B = 26$ cm), and a Wi-Fi system to communicate with a ground station. An HQ camera (IDS UI-3252-LE) is also installed on each robot to generate a GT map (for more details on the software/hardware specifications, see [70]). The other parameters used in our experiments are reported in Table VI. Only minor changes have been made to adapt our ROS-based system to the real sensors and physical constraints of the Wifibots. In particular, the code that generated the emulated depth maps and odometry, has been replaced with validated software modules: the depth maps are computed with the ELAS algorithm [72], and the pose of the robots is estimated with the vision-based eVO algorithm [73]. This latter algorithm does not address the loop closure problem: hence, a localization drift, proportional to the distance traveled may occur. In the centralized architecture, the map is updated upon receipt of a new depth map (at around 1 Hz). Instead, in the distributed architecture, a new patch is stored and broadcast to the robots, every time that 5 depth maps have been integrated into the current TSDF map. The mapping and planning modules of the centralized and distributed architectures are identical to those presented in Sect. VII-.2.

2) Environments: Two different indoor environments have been considered in our experiments. An $8 \times 7 \times 2$ m^3 **Test arena**, consisting of a central obstacle surrounded by four walls covered by mattresses (green in Fig. 6-[top-left]), and an underground $21 \times 14 \times 2$ m^3 **Parking lot**, containing several obstacles at ground level. The 2D maps and photos of these environments are shown in Fig. 6.

3) Metrics: The same metrics as in the numerical experiments, have been considered (please refer to Sect. VII-.3). Data exchange has been monitored during the experiments: in the single-robot and centralized multi-robot algorithms, it pertains to depth maps, odometry and path messages transmitted between the ground station and the vehicles, whereas in the distributed algorithm, to patches, odometry, and paths exchanged by the robots.

4) Results: Table VII summarizes our experimental results. For **Test arena**, the reconstruction has been performed with a single robot, and a team of 2 robots for the centralized and distributed architectures. On the other hand, for **Parking lot**, we considered a single robot and teams of 2, 3 and 4 robots for the centralized architecture, and a team of 2 robots for the distributed architecture. For each environment/team, we carried out 5 trials with identical initial positions and orientations for the robots. Fig. 7 shows different snapshots of the 3D reconstruction of **Parking lot**, obtained with the centralized architecture. The GT, reconstructed mesh, and signed distance errors for the distributed architecture, are reported in Fig. 8 (left, center, and right, respectively). The single-robot case is considered as a reference, in both environments. From Table VII, we can see that as the number of robots grows, the completion time decreases while the cumulative path length (at the team level), increases. Nevertheless, taken individually,

TABLE VI
PARAMETERS USED IN THE REAL-WORLD EXPERIMENTS.

Parameter	**Test arena**	**Parking lot**
Voxel resolution $r_{\mathbf{v}}$ [m]	0.1	0.2
Threshold W_{th}	0.3	0.3
$e_{\max}$ [m]	0.0866	0.1732
Camera range [m]	[0.3, 5]	[0.3, 5]
Camera FOV [deg.] (H, V)	90×60	90×60
Collision radius [m]	0.55	0.55
Robot nominal speed [m/s]	0.5	0.5
δ_{pose} [m]	1.3	1.3
d_{ν} [m]	2.0	2.0
$d_{\nu}^{\max}$ [m]	3.5	3.5
Penalty term λ_{tc}	0.5	0.7
Penalty term λ_{ic}	0.01	0.01

[7] https://wifibot.com/

Fig. 6. *Real-world experiments*: [top] **Test arena**, [bottom] **Parking lot**. [left] 2D maps, and [center, right] photos of the two environments, including two panoramic views of **Test arena**. The circled numbers indicate obstacles or areas of interest.

