HAL
open science

Inspection of Ship Hulls with Multiple UAVs: Exploiting Prior Information for Online Path Planning

Pasquale Grippa, Alessandro Renzaglia, Antoine Rochebois, Melanie Schranz, Olivier Simonin

To cite this version:

Pasquale Grippa, Alessandro Renzaglia, Antoine Rochebois, Melanie Schranz, Olivier Simonin. Inspection of Ship Hulls with Multiple UAVs: Exploiting Prior Information for Online Path Planning. IROS 2022 - IEEE/RSJ International Conference on Intelligent Robots and Systems, Oct 2022, Kyoto, Japan. hal-03776012

HAL Id: hal-03776012

https://inria.hal.science/hal-03776012v1

Submitted on 16 Sep 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

This is the authors' version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record will be published in Proc. IEEE IROS 2022. DOI will follow as soon as possible.

Inspection of Ship Hulls with Multiple UAVs: Exploiting Prior Information for Online Path Planning

Pasquale Grippa[a], Alessandro Renzaglia[b], Antoine Rochebois[b], Melanie Schranz[a] and Olivier Simonin[b]

***Abstract*— This paper addresses a path planning problem for a fleet of Unmanned Aerial Vehicles (UAVs) that uses both prior information and online gathered data to efficiently inspect large surfaces such as ship hulls and water tanks. UAVs can detect corrosion patches and other defects on the surface from low-resolution images. If defects are detected, they get closer to the surface for a high-resolution inspection. The prior information provides expected defects locations and is affected by both false positives and false negatives. The mission objective is to prioritize the close-up inspection of defected areas while keeping a reasonable time for the coverage of the entire surface. We propose two solutions to this problem: a coverage algorithm that divides the problem into a set of Traveling Salesman Problems (Part-TSP) and a cooperative frontier approach that introduces frontier utilities to incorporate the prior information (Coop-Frontier). We finally provide extensive simulation results to analyze the performance of these approaches and compare them with alternative solutions. These results suggest that both Part-TSP and Coop-Frontier perform better than the baseline solution. Part-TSP has the best performance in most cases. However, coop-Frontier is preferable in extreme cases because more robust to inhomogeneous corrosion distribution and imperfect information.**

I. Introduction

Periodic service and maintenance of large structures are increasingly important tasks where cooperative autonomous robots can have a crucial role in helping humans to improve performances and reduce costs. There are approximately 56000 medium and large ships (larger than 500 tons) worldwide[1] that require periodic inspection. In the context of the European project BugWright2 [1], we aim at combining the survey capabilities of autonomous Unmanned Aerial Vehicles (UAVs) and small Autonomous Underwater Vehicles (AUV) with magnetic-wheeled crawlers operating on the surface of the structure.

In this work, we focus on path planning for a fleet of UAVs (typically between 4 and 8) performing a visual inspection mission to detect corrosion patches with minimal user intervention. The goal is to provide the operator with the location and details of existing corroded areas. To this end, UAVs perform a non-uniform coverage of a vertical surface. They fly in two planes at different distances from the surface of interest: a medium distance ζ_2 to detect corrosion on large areas and a close-up distance ζ_1 to inspect corrosion patches with higher resolution (see Fig. 1). In particular, we study partially informed scenarios, where a partial and imperfect map of the expected corrosion position is initially available. The goal is to exploit this information to give higher priority to the inspection of corroded areas while maintaining the guarantee of an efficient inspection of the entire surface.

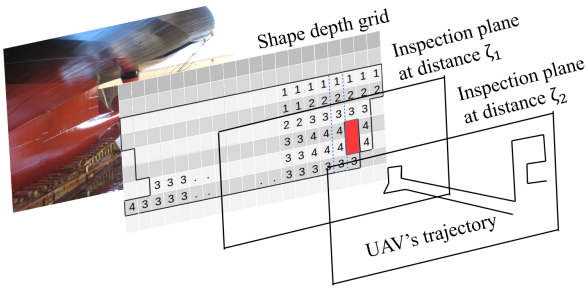

Fig. 1: Hull modeling and UAV's trajectory in the 2.5D grid.

To cope with this problem, we investigate two different solutions: one based on a pre-partition of the environment to efficiently distribute the work among the UAVs (Part-TSP), and a second fully-cooperative solution that provides higher robustness with respect to robots failures (Coop-Frontier). More specifically, the main contributions of this paper can be summarized as follows:

- First, we propose an inspection approach that divides the problem into a set of Traveling Salesman Problems. The approach considers prior information and re-computes paths online when new information is disclosed.
- Second, we present a fully-cooperative frontier-based approach, where frontier utilities are defined to exploit the initial information and prioritize the search of corrosion.
- Finally, we test and analyze the proposed algorithms by using a 2.5D simulator built upon Mesa [2], and compare the proposed solutions with alternative approaches by considering several performance metrics. This analysis provides insights on the role of prior information and on some system parameters, and shows that both proposed approaches are superior to the alternatives.

The rest of the paper is organized as follows. Section II discusses the main related work. Section III proposes a formal definition of the problem. Section IV presents the proposed multi-UAV inspection approaches. Section V compares the algorithms and presents quantitative and qualitative results. Section VI concludes the paper.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871260.

