

Efficient Frontier Management for Collaborative Active SLAM*

Matteo Maragliano¹, Muhammad Farhan Ahmed², Carmine Tommaso Recchiuto¹
Antonio Sgorbissa¹, Vincent Frémont²

Abstract—In autonomous robotics, a critical challenge lies in developing robust solutions for Active Collaborative SLAM, wherein multiple robots collaboratively explore and map an unknown environment while intelligently coordinating their movements and sensor data acquisitions. In this article, we present an approach for coordinating a system consisting of multiple robots to perform Active Collaborative SLAM (AC-SLAM) for environmental exploration. Our method efficiently spreads the robots for maximum exploration while keeping SLAM uncertainty low. Additionally, We also present two coordination approaches, synchronous and asynchronous to prioritize robot goal assignments by the central server. The proposed method is implemented in ROS and evaluated through simulation and experiments on publicly available datasets and similar methods, rendering promising results.

I. INTRODUCTION

Autonomous robotics has emerged as a transformative force in the exploration of complex and uncharted environments. From planetary exploration missions to disaster relief operations, the deployment of autonomous robots has demonstrated a revolutionary potential across a diverse range of applications. At the heart of this success lies the robots' ability to autonomously explore an environment while gathering data and constructing detailed maps of the surrounding environment in real-time—a process known as Active Simultaneous Localization and Mapping (A-SLAM).

Many research works have recently focused on Active Collaborative SLAM (AC-SLAM), which capitalizes on the power of multiple robots working in collaboration. The potential advantages are manifold, from accelerated mapping of terrains to resilient operation in challenging and dynamic scenarios. However, the utilization of multiple robots in collaborative SLAM is not without its challenges. Coordination, resource allocation, and sensor fusion become critical facets that demand careful consideration. Furthermore, the seamless integration of individual robot efforts into a coherent, unified map poses a non-trivial computational and algorithmic challenge.

We propose an implementation of an AC-SLAM algorithm and extend the work in [1] to a multi-agent system, where multiple robots collaboratively map an environment. To achieve this aim, we propose an efficient method to

distribute robots in the environment hence favoring exploration and considering agent priorities using reward- and distance-based metrics to optimize goal selection. We also propose two communication strategies namely synchronous and asynchronous respectively in a centralized approach with a central server, to establish effective communication and coordination of goals among the agents. In the synchronous case, agents await the completion of tasks by all agents before receiving new exploration targets (goals), while in asynchronous implementation robots immediately request new goals from the server once their target goal has been reached.

The subsequent sections are organized as follows: Section II provides a review of related work, Section III explains the methodology of the proposed approach, Section IV shows the simulation and experimental results, finally Section V presents conclusions and prospects for future work.

II. RELATED WORK

A. Active SLAM

In A-SLAM implementations, the robot can actively choose its actions, such as selecting views or locations to investigate, to reduce the uncertainty of its location and map representation. By intelligently planning and executing robot operations to minimize uncertainty, the objective is to increase the efficiency and accuracy of SLAM. What distinguishes Active SLAM from conventional SLAM methods is the integration of optimal decision-making with SLAM, so as to reduce the uncertainty, as reported by [2], [3].

Modern SLAM method is usually modeled with a graph-based approach. Graph-based approaches, like pose graph optimization [4], [5], [6] and Bundle Adjustment (B.A) [7] are used because to their capacity to manage big data and the interaction between robots and the environment. Distributed pose graph optimization and incremental bundle adjustment are adaptations designed for scalability and real-time performance in collaborative active SLAM. We direct interested readers to [8] for the introduction of SLAM methods Furthermore, factor graphs [5], [9] are used to represent a wide variety of problems across robotics, are used for their ability to handle nonlinear relationships and incorporate prior knowledge.

Graph-based approaches define the environment as $\mathcal{G} \triangleq (\mathcal{V}, \mathcal{E})$ ([10]), each vertex $\mathbf{v}_i \in \mathcal{V}$ represents a robot's pose, and each edge $\mathbf{e}_j \triangleq (\mathbf{v}_i, \mathbf{v}_k) \in \mathcal{E}$ denotes a constraint between two poses. The pose-graph edges are weighted by the covariance matrix Σ_j , representing uncertainty or covariance of estimated poses and landmarks [11] [2]. In the context of

*This work was carried out in the framework of the NExT Senior Talent Chair DeepCoSLAM, which was funded by the French Government, through the program Investments for the Future managed by the National Agency for Research ANR-16-IDEX-0007, and with the support of Région Pays de la Loire and Nantes Métropole.