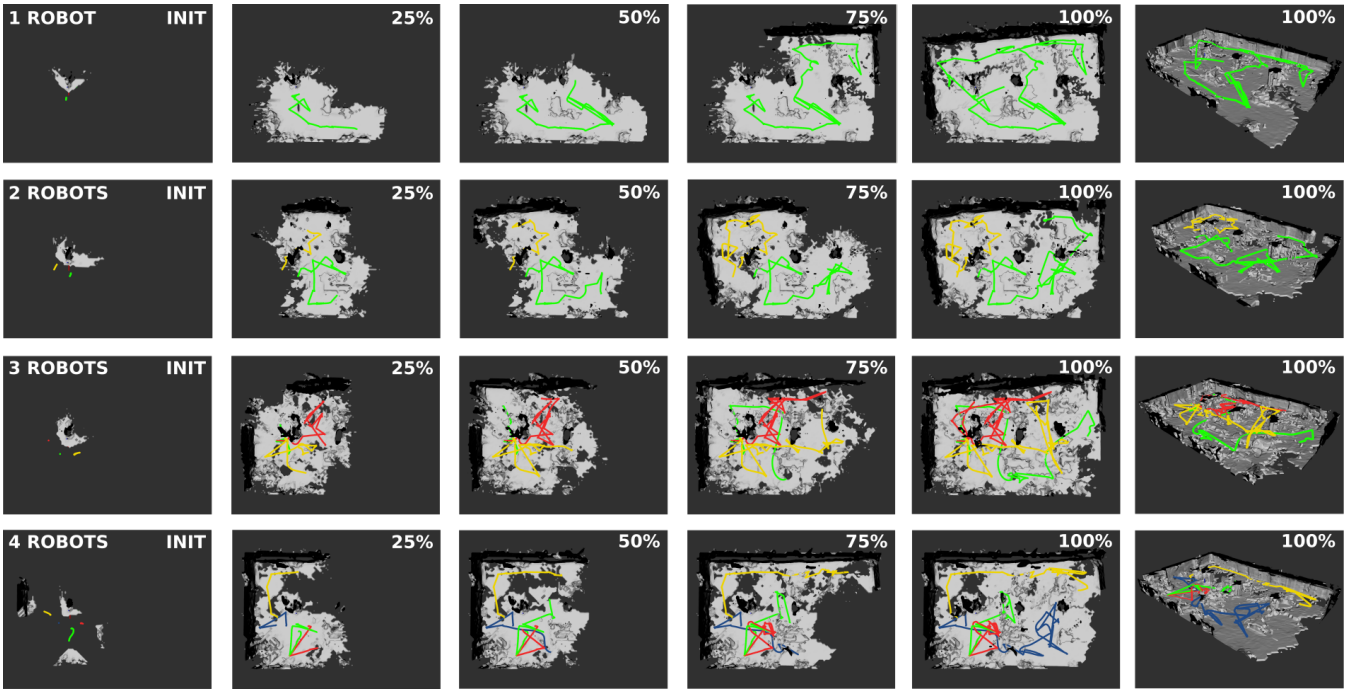


Fig. 7. *Real-world experiments*: **Parking lot**, centralized architecture. From left to right, time progression expressed in percentage of final 3D reconstruction (top view). In the rows, the number of robots varies between 1 and 4. The last column reports an isometric view of reconstructed mesh and the exploration paths of the robots. The ceiling of the parking lot has been removed to provide visibility of the interior.

the distance traveled by each robot, decreases. For example, in **Parking lot**, a team of 4 robots allows to reduce the completion time by 31.1%, compared to the single-robot case. Doubling the number of robots, the gain in completion time is 6.25%, in **Test arena**. The distributed architecture works just as well as the centralized one in terms of distance traveled and time to completion. However, the volume of data exchanged by the robots using the distributed architecture, in "nominal operation", is smaller (3.031 GB vs 6.029 GB for the centralized case). In fact, during our experiments, additional information (mainly meshes) was transmitted on the communication network, to monitor the progression of the robots. This resulted in an 87.5% increase in the volume of data exchanged (24.111 GB), which tended to saturate the network. To circumvent this problem, simpler spatial representations (such as, TSDF maps) could be used for visualization, which is a priority area for future research. As an indication on the scalability of the proposed algorithms, the last row of Table VII, also reports the average bandwidth usage.