[a]Lakeside Labs GmbH, Lakeside Park B04b, 9020 Klagenfurt, Austria. Email: surname@lakeside-labs.com

[b]Univ Lyon, Inria, INSA Lyon, CITI, 69621 Villeurbanne, France. Email: firstname.lastname@inria.fr

[1]Source: European Maritime Safety Agency

II. Related Work

The inspection problem tackled in this paper lies between the coverage path planning and the optimal search problems. In the coverage problem, the goal is to minimize the time to visit every point in the environment at least once, with at least one robot. Its relevance to many robotic applications produced a vast related literature in the years. A first survey focused on solutions based on a cellular division of the environment, and including the multi-robot case, was presented in [3]. More recent and rich surveys that consider the coverage of fully 3D environments are [4] and [5]. Most of the existing solutions build on the idea of Spanning-Tree Coverage, which ensures to cover the environment completely without any backtracking. The idea was initially presented in [6] for a single robot and extended to multi-robot systems in [7]. In [8], Kapoutsis et al. presented an approach to transform the n-robot problem into n single-robot problems by optimally partitioning the environment in equal areas and assigning each of them to one robot. As described in Section IV, we partially adopt this algorithm for one of our solutions to partition the inspection area. Also based on this approach, a more recent solution for the coverage of complex areas with a team of UAVs and taking into account also the presence of no-fly-zones was presented in [9]. In [10], Sadat et al. present a solution to a non-uniform aerial coverage problem with a single UAV. Contrary to our work, the paper considers no prior information and several sensing planes corresponding to very different resolutions, and assumes that the sensor footprint covers the entire surface at the root node. The set of all sensing planes generates a coverage tree and the authors propose a strategy to navigate it assuming no false positives in identifying the presence of a region of interest from each node. A similar multi-resolution coverage problem with a single UAV is presented in [11]. Also in this case, no prior information is taken into account and the UAV adapts at each step its altitude to obtain a suitable resolution to perform deep-learning based semantic segmentation of the terrain.

Although a coverage path can ensure to visit in a minimum time every point in the environment, coverage solutions can be sub-optimal if the mission priority is to find and inspect first some regions of interest. In this case, the problem becomes strictly connected to a search problem [12]. As we show in our work, navigating through the environment searching for regions of interest can be related to a standard autonomous exploration problem. A similar connection can be found in [13], where the authors propose an exploration-based strategy that includes semantic information to identify areas of particular relevance in a search and rescue scenario. A combination of grid search and evolutionary optimization is used in [14] to generate the trajectory of an aerial vehicle in a multi-resolution informative path planning problem. This strategy exploits Gaussian Process to model the environment and adaptively focus on areas of high interest. In our work, we exploit the concept of frontier to guide UAVs through the mission. Frontier-based algorithms are popular approaches to conduct autonomous exploration in unknown environments

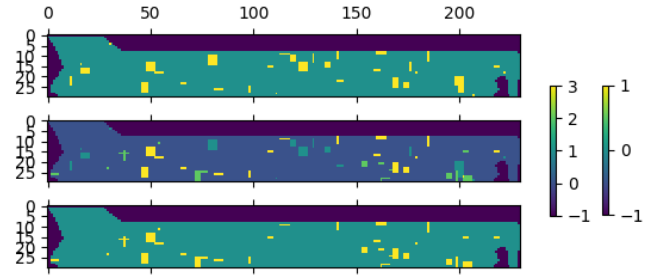

Fig. 2: Corrosion matrix or ground truth C (top), contingency matrix (middle), and estimated corrosion matrix C' (bottom). Instance generated with parameters: probability of a cell to be the center of a corroded cluster $p_{\text{C}} = 0.5\%$; maximum side of corroded clusters $l_{\text{C}} = 5$; probability that a cluster is known in the estimated corrosion matrix (true positive) $p_{\text{TP}} = 60\%$; probability that a cell is the center of a false negative cluster $p_{\text{FP}} = 0.3\%$.

[15], [16], [17]. Although this class of solutions is well adapted to complex environments with obstacles, this is not the case in our problem where UAVs know the area to inspect and have no obstacle to avoid. However, we show that by modifying the definition of frontiers utility, frontier-based algorithms are also suitable solutions to take into account prior information in our cooperative inspection problem.

III. Problem Definition

We consider a fleet of n UAVs to inspect a surface discretized as a rectangular grid of length and height equal to a and b cells, respectively. The position of the UAV k is

$$(x_k, y_k, z_k) \in \{0, \dots, a-1\} \times \{0, \dots, b-1\} \times \{1, 2\}\,.$$

A UAV with $z = 1$ in the discrete environment is at the distance ζ_1 from the hull in the real world. This distance is sufficient to acquire high-resolution images for close-up inspection. A UAV with $z = 2$ is at the distance ζ_2 sufficient to detect corrosion. UAVs acquire square images with side s (sensing side) that is proportional to the distance from the hull ζ with a factor depending on the camera's aperture angle θ: $s = 2\zeta \tan(\theta/2)$. Flying at distance ζ_2 is preferable to inspect non-corroded areas since UAVs can cover a larger area. UAVs can move from their position to one of the four neighbor cells (up, down, left, right) in a constant time step u_{xy}, and change distance from the hull in a constant time step u_{z}, which depends on $\zeta_2 - \zeta_1$.

Cells of the grid with a level of corrosion greater than a given threshold require close-up inspection. We refer to these cells as corroded. The grid is mathematically represented by the corrosion matrix C of size $b \times a$ and entries

$$C_{i,j} = \begin{cases} -1 & \text{cell not part of the hull} \\ 0 & \text{cell not corroded} \\ 1 & \text{cell corroded} \end{cases}\,. \tag{1}$$

The position of the -1 entries are exactly known in advance. Regarding the corrosion, we consider the presence of prior information expressed as a prediction map that indicates the expected location of corrosion clusters. The availability of this predicted map is justified by the fact that,

even though the exact corrosion location is not known in advance, high-probability corroded areas can emerge from past data in the history of the boat or be identified by observations from a human operator from the ground. We so introduce the expected corrosion matrix C', with the same structure as C, but where the 0s and 1s entries correspond to high-probability non-corroded and high-probability corroded cell respectively. To generate this information, we consider probabilities to represent both false positives p_{FP} (i.e., expected corrosion in areas that are not corroded) and false negatives $1-p_{\text{TP}}$ (i.e., expected absence of corrosion in areas that are corroded), as shown in Fig. 2. In simulations, C and C' are generated as follows: every cell that is part of the hull has probability p_{C} to be the center of a corrosion cluster in C (ground truth). After this first step, every center in C is given to C' with probability p_{TP}, whereas any other cell part of the hull is given to C' as a center with probability p_{FP}. Then, a rectangular cluster with length and height uniformly chosen between 1 and $l_{\text{C}} \in \{1, 3, \ldots\}$ is generated for every center (ground truth and false positives). Clusters can overlap. The system keeps then track of the mission status with the map matrix M of size $b \times a$ and entries

$$M_{i,j} = \begin{cases} -1 & \text{cell not part of the hull} \\ 0 & \text{cell's status unknown (not covered yet)} \\ 1 & \text{non-corroded cell inspected } (z = 1, 2) \\ 2 & \text{corroded cell expected/detected } (z = 2) \\ 3 & \text{corroded cell inspected } (z = 1) \end{cases} . \tag{2}$$

The matrix is initialized with -1, 0, and 2 values, and contains -1, 1, and 3 values at the end of the inspection mission. The values 2 in the initialization come from C', the others are then discovered during the mission.