¹ Matteo Maragliano, Carmine Tommaso Recchiuto, Antonio Sgorbissa are with University of Genoa, DIBRIS Department, Genoa Italy

² Muhammad Farhan Ahmed, Vincent Frémont are with École Centrale de Nantes, LS2N, Nantes France

A-SLAM, Theory of Optimal Experimental Design (TOED) [12] is used to provide a scalar mapping of Σ_j debating on its determinant and Eigenvalues (D-Optimally criterion) to guide the reward function to the goal location (frontier). Interested readers are guided to [13],[14], [15], [16] for discussion of uncertainty quantification methods.

B. Frontiers

A key component of A-SLAM is frontier based exploration initially coined by [17], where the frontier is the boundary separating known map locations from unknown ones, as observed by the robot's sensors. Frontiers play a pivotal role in augmenting the precision of robot localization within the context of Active SLAM by enabling intelligent exploration and data acquisition strategies, effectively reducing uncertainty, and enhancing the map-building and localization processes. In [18] an active exploration strategy is proposed where each frontier is weighted based on distance and surrounding unknown cells. In [19] each frontier is segmented, a trajectory is planned for each segment, and the trajectory with the highest map-segment covariance is selected from the global-cost map. The work presented in [20] uses frontier exploration for autonomous exploration a utility function based on Shannon's and Renyi entropy is used for the computation of the utility of paths. Once a frontier has been identified, the robot can use path planning algorithms to reach it and maximize the exploration while minimizing its SLAM uncertainty.

C. Collaborative SLAM

When considering multi-robot systems, two primary aspects come into play. Firstly, teams can be either homogeneous, consisting of robots of the same type, or heterogeneous, [21], with various robot types working together. Secondly, the system's architecture can be centralized, decentralized, or distributed, [13]. Centralized control offers precise coordination but is susceptible to delays and single points of failure. On the other hand, decentralized systems distribute control for enhanced robustness and scalability while requiring effective coordination. Distributed systems empower individual robots for autonomous decision-making, providing fault tolerance and adaptability while demanding efficient communication protocols. Sometimes, systems can combine centralized and distributed elements [21], sharing computational tasks among agents while central nodes handle decision-making.

When multi-robot systems are applied to the exploration and mapping of an unknown environment, significant challenges arise. A crucial aspect is distinguishing between "global" and "local" perspectives, [21]. In traditional single-robot SLAM, the local view is the default, with pose and map estimates based on the robot's internal reference frame. In multi-robot scenarios, maintaining consistency between perspectives becomes crucial, requiring robots to describe landmarks using the same coordinate system. Collaborative SLAM aims to establish this global reference frame, enabling

robots to collectively perceive and learn from each other's observations.

In Centralised Collaborative SLAM, a central server manages information from all agents, centralizing processing and coordination. However, Centralised approaches can become computationally expensive with a large number of robots, while decentralized approaches allow each robot access to a local view and are suitable for large-scale operations, as demonstrated by [4] and [22]. Decentralized Collaborative SLAM offers scalability and efficient distribution of computational tasks among agents. Some approaches involve redundancy in data, [23] and [24], where mapping data are shared among robots, while others are designated for computations. Decentralized systems address certain bottlenecks, being suitable for large-scale operations and providing each robot with its local view, [4], [17], but still face challenges like bookkeeping, information double-counting, and synchronization issues.

Collaborative SLAM consists of two vital components: Front-end processes and Back-end processes [13]. Front-end processes gather data, produce landmark estimates, and manage loop closures, both intra-robot and inter-robot. Intra-robot loop closures mitigate odometry errors, while inter-robot loop closures stitch together poses and local maps of different robots as described in [25], creating a global map of the environment. The Back-end processes focus on estimating robot and map states using data generated by the front end. It is responsible for high computational tasks involving B.A, pose graph optimization using iterative solvers e.g [26] using graph optimization libraries like g2o [5] to compute the estimated robot pose and map.

III. METHODOLOGY

While many research works have been focused on collaborative strategies for SLAM, or single-robot active-SLAM, only a few works have dealt with AC-SLAM. However, these approaches present common limitations: they have high computational costs as the uncertainty is quantified by a scalar mapping of the entire pose graph covariance matrix which may become very large, especially in landmark-based SLAM methods. Secondly, they do not explicitly implement strategies for efficient management of frontiers to speed up map discovery and robot localization. In this work, we propose an AC-SLAM approach that deals with overcoming these limitations.