In the two environments, the robots were left free to cover the entire accessible area. In the single-robot case, the reconstructed mesh covered 89.1% of the surface of **Parking lot** for a voxel resolution $r_{\mathbf{v}} = 20$ cm and an admissible error $e_{\max} = 17.32$ cm. For **Test arena**, instead, the surface coverage was 85.6% with $r_{\mathbf{v}} = 10$ cm and $e_{\max} = 8.66$ cm. Differently from the numerical experiments (cf. Sect. VII-.5), as the number of robots increases, surface coverage decreases, until reaching the 73% level with 4 robots (**Parking lot**), and the 80.3% level

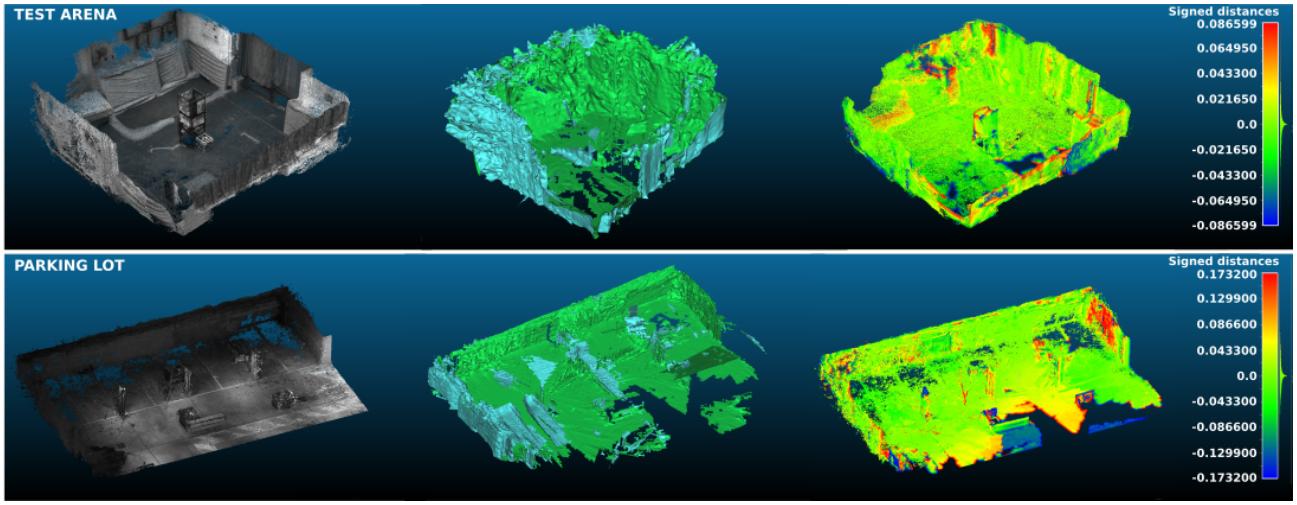


Fig. 8. *Real-world experiments*: [top] **Test arena**, [bottom] **Parking lot**. [left] GT point cloud. [center] Reconstructed mesh using the distributed architecture with 2 robots. [right] Signed distance error: the color bar shows the error in meters with respect to the GT, computed with CloudCompare's M3C2 plugin.

TABLE VII

RESULTS OF THE REAL-WORLD EXPERIMENTS (STATISTICS OVER 5 TRIALS).

	Test arena			**Parking lot**				
Number of robots	1	2		1	2		3	4
Architecture	–	Central.	Distrib.	–	Central.	Distrib.	Central.	Central.
Cumulative path length [m]	20.6	27.0	24.2	106.0	113.6	107.1	134.2	161.0
Completion time [min.]	$2'53''$	$2'45''$	$2'37''$	$13'56''$	$11'06''$	$10'41''$	$10'14''$	$9'36''$
Time gain [%] w.r.t. single robot	–	6.25	9.25	–	20.30	23.30	26.60	31.10
Surface coverage [%]	85.6	80.3	91.0	89.1	81.1	88.4	76.4	73.0
M3C2 avg. error [cm]	0.27	0.40	0.12	0.01	−0.48	0.07	−0.18	0.61
M3C2 std. error [cm]	3.99	4.16	3.33	8.37	8.29	5.85	8.93	8.97
RMSE [cm]	4.00	4.18	3.33	8.37	8.30	5.85	8.94	8.99
Data exchanged [GB]	0.750	0.888	0.501	4.849	6.029	3.031	10.715	13.271
Bandwidth [Mb/s]	35.51	44.09	26.14	47.52	74.16	38.74	142.96	188.74