In Part-TSP, the grid is firstly partitioned in n regions and each region is assigned to a UAV. The partition information is stored in the assignment matrix A of size $b \times a$ and entries

$$A_{i,j} = \begin{cases} -1 & \text{cell not part of the hull} \\ k & \text{cell assigned to the UAV } k \end{cases} . \tag{3}$$

For this algorithm, UAVs do not communicate all the time. Every UAV k keeps a local version M^k of the map matrix where only the cells assigned to the UAV itself are different than -1. For all algorithms, we assume that UAVs have access to their global position.

The mission goal is to provide to the operators on the ground the information retrieved by the close-up inspection of all corrosion patches as soon as possible while keeping a reasonable time for the coverage of the entire surface. To do so, the UAVs team needs to find a compromise between exploiting the prior available information to give priority to expected corroded regions and minimizing the complete mission time to ensure that all corroded spots are inspected. For this reason, the system is evaluated according to four performance indicators: the percentage of corroded area inspected in time $c(t)$; the mission progress (percentage of area inspected) in time $m(t)$; the time T_{c} necessary to perform the close-up inspection of all corroded cells (close-up inspection time); and the time T_{m} necessary to inspect all cells (mission time).

$$T_{\text{c}} = \{t : c(t) = 100\} \tag{4}$$

$$T_{\text{m}} = \{t : m(t) = 100\} \tag{5}$$

It is worth noticing that T_{c} can be exclusively used to provide an a posteriori evaluation of the algorithms and not during the mission since the UAVs cannot know when all corroded spots have been inspected.

IV. Multi-UAV Inspection Approaches

We propose the Part-TSP and Coop-Frontier approach to exploit the prior information on corrosion patterns and solve the bi-resolution inspection problem. These approaches differ in the way UAVs are coordinated, and the path for each UAV is computed. Part-TSP assigns each UAV to an area of interest at the beginning of the mission, eliminating overlap between different UAV. Every UAV plans its path by solving a TSP problem online. In Coop-Frontier, UAVs are regularly assigned to the frontiers that maximize a given utility, designed to minimize overlap and exploit prior information. The path is determined by the frontier chosen.

A. Partitioned-TSP Based Inspection

As introduced above, Part-TSP solve the multi-UAV path planning problem by dividing the area into sub-areas and assigning each sub-area to a UAV. Thus, turning a multi-UAV problem into many single-UAV problems. For this purpose, we use the DARP algorithm [8], which partitions the space in connected sub-areas of similar size. It is worth noticing that DARP was developed for a squared surface with obstacles. We do not consider obstacles, but some cells are not part of the hull. These cells are given in input to DARP as obstacles but are not considered obstacles for the navigation, i.e., UAVs can fly over them. Using DARP together with a good path planning algorithm allows achieving a coverage time close to the minimum for uniform coverage problems. However, this is not the case in our problem since UAVs can change their distance from the hull.

The problem of covering the sub-areas at different resolutions is further divided in two problems that are solved online: decide when to change inspection distance and compute the best path at a given inspection distance. At every time step, each UAV decides whether to change distance considering its current state and sensing information according to the finite-state machine (FSM) in Fig. 3. Every time the distance changes or a plan is completed (e.g., all corroded cells detected are inspected) the UAV re-plans the path. The path planning at a given distance ζ_z is further divided in two steps: divide the area assigned to the UAV into segments, and minimize the travel time between segments solving a traveling salesman problem (TSP). The division in segments allows to keep the planning complexity tractable since large unexplored areas are divided into a small number of long segments, which result in small TSPs. Given the solution of the TSP, it is trivial to calculate the path of the UAV.

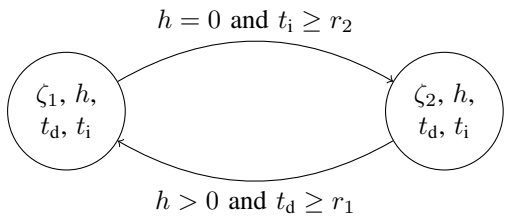

Fig. 3: Finite-state machine defining when UAVs change distance in Part-TSP. At every time step, the UAV decides to change distance considering the current distance ζ_z, the number of corroded cells to inspect h, as well as the time from the last corroded cell detected t_d and inspected t_i with respect to the thresholds r_1 and r_2.

1) Distance Change with FSM: The algorithm for changing inspection distance has a considerable impact on T_c and T_m. As depicted in Fig. 3, a UAV at distance ζ_2 changes to ζ_1 if corroded cells are detected ($h > 0$) and the time from last corroded cell detected t_d is greater than the threshold r_1. The UAV at distance ζ_1 changes to ζ_2 if all corroded cells detected were inspected ($h = 0$) and the time from last corroded cell inspected t_i is greater than the threshold r_2. If corrosion is expected in the sub-area assigned to the UAV (matrix C'), the UAV starts with $h > 0$, changes to distance ζ_1 after t_d time steps (if starting from ζ_2), and visits all expected corroded cells according to the shortest path. This behavior allows achieving a short T_c.