In the context of single robot A-SLAM the work of [1] uses the Open Karto¹ in ROS noetic² as SLAM Back-end and proposes a modern D-Optimality criterion which is not computationally expensive for uncertainty quantification. This D-optimality criterion is computed as the maximum number of spanning trees in the weighted graph Laplacian. The reward for Each frontier candidate is weighted by this optimality criterion and is passed to the path planner to guide the robot to perform A-SLAM. For a set of frontiers $F =$

¹http://wiki.ros.org/open_karto

²<http://wiki.ros.org/noetic>.

$\{f_0, f_1, \dots, f_N\} \in \mathbb{R}^2, \forall f_{0:N} = (x_{0:N}, y_{0:N})$, each robot computes a matrix of rewards $R = \{r_0, r_1, \dots, r_N\} \in \mathbb{R}$ as shown in Equation 1.

$$R_m = \begin{bmatrix} \text{Reward} & X & Y \\ r_0 & x_0 & y_0 \\ \vdots & \vdots & \vdots \\ r_N & x_N & y_N \end{bmatrix} \quad (1)$$

In this article, we expand [1] to a multi-robot AC-SLAM and propose an efficient frontier sharing and exploration method for AC-SALM. We propose two exploration approaches namely Synchronous and Asynchronous respectively III-C for goal assignment to robots. Additionally, we also present an efficient spread policy III-B for encouraging exploration.

We developed our proposed approach in ROS Noetic using the ROS actionlib³ library. We add a *central server* that receives the list of local frontier points from each robot, computes a global list, and replies with the next target to be reached by the robot. As shown in Figure 1 each robot detects local frontiers in its map using OpenCV and RRT-based frontier detection from [1] and passes them to the *manager node* which acts as a communication gateway between the server and robot. The *merge points* action server creates a unique list of frontier points to be used by all the agents and *choose goals* action server chooses the best goal position for each agent depending on the reward matrix as shown in Equation 1 and spread criterion. Finally, the *assigner* node receives the chosen goal frontier and executes the path planning action using Dijkstra’s algorithm and DWA planners from the ROS navigation stack⁴ as local and global planners respectively. Figure 2 is adopted from our previous work [27] and shows the resultant architecture of the proposed method in ROS with their namespaces. The orange and pink nodes represent the *assigner node* and central server nodes from Figure 1. The grey node is map merging responsible for taking the local maps from each agent and computing a merged map⁵. It also provides the required transformation of detected frontier points from the robot’s local map to the merged map frame. Filtering with percentage and update rewards & goal selection will be explained in Sections III-A and III-B respectively. Throughout this text, we will use frontiers and points interchangeably as they mean the In the following Sections, we elaborate the management policy of the frontiers (Section III-A) and the spreading policy used to speed up the exploration (Section III-B) are discussed. We also present two communication methodologies i.e., synchronous and asynchronous (Section III-C) which deal with the goal assignment to robots.

A. Frontiers Management

Each agent identifies a certain list of frontier points that will be merged on the server side. Depending on the extent of

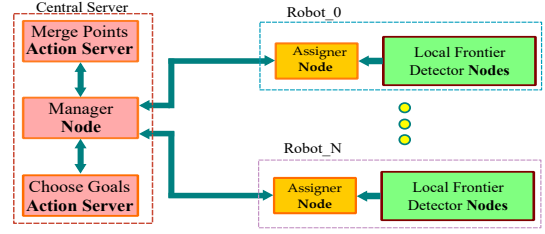


Fig. 1: Central server (red), and local nodes (yellow, green) communication.

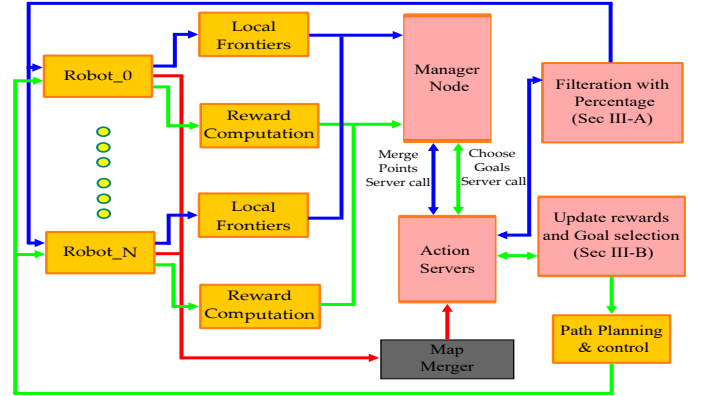


Fig. 2: The architecture of resultant system.