with 2 robots (**Test arena**). In fact, a depth map depends on the pose estimated by a visual odometry algorithm, which is prone to drift. The estimation error due to the drift, has an impact on the TSDF volume when the depth map is integrated. Hence, accumulation of errors is experienced, as the number of robots increases: the overall mesh accuracy decreases, more outliers need to be pruned, and less free space is covered. The RMSE ranges between 8.37 cm and 8.99 cm for **Parking lot**, and between 4 cm and 4.18 cm for **Test arena**. The distributed mapping outperform the centralized mapping in terms of surface coverage (88.4% vs 80.3%). This depends on the superior accuracy of the distributed mapping algorithm compared to the centralized one, for a given resolution. It is also worth mentioning that the fusion policy in the distributed case (cf. Sect. V-C), superimposes the TSDF patches with an integration rule which prioritizes those which have maximum weights, thus mitigating the impact of depth noise. Fig. 7 and Fig. 8 show that the centralized and distributed algorithms provide accurate 3D reconstruction, with a decent surface coverage despite the odometry drift.

Finally, the difference in speed-up observed in the numerical and real-world experiments with an increasing number of robots, can be ascribed to the different specifications of the robotic platforms and set-ups considered. In fact, the motion of the UAVs in the 3D space, is far less constrained than that of the Wifibots on the ground.

IX. CONCLUSIONS AND FUTURE WORK

In this paper, we have described a new multi-robot system for surface inspection of large-scale unknown 3D environments. The proposed approach relies on Next-Best-View planning to address the coordinated inspection problem, and directly exploits the 3D surface representation. Centralized and distributed architectures (TSGA and dist-TSGA) have been developed and analyzed in detail. To illustrate and validate our algorithms, we performed extensive numerical simulations with quadrotor UAVs using two challenging ROS-Gazebo 3D models, and real-world experiments with up to 4 wheeled robots in two indoor environments. The simulation results indicate that our solution is competitive with the state-of-the-art in terms of navigation and reconstruction accuracy, and that it can be easily tailored to different software stacks. The experimental results complement the tests with synthetic data, and confirm that our approach is versatile with different types of robots and environments, and effective in generating accurate meshes in real-time.

The experiments also provide evidence that odometry drift, via uncertainty propagation, has a non-negligible impact on the overall reconstruction process. To circumvent this problem, one could envisage a hybrid approach which takes advantage of a SLAM landmark map in the planning module. This would endow the robots with loop-closure capabilities,

thus minimizing the effect of drift and ultimately boosting the map accuracy. In future works, we also place a premium on a more efficient mechanism to monitor the progression of the robots, and we plan to perform real-world experiments with larger teams of heterogeneous agents, in obstacle-rich dynamic environments.

MULTIMEDIA MATERIAL

The supplementary material accompanying this article, is a video which presents a selection of numerical simulations and real-world experiments reported in Sect. VII and Sect. VIII.

REFERENCES

[1] N. Leonard, D. Paley, F. Lekien, R. Sepulchre, D. Fratantoni, and R. Davis, "Collective motion, sensor networks, and ocean sampling," *Proc. IEEE*, vol. 95, no. 1, pp. 48–74, 2007.

[2] A. Alcántara, J. Capitán, R. Cunha, and A. Ollero, "Optimal trajectory planning for cinematography with multiple Unmanned Aerial Vehicles," *Robot. Autonom. Syst.*, vol. 140, p. 103778, 2021.

[3] K. Shah, G. Ballard, A. Schmidt, and M. Schwager, "Multidrone aerial surveys of penguin colonies in Antarctica," *Sci. Robot.*, vol. 5, no. 47, p. eabc3000, 2020.

[4] V. Digani, L. Sabattini, C. Secchi, and C. Fantuzzi, "Ensemble Coordination Approach in Multi-AGV Systems Applied to Industrial Warehouses," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 3, pp. 922–934, 2015.

[5] I. Miller, F. Cladera, A. Cowley, S. Shivakumar, E. Lee, L. Jarin-Lipschitz, A. Bhat, N. Rodrigues, A. Zhou, A. Cohen, A. Kulkarni, J. Laney, C. Taylor, and V. Kumar, "Mine Tunnel Exploration Using Multiple Quadrupedal Robots," *IEEE Robot. Autonom. Lett.*, vol. 5, no. 2, pp. 2840–2847, 2020.