The conditions $t_\mathrm{d} \geq r_1$ and $t_\mathrm{i} \geq r_2$ help to reduce the number of distance changes and, in turn, to minimize T_c and T_m. If these conditions are not used ($r_2 = r_1 = 0$), the UAV goes immediately to ζ_1 when one or more corroded cells are detected and transitions back to ζ_2 just after inspecting all of them. This strategy might work well for small and sparse corrosion patches, but not for large clusters. When detecting part of a large cluster, the UAV immediately reduces the distance to ζ_1, inspects the cells detected, and goes back to ζ_2. In going back, the sensing range increases and the UAV discovers new corroded cells part of the cluster. Therefore, it reduces the distance again, i.e., it oscillates between the two distances until the entire cluster is inspected. An intuitive way to avoid or mitigate this behavior is to delay the transition to ζ_2 as long as the drone is inspecting new corroded cells, i.e., $t_\mathrm{i} \geq r_2$. It might also be advantageous delaying the transition to ζ_1 to detect more corroded cells during the coverage phase, i.e., $t_\mathrm{d} \geq r_1$.

The values of r_1 and r_2 that minimize T_c and T_m depend on the corrosion distribution, sensing ranges at different distances, and time necessary to switch between distances. Simulation results (not shown for space reasons) based on a scenario with medium corrosion ($p_\mathrm{C} = 0.005$ and $l_\mathrm{C} = 5$ as in Fig. 2) lead to the following insights: 1) r_1 has the most significant impact on performance. 2) For a small difference in sensing range ($s_1 = 5$, $s_2 = 7$) it is advantageous to have a small delay on both plane transitions ($r_1 = 1$, $r_2 = 1$). 3) As s_2 increases ($s_2 = 9, 11$) it is better to transition back to ζ_2 as soon as possible ($r_2 = 0$) and delay the transition to ζ_1 ($r_1 = 2, 4$). This is intuitively explained by the fact that as s_2 increases, the time step to change distance u_z increases linearly, but the number of cells sensed increases quadratically. For the numerical examples presented in the paper, we assume $r_1 = 2$ and $r_2 = 0$.

2) Division in Segments: The region assigned to the UAV is divided in horizontal segments with height equal to the sensing side s in the current inspection plane. In terms of matrix representation, segments are built selecting s rows at a time in the map matrix M, and sweeping along columns in search of cells with a targeted value. The targeted value is 0 (unknown) if the UAV is planning for coverage, or 2 (expected/detected corrosion) if the UAV is planning for inspecting corrosion patches. The process is repeated for the entire matrix. The outcome of this process is a graph with $2g$ nodes, where g is the number of segments that cover the region. Using the division in segments allows to have small graphs for large uniform areas and, in turn, to solve small TSPs.

3) Path Computation with TSP: The TSP is solved on a complete undirected graph with nodes in $\mathcal{V} = \{0, 1, \ldots, 2g+1\}$, where node 0 is the current position of the UAV, nodes $1, 3, \ldots, 2g-1$ are left ends of segments, nodes $2, 4, \ldots, 2g$ are right ends of segments, and $2g+1$ is an artificial node necessary to minimize the open path instead of the closed one. Since the graph is complete, it contains $|\mathcal{E}| = |\mathcal{V}|\,(|\mathcal{V}| - 1)\,/2$ edges, with $|\mathcal{V}| = 2\,(g+1)$. The cost of going from node i to j is the Manhattan distance $d_{i,j}$ between the positions of the two nodes on the grid (x_i, y_i) and (x_j, y_j).

We use a Dantzig–Fulkerson–Johnson formulation for the TSP [18] (6-12).

$$\min \sum_{(i,j)\in\mathcal{V}^2 : i>j} d_{i,j} e_{i,j} \tag{6}$$

$$\sum_{i=v, j<i} e_{i,j} + \sum_{j=v, i>j} e_{i,j} = 2 \quad \forall v \in \mathcal{V} \tag{7}$$

$$\sum_{(i,j)\in\mathcal{S}^2 : i>j} e_{i,j} \leq |\mathcal{S}| - 1 \quad \mathcal{S} \subsetneq \mathcal{V},\ |\mathcal{S}| \geq 2 \tag{8}$$

$$e_{2l+2,2l+1} = 1 \quad \forall l = 0, 1, \ldots, g-1 \tag{9}$$

$$e_{2g+1,0} = 1 \tag{10}$$

$$d_{2g+1,j} = 0 \quad \forall j \in \mathcal{V},\ j \neq 0,\ j \neq 2g+1 \tag{11}$$

$$e_{i,j} \in \{0, 1\} \quad \forall i, j \in \mathcal{V},\ i \neq j\,. \tag{12}$$

In the integer program, the variable $e_{i,j}$ is 1 if the edge is part of the solution path in one of the two directions, and 0 otherwise. Since the graph is undirected, $e_{i,j} \equiv e_{j,i}$. However, we only use $e_{i,j}$ with $i > j$. Eq. (6) states that the objective is to minimize the path length. The sum is done on all pair of nodes such that the second node is smaller than the first, i.e., on all edges of the graph in one direction. Eqs. (7) state that every node must be visited once, i.e., must have two edges, one incoming and one outgoing. Eqs. (8) state that there cannot be sub-tours. Having a constraint for every possible sub-tour would be unfeasible. Therefore, we first solve the problem without sub-tour constraints. As sub-tours appear, we add the related constraints and compute a

new solution until finding one without sub-tours. Eqs. (9) guarantee that the paths along the segments are included in the solution, i.e., the surface of interest is covered. Since we are interested in minimizing the open path between the position of the UAV and one of the ends of a segment, we introduce the artificial node $2g+1$ that is connected to node 0 (UAV position) (10), and has equal distance from all other nodes. These distances are set to zero to not impact the path length (11). We find the minimum closed path on this problem and remove node $2g+1$ from the solution. Eqs. (12) state the values that the decision variables can assume.

B. Frontier-Based Inspection

We here propose a second solution that, instead of relying on an initial partition of the surface, provides a fully cooperative strategy. Although this might result in an increase of overlapping information if compared with the partition-based solution, this approach presents the advantage of being robust with respect to the possible failure of one or more UAVs.

To solve the considered inspection problem, integrating the prior information on the corrosion distribution and maintaining the guarantees of complete coverage of the entire environment, we propose a solution based on the concept of frontier [19]. A frontier, which is usually introduced to efficiently explore unknown environments, is defined as the division between the already detected and the still unknown environment. Once the set of frontiers at a given time has been identified, each robot selects one of them as its new goal. In the case of single-robot exploration, the most intuitive solution is to travel towards the closest frontier. When multiple robots are involved, a more complex strategy needs to be adopted to avoid overlap between them. In our scenario, the information on the expected presence of corrosion (both from prior information and online detection) will also be taken into account.