the map, the final global list may consist of several points, which can lead to high computational time on the server side. For this reason, a strategy to reduce the overall number of frontiers was developed. Also, since we are working on multiple robots, some of the points that are considered frontiers in a local map will be located in a region that is fully mapped when considering the global map. To solve both the aforementioned problems, we decided to consider only those points that have a given percentage of unknown cells within a given radius, using a discretized circle and the global merged map. Figure 3 shows an example list of two points P and Q in a partially discovered global Occupancy Grid (O.G) map. For both points, a circle of known radius RAD is drawn and we compute the percentage of the unknown cells over the total inside the circle. Once the percentage is computed, this point is kept or discarded based on the threshold of `PER_UNK` set. In the specific case, opportunistly setting `PER_UNK`, point P will be added to the global list, i.e., considered as a border point, whereas point Q will be discarded.

The usage of a discretized circle can lead to an Inclusion error: the discretized circle may include some cells outside the circular boundary. This error leads to false positives. Exclusion error: the discretized circle may exclude some cells within the actual circular boundary. This error leads to false negatives.

The magnitude of the error depends on the resolution used for the O.G map: higher resolutions provide a more accurate approximation of the circle and consequently negligible errors. Unfortunately, using this approach to reduce the list of frontier points (points) may not be sufficient to meet

³<http://wiki.ros.org/actionlib>

⁴<http://wiki.ros.org/navigation>

⁵http://wiki.ros.org/multirobot_map_merge

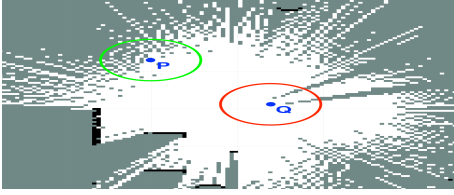


Fig. 3: O.G map representing two points and their radius.

time constraints on the server side. Therefore, we devised an algorithm aimed at further reducing the point number through the adjustment of the radius considered before. The algorithm checks if the global number of points in the list is above a certain threshold: in this case, it recomputes a new list of frontiers by increasing the radius RAD to 0.25m; conversely, if the number of points is below another fixed threshold MIN_PTS or greater than MAX_PTS , the list is reprocessed by decreasing PER_UNK by 10% which is the a-priori fixed given percentage of unknown cells on a given radius. Conversely if the list has more points than a maximum threshold NUM_PTS , the list is reprocessed by increasing the radius RAD and recomputing it. This strategy gives a sufficient number of frontier points in the list for the robots to compute their reward.

B. Spread Policy

To choose targets that allow the agents to explore the map efficiently, a specific spread policy has been implemented. The server keeps track of the already assigned goals both for the asynchronous and the synchronous approach. When a target goal for one agent is selected, the server updates the reward R_{new} for all other agents from the old reward R_{old} , by using a subtractive factor k as shown in Equation 2.

$$R_{new} = R_{old} - k \quad (2) \quad k = \frac{K}{d^2} \quad (3)$$

$$K = \frac{\text{max reward}}{\text{number of targets assigned}} \quad (4)$$

The numerator K , in Equation 3, is set at run-time since it depends on the maximum reward for each agent, and the number of targets already assigned. The denominator d represents the Euclidean distance computed between the last chosen goal and the frontier points in the matrix. In other words, when the server assigns a target to robot j , it will reprocess all the reward matrices for the other agents, updating the reward with a subtractive factor k , which strictly depends on the position of the target assigned to robot j .

Since the k is inversely dependent on d , the more the points are closer to already chosen goals the less likely they are to be chosen as the next goals, thus achieving the task of spreading the goals in the environment. Normalizing K in Equation 4 with the size of the rewards in each matrix, allows for having a subtractive factor that is scaled with respect to the reward matrix of each agent. Thus taking into account the number of already selected points, possibly distributing the reward "budget" among them. By dividing the maximum reward by the total number of selected points, when the

number of targets already explored becomes significant each point will only receive a smaller portion of the total reward, resulting in a more limited effect of k . In the case of asynchronous approach discussed in Section III-C, the priority assigned to robots can lead to having one or more robots with low priority being stuck because they are always prioritized by higher-priority agents. To avoid this issue, the server also takes into account the number of requests not related to each agent. Once this number exceeds a certain predefined threshold $GOAL_SKIP_WAIT$, the corresponding agent will be associated with the higher priority. This approach will avoid having robots stuck, distributing goals more uniformly.