[6] G. Hardouin, F. Morbidi, J. Moras, J. Marzat, and E. Mouaddib, "Surface-driven Next-Best-View planning for exploration of large-scale 3D environments," in *Proc. 21st IFAC World Congress*, 2020, pp. 15 501–15 507.

[7] G. Hardouin, J. Moras, F. Morbidi, J. Marzat, and E. Mouaddib, "Next-Best-View planning for surface reconstruction of large-scale 3D environments with multiple UAVs," in *Proc. IEEE/RSJ Int. Conf. Intel. Robots Syst.*, 2020, pp. 1567–1574.

[8] 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. Autonom. Lett.*, vol. 5, no. 2, pp. 1500–1507, 2020.

[9] L. Yoder and S. Scherer, "Autonomous Exploration for Infrastructure Modeling with a Micro Aerial Vehicle," in *Field and Service Robotics: Results of the 10th Int. Conf.*, D. Wettergreen and T. Barfoot, Eds. Springer, 2016, pp. 427–440.

[10] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart, "Receding horizon path planning for 3D exploration and surface inspection," *Auton. Robot.*, vol. 42, no. 2, pp. 291–306, 2018.

[11] S. Song and S. Jo, "Surface-based Exploration for Autonomous 3D Modeling," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 4319–4326.

[12] Y. Kompis, L. Bartolomei, R. Mascaro, L. Teixeira, and M. Chli, "Informed Sampling Exploration Path Planner for 3D Reconstruction of Large Scenes," *IEEE Robot. Autonom. Lett.*, vol. 6, no. 4, pp. 7893–7900, 2021.

[13] A. Mannucci, S. Nardi, and L. Pallottino, "Autonomous 3D Exploration of Large Areas: A Cooperative Frontier-Based Approach," in *Proc. 4th Int. Conf. Model. Simul. Auton. Syst.*, 2017, pp. 18–39.

[14] M. Corah and N. Michael, "Volumetric Objectives for Multi-Robot Exploration of Three-Dimensional Environments," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 9043–9050.

[15] M. Lauri, J. Pajarinen, J. Peters, and S. Frintrop, "Multi-Sensor Next-Best-View Planning as Matroid-Constrained Submodular Maximization," *IEEE Robot. Autonom. Lett.*, vol. 5, no. 4, pp. 5323–5330, 2020.

[16] A. Hornung, K. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Auton. Robot.*, vol. 34, no. 3, pp. 189–206, 2013.

[17] R. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon, "KinectFusion: Real-time dense surface mapping and tracking," in *Proc. 10th IEEE Int. Symp. Mixed Augmen. Real.*, vol. 11, 2011, pp. 127–136.

[18] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proc. 23rd Annual Conf. Comput. Graph. Interact. Tech.*, 1996, pp. 303–312.

[19] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3D reconstruction at scale using voxel hashing," *ACM Trans. Graphic*, vol. 32, no. 6, pp. 1–11, 2013, article no. 169.

[20] M. Klingensmith, I. Dryanovski, S. Srinivasa, and J. Xiao, "CHISEL: Real Time Large Scale 3D Reconstruction Onboard a Mobile Device using Spatially Hashed Signed Distance Fields," in *Proc. Robotics: Science and Systems XI*, vol. 4, no. 1, 2015, article n. 40.

[21] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3D Euclidean Signed Distance Fields for on-board MAV planning," in *Proc. IEEE/RSJ Int. Conf. Intel. Robots Syst.*, 2017, pp. 1366–1373.

[22] A. Howard, "Multi-robot mapping using manifold representations," in *Proc. IEEE Int. Conf. Robot. Automat.*, vol. 4, 2004, pp. 4198–4203.

[23] A. Millane, Z. Taylor, H. Oleynikova, J. Nieto, R. Siegwart, and C. Cadena, "C-blox: A Scalable and Consistent TSDF-based Dense Mapping Approach," in *Proc. IEEE/RSJ Int. Conf. Intel. Robots Syst.*, 2018, pp. 995–1002.

[24] T. Duhautbout, J. Moras, and J. Marzat, "Distributed 3D TSDF Manifold Mapping for Multi-Robot Systems," in *Proc. 9th Europ. Conf. Mobile Robots*, 2019.

[25] S. LaValle and J. Kuffner Jr, "Randomized Kinodynamic Planning," *Int. J. Robot. Res.*, vol. 20, no. 5, pp. 378–400, 2001.