To do so, we introduced the following utility function:

$$U_f = \alpha \frac{1}{d_{\text{HP}}} + (1-\alpha)\frac{1}{d_{\text{C}}}\,, \tag{13}$$

where d_{HP} is the distances between the frontier and its closest undetected point with high probability of corrosion, d_{C} is the distance between the frontier and its closest detected corroded point (equal to the grid resolution if the frontier itself is corroded), and α is a weight constant. The second term is justified by the assumption that corrosion is often present in clusters and the detection of a corroded spot can be interpreted as a high probability of finding more corrosion in proximity.

To select the best frontier to assign to a given UAV, we also take into account the traveling cost by introducing an additional term depending on the distance d_f between the UAV and the frontier. In this way, in the case of a single UAV performing the inspection task, the frontier assigned at each iteration would be the one solving:

$$f^* = \arg\max_f \left(U_f + \beta \frac{1}{d_f} \right)\,. \tag{14}$$

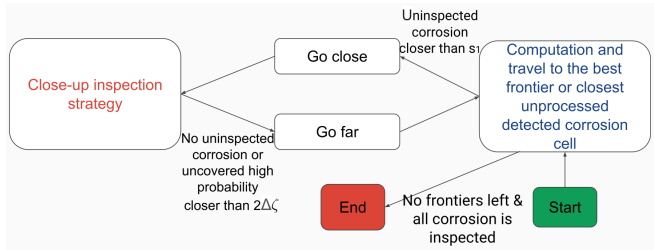

Fig. 4: The main steps of the frontier-based approach.

In case of multiple cooperating UAVs, we propose a solution inspired by the exploration approach presented in [15] to exploit the same utility function defined in eq. (13) and coordinate the UAVs in order to minimize the possible redundancy in the acquired information. The idea behind this algorithm is to proceed with an iterative robot-frontier assignment where, at each step, the utility of frontiers close to the already assigned frontiers is reduced. More formally, at each iteration, a couple (robot, frontier) is selected by solving a problem analogous to (14), i.e.

$$(r^*, f^*) = \arg\max_{(r,f)} \left(U_f + \beta \frac{1}{d_{r,f}} \right) \tag{15}$$

with $d_{r,f}$ equal to the distance between the UAV r and the frontier f. Once an optimal couple has been identified, the utility function U_f is updated. Let us consider that the first m frontiers $f_1, ..., f_m$ has been selected, the utility of the remaining frontiers will become:

$$U(f|f_1, ..., f_m) = \prod_{i=1}^{m} P(||f - f_i||)\, U_f \tag{16}$$

where the reduction function $P(d)$ is defined as:

$$P(d) = \begin{cases} \frac{d}{2s_2} & \text{if } d < 2s_2 \\ 1 & \text{otherwise.} \end{cases} \tag{17}$$

In this way, the utility of all frontiers closer than $2s_2$ from the assigned targets is significantly decreased, forcing the fleet of UAVs to spread over the environment and so reducing the probability of having overlapping fields of view.

When a corrosion point appears in the newly explored environment, it is inspected before continuing the selection of a new frontier. The search is then maintained at close-up distance until no detected corrosion is left to inspect or high probability regions are present within a distance of $2\Delta\zeta = 2(\zeta_2 - \zeta_1)$ (see Fig. 4). It is important to note that relying on frontiers guarantees that the environment can be completely covered and thus, no target will remain undiscovered.

V. Simulation Results and Analysis

A. Simulation Setup

The approaches proposed in this paper are evaluated with a multi-agent simulator (see Fig. 5) based on the MESA framework [2]. UAVs can move in four directions on a rectangular grid (x, y coordinates) and have a state variable to store the distance from the hull (z coordinate). The grid

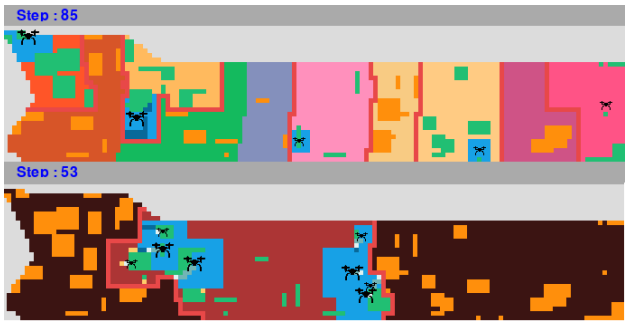

Fig. 5: MESA simulation, Part-TSP (top) and Coop-Frontier (bottom). The DARP areas (top) are in different colors. Green: inspected corroded cells, orange: corroded cells, blue: cells in sight, light red: frontiers, dark red(bottom)/lighter color(top): explored cells, white: target position of a UAV.

has length $a = 230$ and height $b = 30$ cells. Every cell represents a square with a side of 0.5 m in real life. The shape of the hull is a 2D section taken from the 3D model of a real hull. We assume that UAVs move from one cell to the other in a constant time $u_{\text{xy}} = 1$ second, and change z in a constant time u_{z} whose value depends on the difference between the distances for close-up ζ_1 and standard ζ_2 inspection. These distances are related to the sensing side s_1 and s_2 through the camera angle θ. For all simulations, we fix $s_1 = 5$ cells (2.5 m) and $s_2 = 7, 9, 11$ cells (3.5, 4.5, 5.5 m). Assuming a camera angle of $\theta = 90$ degrees, the difference between the inspection distances $\zeta_2 - \zeta_1 = 1, 2, 3$ cells (0.5, 1, 1.5 m), which correspond to a time $u_{\text{z}} = 1, 2, 3$ seconds, respectively. To create the map reflecting an imprecise prior knowledge on the positions of corrosion clusters, we consider both false positive and false negative probabilities as explained in Section III (see Fig. 2).