[26] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, 2011.

[27] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Autom.*, vol. 12, no. 4, pp. 566–580, 1996.

[28] K. Hauser, "Lazy Collision Checking in Asymptotically-Optimal Motion Planning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2015, pp. 2951–2957.

[29] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart, "Receding Horizon "Next-Best-View" Planner for 3D Exploration," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 1462–1468.

[30] C. Papachristos, M. Kamel, M. Popović, S. Khattak, A. Bircher, H. Oleynikova, T. Dang, F. Mascarich, K. Alexis, and R. Siegwart, "Autonomous Exploration and Inspection Path Planning for Aerial Robots Using the Robot Operating System," in *Robot Operating System (ROS)*, A. Koubaa, Ed. Springer, 2019, vol. 778, pp. 67–111.

[31] M. Selin, M. Tiger, D. Duberg, F. Heintz, and P. Jensfelt, "Efficient Autonomous Exploration Planning of Large-Scale 3-D Environments," *IEEE Robot. Autonom. Lett.*, vol. 4, no. 2, pp. 1699–1706, 2019.

[32] A. Batinovic, T. Petrovic, A. Ivanovic, F. Petric, and B. Stjepan, "A Multi-Resolution Frontier-Based Planner for Autonomous 3D Exploration," *IEEE Robot. Autonom. Lett.*, vol. 6, no. 3, pp. 4528–4535, 2021.

[33] V. M. Respall, D. Devitt, R. Fedorenko, and A. Klimchik, "Fast Sampling-based Next-Best-View Exploration Algorithm for a MAV," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 89–95.

[34] Z. Xu, D. Deng, and K. Shimada, "Autonomous UAV Exploration of Dynamic Environments Via Incremental Sampling and Probabilistic Roadmap," *IEEE Robot. Autonom. Lett.*, vol. 6, no. 2, pp. 2729–2736, 2021.

[35] J. Bresenham, "Algorithm for computer control of a digital plotter," *IBM Syst. J.*, vol. 4, no. 1, pp. 25–30, 1965.

[36] C. Connolly, "The Determination of Next Best Views," in *Proc. IEEE Int. Conf. Robot. Automat.*, vol. 2, 1985, pp. 432–435.

[37] J. Vasquez-Gomez, L. Sucar, R. Murrieta-Cid, and E. Lopez-Damian, "Volumetric Next-best-view Planning for 3D Object Reconstruction with Positioning Error," *Int. J. Adv. Robot. Syst.*, vol. 11, no. 10, 2014.

[38] R. Border, J. Gammell, and P. Newman, "Surface Edge Explorer (SEE): Planning Next Best Views Directly from 3D Observations," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 6116–6123.

[39] R. Monica and J. Aleotti, "Contour-based next-best view planning from point cloud segmentation of unknown objects," *Auton. Robot.*, vol. 42, no. 2, pp. 443–458, 2018.

[40] S. Song and S. Jo, "Online Inspection Path Planning for Autonomous 3D Modeling using a Micro-Aerial Vehicle," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 6217–6224.

[41] S. Song, D. Kim, and S. Choi, "View Path Planning via Online Multiview Stereo for 3-D Modeling of Large-Scale Structures," *IEEE Trans. Robot.*, vol. 38, no. 1, pp. 372–390, 2022.

[42] F. Amigoni and A. Gallo, "A Multi-Objective Exploration Strategy for Mobile Robots," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2005, pp. 3850–3855.

[43] W. Burgard, M. Moors, C. Stachniss, and F. Schneider, "Coordinated multi-robot exploration," *IEEE Trans. Robot.*, vol. 21, no. 3, pp. 376–386, 2005.

[44] M. Juliá, A. Gil, and O. Reinoso, "A comparison of path planning strategies for autonomous exploration and mapping of unknown environments," *Auton. Robot.*, vol. 33, no. 4, pp. 427–444, 2012.

[45] G. Best, O. Cliff, T. Patten, R. Mettu, and R. Fitch, "Decentralised Monte Carlo Tree Search for Active Perception," in *Proc. 12th Int. Workshop Algor. Found. Robot.*, 2016, article n. 50.

[46] N. Atanasov, J. Le Ny, K. Daniilidis, and G. Pappas, "Decentralized active information acquisition: Theory and application to multi-robot SLAM," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2015, pp. 4775–4782.

[47] B. Charrow, S. Liu, V. Kumar, and N. Michael, "Information-theoretic mapping using Cauchy-Schwarz Quadratic Mutual Information," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2015, pp. 4791–4798.

[48] M. Corah, C. OMeadhra, K. Goel, and N. Michael, "Communication-Efficient Planning and Mapping for Multi-Robot Exploration in Large Environments," *IEEE Robot. Autonom. Lett.*, vol. 4, no. 2, pp. 1715–1721, 2019.

[49] M. Corah and N. Michael, "Distributed matroid-constrained submodular maximization for multi-robot exploration: Theory and practice," *Auton. Robot.*, vol. 43, no. 2, pp. 485–501, 2019.

[50] G.-B. Chaslot, "Monte-Carlo Tree Search," Ph.D. dissertation, Maastricht University, September 2010.

[51] W. Lorensen and H. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," in *Proc. ACM SIGGRAPH Comp. Graph.*, vol. 21, no. 4, 1987, pp. 163–169.

[52] J. Engel, J. Stückler, and D. Cremers, "Large-scale direct SLAM with stereo cameras," in *Proc. IEEE/RSJ Int. Conf. Intel. Robots Syst.*, 2015, pp. 1935–1942.

[53] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras," *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, 2017.

[54] C. Nguyen, S. Izadi, and D. Lovell, "Modeling Kinect Sensor Noise for Improved 3D Reconstruction and Tracking," in *Proc. 2nd IEEE Int. Conf. 3D Imag. Model. Proc. Visual. & Transm.*, 2012, pp. 524–530.

[55] H. Oleynikova, C. Lanegger, Z. Taylor, M. Pantic, A. Millane, R. Siegwart, and J. Nieto, "An opensource system for visionbased microaerial vehicle mapping, planning, and flight in cluttered environments," *J. Field Robot.*, vol. 37, no. 4, pp. 642–666, 2020.

[56] S. Kriegel, C. Rink, T. Bodenmüller, and M. Suppa, "Efficient next-best-scan planning for autonomous 3D surface reconstruction of unknown objects," *J. Real-Time Image Pr.*, vol. 10, no. 4, pp. 611–631, 2015.

[57] C. Godsil and G. Royle, *Algebraic Graph Theory*. Springer, 2001.

[58] H. González-Banos and J.-C. Latombe, "Navigation strategies for exploring indoor environments," *Int. J. Robot. Res.*, vol. 21, no. 10-11, pp. 829–848, 2002.

[59] K. Helsgaun, "An effective implementation of the Lin–Kernighan traveling salesman heuristic," *Eur. J. Oper. Rer.*, vol. 126, no. 1, pp. 106–130, 2000.

[60] R. Wilson, "An Introduction to Matroid Theory," *Am. Math. Mon.*, vol. 80, no. 5, pp. 500–525, 1973.

[61] A. Bian, J. Buhmann, A. Krause, and S. Tschiatschek, "Guarantees for Greedy Maximization of Non-submodular Functions with Applications," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 498–507.

[62] G. Nemhauser, L. Wolsey, and M. Fisher, "An Analysis of Approximations for Maximizing Submodular Set Functions I," *Math. Program.*, vol. 14, pp. 265–294, 1978.

[63] M. Fisher, G. Nemhauser, and L. Wolsey, "An Analysis of Approximations for Maximizing Submodular Set Functions II," *Math. Program. Stud.*, vol. 8, pp. 73–87, 1978.

[64] S. T. Jawaid and S. L. Smith, "Informative path planning as a maximum traveling salesman problem with submodular rewards," *Discrete Appl. Math.*, vol. 186, pp. 112–127, 2015.

[65] I. Şucan, M. Moll, and L. Kavraki, "The Open Motion Planning Library," *IEEE Rob. Autom. Mag.*, vol. 19, no. 4, pp. 72–82, 2012.

[66] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, "RotorS – A Modular Gazebo MAV Simulator Framework," in *Robot Operating System (ROS)*, A. Koubaa, Ed. Springer, 2016, vol. 625, pp. 595–625.