B. Comparative Performance Evaluation

We evaluate Part-TSP and Coop-Frontier in three corrosion scenarios (Fig. 6 and 7): 1) the first scenario is a reference, i.e., the parameters of the corrosion distribution (p_{C} and l_{C}) are homogeneous on the hull, and the prior information is perfect ($C = C'$); 2) the second scenario investigates how the performance changes with imperfect information, i.e. C' has both false negatives and false positives; 3) the third scenario studies the effect of a inhomogenous corrosion distribution, i.e., the front half of the hull has more and larger clusters.

The proposed algorithms are compared to a partitioned lawnmower approach at fixed distance ζ_1, a solution widely used in literature and our baseline. In this approach, the area of interest is partitioned in sub-areas as for the Part-TSP. Every UAV swipes its sub-area at close distance ζ_1 in horizontal segments from one side of the sub-area to the other, and moves to the next segment (from bottom to top) when the current segment is covered. Since UAVs do not change distance or use prior information, the performance of this policy does not depend on the parameters s_2, p_{TP}, and p_{FP}. Furthermore, we evaluate a version of the TSP approach non using prior information (Part-TSP$_0$) to investigate the impact of prior information on performance. All these

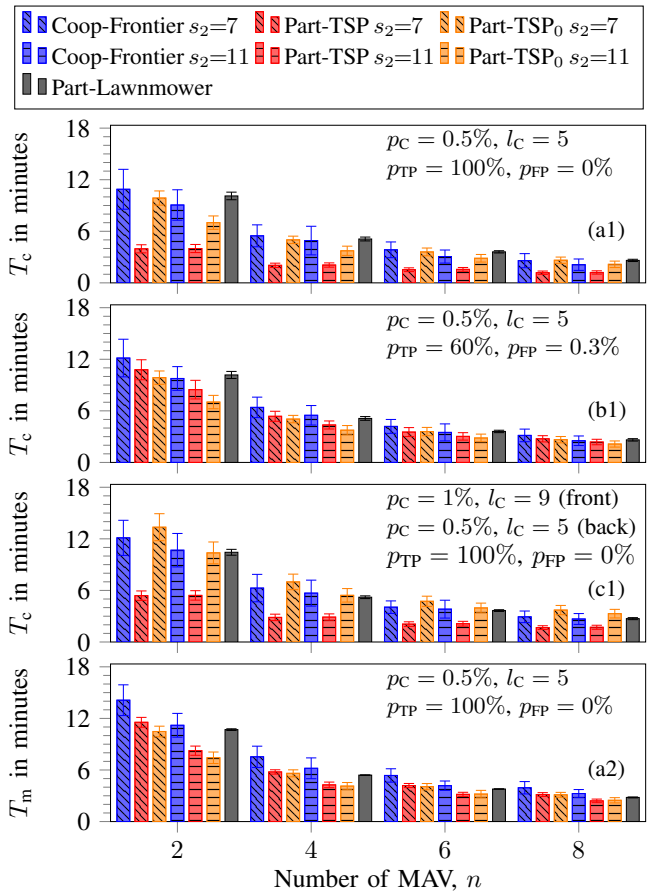

Fig. 6: Corrosion inspection time T_{c} and mission time T_{m} for scenarios with perfect information (a1/a2), imperfect information (b1), and inhomogeneous corrosion distribution (c1). Columns heights are averages and error bars are standard deviations.

comparisons are done with a light/medium corrosion as more commonly encountered in real applications. Heavy corrosion is unrealistic for the application of interest. Quantities are obtained averaging on 50 instances of corroded maps.

1) Perfect information: Given perfect information, Part-TSP is the fastest at inspecting corroded areas, i.e., $c(t)$ is at the left of other curves (Fig. 7 a1). The sensing side s_2 impacts the discovery of corroded cells not known in advance, but not $c(t)$ directly. Therefore, it does not impact Part-TSP if $C = C'$. On the contrary, Part-TSP$_0$ has considerably better $c(t)$ for $s_2 = 11$ than for $s_2 = 7$ because it is able to discover more corroded cells per time unit. The better $c(t)$ of Part-TSP compared to Part-TSP$_0$ comes at the cost of slightly worse T_{m} (Fig. 6 a2) due to the fact that UAVs cover some areas more than once to inspect the corroded cells first. Coop-Frontier is faster than Part-TSP$_0$ to inspect most of the corroded cells (up to 80-95% approximately), but slower to inspect the last corroded cells (Fig. 7 a1), which results in a greater T_{c} (Fig. 6 a1). A possible explanation is that UAVs using Coop-Frontier interfere with each others path and tend to cover the same area more than once toward the end of the mission because space partitioning is not used. Regarding the mission time T_{m}, Lawnmower is better than