[67] L. Keselman, J. Woodfill, A. Grunnet-Jepsen, and A. Bhowmik, "Intel RealSense Stereoscopic Depth Cameras," in *Proc. IEEE Conf. Comp. Vis. Pattern Recogn. Workshops*, 2017, pp. 1–10.

[68] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, and T. Funkhouser, "3DMatch: Learning Local Geometric Descriptors from RGB-D Reconstructions," in *Proc. IEEE Conf. Comp. Vis. Pattern Recogn.*, 2017, pp. 1802–1811.

[69] M. Kamel, T. Stastny, K. Alexis, and R. Siegwart, "Model Predictive Control for Trajectory Tracking of Unmanned Aerial Vehicles Using Robot Operating System," in *Robot Operating System (ROS)*, A. Koubaa, Ed. Springer, 2017, vol. 707, pp. 3–39.

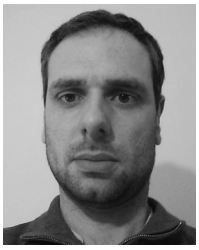
[70] G. Hardouin, "A centralized and distributed multi-robot system for 3D surface reconstruction of unknown environments," Ph.D. dissertation, Université de Picardie Jules Verne, Amiens, France, March 2022.

[71] D. Lague, N. Brodu, and J. Leroux, "Accurate 3D comparison of complex topography with terrestrial laser scanner: Application to the Rangitikei canyon (N-Z)," *ISPRS J. photogramm.*, vol. 82, pp. 10–26, 2013.

[72] A. Geiger, M. Roser, and R. Urtasun, "Efficient large-scale stereo matching," in *Proc. Asian Conf. Comp. Vis.*, 2010, pp. 25–38.

[73] M. Sanfourche, V. Vittori, and G. Le Besnerais, "eVO: A realtime embedded stereo odometry for MAV applications," in *Proc. IEEE/RSJ Int. Conf. Intel. Robots Syst.*, 2013, pp. 2107–2114.

Guillaume Hardouin received the Ph.D. degree in Robotics from the Université de Picardie Jules Verne, France, in 2022. He was an embedded software engineer with PSA Group, Vélizy-Villacoublay, France, in 2016. He is currently a research engineer with Naval Group, Saint-Tropez, France. His research interests include autonomous mobile robots and network systems.



Julien Moras was born in Vénissieux, France, in 1985. He received the engineer's degree from the ISAE in 2008 and the Ph.D. degree in Information Technologies and Systems from the Université de Technologie de Compiègne, France in 2013. Since 2013, he has been a research engineer with the French Aerospace Laboratory, ONERA, Palaiseau. His research interests include data fusion theory in wireless sensor networks and robotic embedded perception for autonomous navigation.

Fabio Morbidi (Senior Member, IEEE) received the Ph.D. degree in Control Engineering and Robotics from the University of Siena, Italy, in 2009. He was a visiting researcher at the University of California, Santa Barbara, USA, in 2008, and he held post-doctoral positions at Northwestern University, University of Texas at Arlington, USA, and at Inria Grenoble Alpes, France. Since 2014, he has been an Associate Professor of Robotics with the Université de Picardie Jules Verne, France. He currently serves as Associate Editor for the IEEE TRANSACTIONS ON ROBOTICS and for the IEEE ROBOTICS AND AUTOMATION LETTERS. His main research interests include network systems and robotic vision.

Julien Marzat graduated as an engineer from ENSEM (INPL Nancy) in 2008, then completed his Ph.D Thesis in 2011 and Habilitation (HDR) in 2019 both from the Université Paris-Saclay, France. He is currently a Research Director at ONERA, with interests in guidance, control and fault diagnosis for autonomous robots and aerospace systems.

El Mustapha Mouaddib received the Ph.D. degree in Robotics and the Habilitation degree from the Université de Picardie Jules Verne, France, in 1991 and 1999, respectively. Since 2001, he has been a Full Professor with the same university. From 1995 to 2022, he was the head of the Robotic Perception group, and he was involved in research projects on omnidirectional vision and structured light. Since 2010, he has been the leader of E-Cathedral, a multidisciplinary research initiative on digital heritage. He has served as Associate Editor for the Conference Board of IEEE ICRA and IEEE/RSJ IROS, and for the IEEE ROBOTICS AND AUTOMATION LETTERS (2015-2018). His main research interests are in computer vision and artificial perception for mobile robotics.