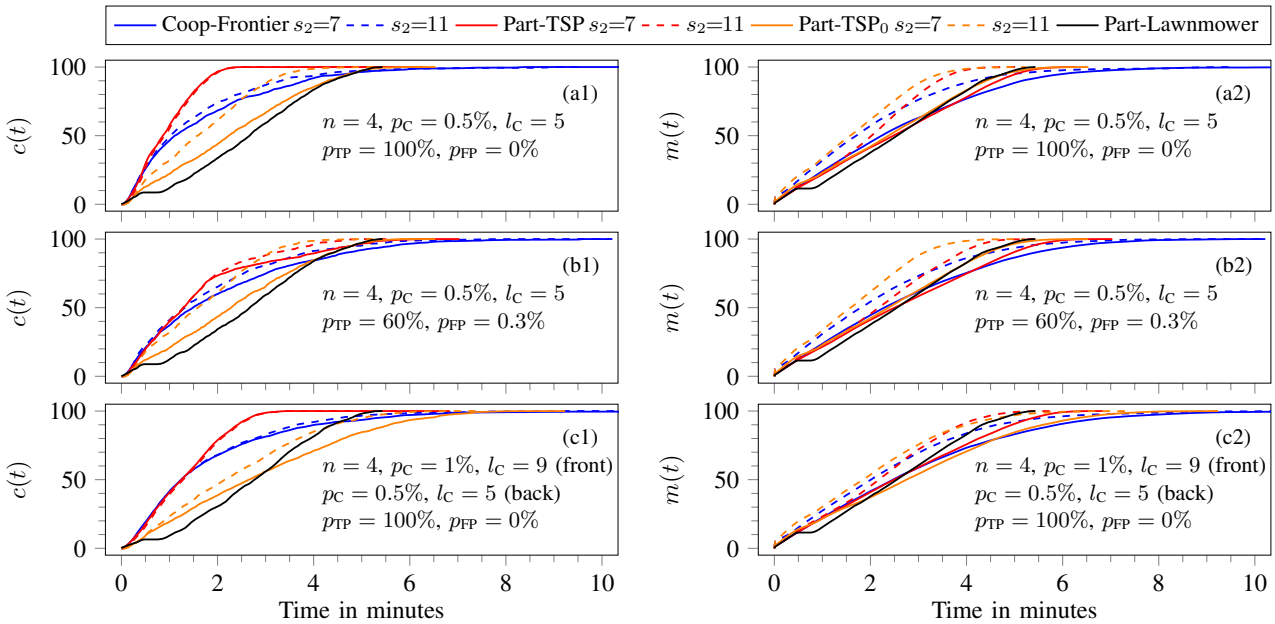

Fig. 7: Percentage of corrosion inspected $c(t)$ and mission progress $m(t)$ in time for scenarios with perfect information (a1/a2), imperfect information (b1/b2), and inhomogeneous corrosion distribution (c1/c2).

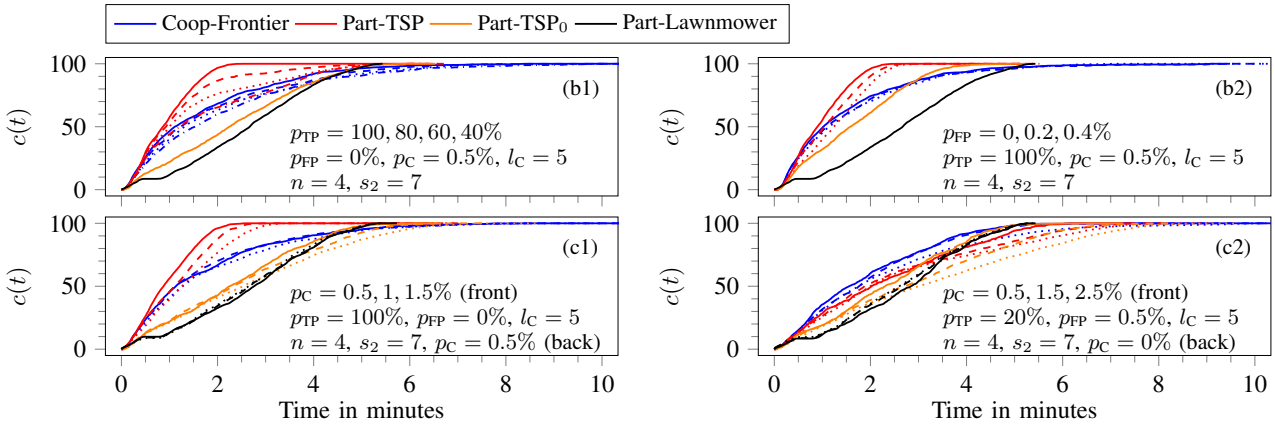

Fig. 8: Percentage of corrosion inspected $c(t)$ in time for different values of p_{TP}, p_{FP}, and p_C front/back (inhomogeneous distribution). Evaluation for the individual parameters (b1/b2/c1) and their combined effect (c2). The values of the varying-parameter from left to right for each sub-figure (e.g., $p_{TP} = 100, 80, 60, 40\%$ for b1) correspond to the line patterns solid, dashed, dotted, and dash-dotted (if applicable), respectively.

Coop-Frontier and similar to Part-TSP for $s_2 = 7$ (Fig. 6 a2). However, is it similar to Coop-Frontier and worse than Part-TSP for $s_2 = 11$. Changing distance with the hull is advantageous to achieve a fast inspection if the sensing range at ζ_2 is sufficiently large because the time necessary to change distance increases linearly while the sensing area increases quadratically. Similar considerations hold for $m(t)$.

Part-TSP achieves considerably smaller T_c than other algorithms and slightly worse T_m than Part-TSP$_0$ (Fig. 6 a1 and a2). This difference in performance between all algorithms is more prominent for smaller fleets: as the fleet size increases, the area per UAV decreases, and UAVs paths become less important. Furthermore, the performance gain from additional UAVs is reduced as the fleet size increases. An explanation for partitioned approaches is that some subregions have significantly more corrosion than others if there are many small subregions. Therefore, some UAVs will finish early and wait for the UAVs in the subregions with more corrosion. Coop-Frontier does not have this problem. However, as the fleet size increases, UAVs interfere more with each others' path and tend to cover the same areas more than once. Similar interpretations apply to the close-up inspection times. However, the difference in the performance of different algorithms is less pronounced.

2) Imperfect information: We consider both false negative and false positives in the prior information. Intuitively, p_{TP} is approximately the percentage of truly corroded clusters known in advance, whereas p_{FP} can be interpreted in comparison with p_C. For light/medium corrosion, in first approximation, the number of clusters given as false positive is p_{FP}/p_C

times the number of truly corroded clusters. Part-TSP performance degrades with imperfect information and becomes dependent on s_2 (compare a1 and b1 in Fig. 6 and 7). Since not all clusters are given as prior information, $c(t)$ bifurcates at the time instant such that all know corrosion cells are inspected. Coop-Frontier is considerably more robust with respect to imperfect information. An intuitive explanation is that, while Part-TSP inspects the prior corrosion first, Coop-Frontier seeks to find a balance between inspecting corrosion known in advance and exploring nearby frontiers. However, Coop-Frontier does not outperform Part-TSP in the scenario discussed. The impact of imperfect information on both Part-TSP and Coop-Frontier is modest for $m(t)$ (Fig. 7 a2 and b2) and very minor for T_{m} (not shown).

3) Inhomogeneous corrosion: Part-TSP$_0$ performance degrades considerably if corrosion is inhomogeneously distributed (compare a1 and c1 in Fig. 7). Since the initial space partitioning does not consider the corrosion distribution, $c(t)$ and $m(t)$ are mostly determined by the UAVs working in the most corroded sub-regions. An important cause of the increased time is the transitioning between distances ζ_1 and ζ_2 in highly corroded regions. Part-TSP exploits prior information to minimize the number of transitions and mitigate the performance loss. Coop-Frontier is also robust to inhomogeneous corrosion because it does not rely on the initial partitioning.

The impact of the single parameters $p_{\mathrm{TP}}, p_{\mathrm{FP}}, p_{\mathrm{C}}$ front/back (inhomogeneity) on $c(t)$ for all algorithms considered are shown in Figure 8 b1, b2, c1, respectively. In all these scenarios, Coop-Frontier appears more robust than Part-TSP, but it is still outperformed by Part-TSP. However, if all the elements come together in one scenario (Fig. 8 c2), Coop-Frontier outperform all other algorithms.

VI. Conclusion

This paper investigates how to exploit prior information to plan the path for a fleet of autonomous UAVs inspecting large surfaces at two resolutions. UAVs cover the surface of interest at low-resolution to detect corrosion patches or other defects. If corrosion is detected, they get close to the surface to acquire high-resolution images. We propose two approaches for this bi-resolution inspection problem (Part-TSP and Coop-Frontier) that compute paths to inspect all corrosion patches as soon as possible while keeping a reasonable mission time.

Extensive simulations involving inhomogeneous corrosion patterns and imperfect prior information show that: 1) Both Part-TSP and Coop-Frontier inspect corrosion faster than the widely used lawnmower approach. 2) Part-TSP performs better than Coop-Frontier in most cases and, in some cases, completes the inspection of corroded areas in less than half the time of other approaches. This excellent performance comes at the price of a minor increase in mission time compared to a similar TSP approach that seeks to minimize the mission time. 3) Coop-Frontier does not use space partitioning and is significantly more robust than Part-TSP to unreliable information. For these reasons, Coop-Frontier is the best performing approach in extreme scenarios such as heavy inhomogeneity combined with little unreliable information.

In the future, we aim to further improve our solutions, firstly by including the prior information also in partition step, to obtain non-uniform regions that take into account the predicted inspection time; secondly by allowing an online recomputation of the partition to avoid that UAVs remain idle when they complete their assigned area while others are still inspecting different parts. It is also our intention to consider more complex surfaces, with forbidden areas and obstacles, and to finally deploy and evaluate our solutions with real UAVs in the context of the BugWright2 project.

References

[1] "The BugWright2 project – autonomous robotic inspection and maintenance on ship hulls and storage tanks," https://www.bugwright2.eu.

[2] "Mesa: Agent-based modeling in python 3+," https://github.com/projectmesa/mesa/, accessed: 2021-05-15.

[3] H. Choset, "Coverage for robotics – a survey of recent results," *Annals of mathematics and artificial intelligence*, vol. 31, pp. 113–126, 2001.

[4] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous systems*, vol. 61, no. 12, pp. 1258–1276, 2013.

[5] T. M. Cabreira, L. B. Brisolara, and P. R. Ferreira Jr, "Survey on coverage path planning with unmanned aerial vehicles," *Drones*, vol. 3, no. 1, p. 4, 2019.

[6] Y. Gabriely and E. Rimon, "Spanning-tree based coverage of continuous areas by a mobile robot," *Annals of mathematics and artificial intelligence*, vol. 31, no. 1, pp. 77–98, 2001.

[7] N. Hazon and G. A. Kaminka, "Redundancy, efficiency and robustness in multi-robot coverage," in *IEEE International Conference on Robotics and Automation*, 2005, pp. 735–741.

[8] A. Kapoutsis, S. Chatzichristofis, and E. Kosmatopoulos, "DARP: Divide Areas Algorithm for Optimal Multi-Robot Coverage Path Planning," *J Intell Robot Syst*, vol. 86, no. 3-4, pp. 663–680, 2017.

[9] S. D. Apostolidis, P. C. Kapoutsis, *et al.*, "Cooperative multi-uav coverage mission planning platform for remote sensing applications," *Autonomous Robots*, pp. 1–28, 2022.

[10] S. A. Sadat, J. Wawerla, and R. Vaughan, "Fractal trajectories for online non-uniform aerial coverage," in *IEEE international conference on robotics and automation (ICRA)*, 2015, pp. 2971–2976.

[11] F. Stache, J. Westheider, *et al.*, "Adaptive path planning for uav-based multi-resolution semantic segmentation," in *IEEE European Conference on Mobile Robots (ECMR)*, 2021.

[12] T. Chung, G. Hollinger, and V. Isler, "Search and pursuit-evasion in mobile robotics," *Autonomous Robots*, vol. 31, no. 4, pp. 299–316, 2011.

[13] A. Quattrini Li, R. Cipolleschi, *et al.*, "A semantically-informed multirobot system for exploration of relevant areas in search and rescue settings," *Autonomous Robots*, vol. 40, no. 4, pp. 581–597, 2016.

[14] M. Popović, T. Vidal-Calleja, *et al.*, "Multiresolution mapping and informative path planning for uav-based terrain monitoring," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 1382–1388.

[15] W. Burgard, M. Moors, *et al.*, "Coordinated multi-robot exploration," *IEEE Transactions on robotics*, vol. 21, no. 3, pp. 376–386, 2005.

[16] N. Mahdoui, V. Frémont, and E. Natalizio, "Cooperative frontier-based exploration strategy for multi-robot system," in *IEEE Annual System of Systems Engineering Conference*, 2018, pp. 203–210.

[17] A. Renzaglia, J. Dibangoye, *et al.*, "Combining stochastic optimization and frontiers for aerial multi-robot exploration of 3D terrains," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 4121–4126.

[18] G. Dantzig, R. Fulkerson, and S. Johnson, "Solution of a Large-Scale Traveling-Salesman Problem," *Journal of the Operations Research Society of America*, vol. 2, no. 4, pp. 393–410, 1954.

[19] B. Yamauchi, "Frontier-based exploration using multiple robots," in *Proceedings of the second international conference on Autonomous agents*. ACM, 1998, pp. 47–53.