C. Synchronous and Asynchronous Approach

The communication between the agents and the server has been implemented with two policies: synchronous and asynchronous. In the synchronous approach, during the execution of the program, each agent receives the same number of goals. Moreover, each agent waits for all the other robots in the system to reach their goal before starting a new goal procedure. In this case, the central server (Figure 1) has to manage n different agents at the same time and, during the reward computation, the server is given n Reward Matrices, (Equation 1) one for each robot. A priority among agents has been set so that goal assignment is performed respecting this sequence: given two agents i and j with $i < j$, i is assigned a goal before j .

In the asynchronous approach, each agent is assigned in sequence as many goals as it can reach, without waiting for other agents. In this case, the priority is used to choose the winning agent in the case multiple agents perform the request at the same time. Since with this policy, an agent with a low priority can be stuck for a long time, there is also a counter that keeps track of this prioritization among the robots and, when an agent with a low priority is not considered for a long time, automatically assigns it the highest priority so as that the server will satisfy its request as soon as possible. Since the server is used once at a time and the goal chosen is for one robot at a time, the server for each goal of the agent i stores it; this allows the server not to choose an already chosen goal and to update the rewards to spread the agents taking into account all the goals set so far.

IV. RESULTS

A. Simulation Results

The simulations¹ were carried out on ROS Noetic, Gazebo, and Ubuntu 20.04 on Intel Core i7[®], with a system RAM of 32GB and NVIDIA RTX 1000 GPU. As described earlier We modified the approach of [1] to multi-robot and implemented the proposed approach as mentioned in Section III using Open Karto as SLAM backend, RosBot² equipped with Lidar, and planners from the ROS navigation stack. We used maps open-source maps of modified Willow Garage (W.G) from Gazebo simulator and AWS hospital environments

¹YouTube link: <https://youtu.be/MsZqoaEA0gY>

²<https://husarion.com/>.

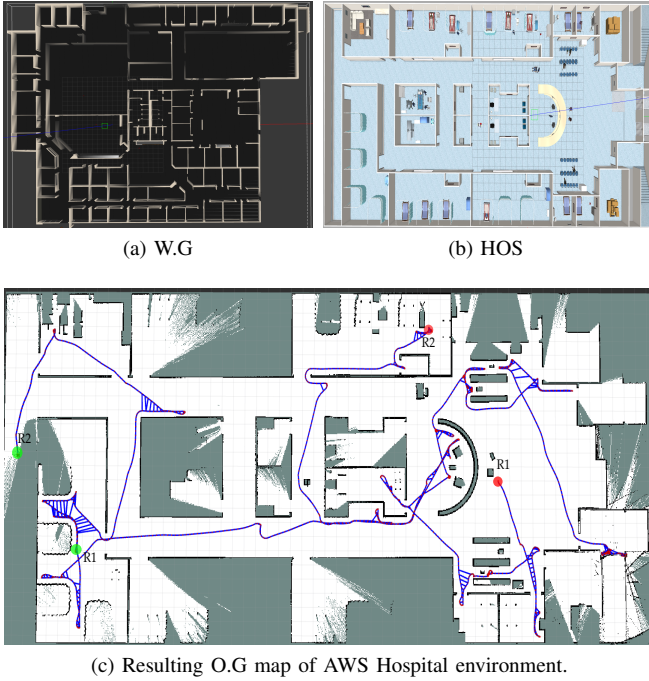


Fig. 4: Environments used and the resulting O.G map showing the initial (red) and final positions (green) of robots.

(HOS)³ measuring $2072m^2$ and $1243m^2$ respectively. Figure 4 shows the Gazebo images and resulting O.G map of HOS indicating the initial, and final poses and resulting pose graphs. The ground truth O.G maps were generated using the gazebo.2Dmap_plugin⁴ which uses wavefront exploration.

We compared our proposed approach against: 1) Frontier Detection based Exploration (Frontier)⁵ of [17] which uses a greedy frontier exploration strategy without any SLAM uncertainty quantification. 2) And [1] by converting it into a multi-robot system namely MAGS. For environment exploration we debate on metrics of percentage of area covered, goal points reduction, percentage of unknown cells (PER_UNK), and radius values (RAD). Regarding map quality, we compared metrics measuring Structural Similarity Index Measurement (SSIM) $\in \{0,1\}$, Root Mean Square Error (RMSE), and Alignment Error (AE) with reference to ground truth maps. We performed 140 simulations, each one lasting 20 minutes rendering a total simulation time of 46 hours. PER_UNK, RAD, MIN_PTS and MAX_PTS were initialized to 60 %, 1m, 0 and 10 respectively.

Figure 5 shows the percentage of maps discovered using Our approach (blue), MAGS (red), and Frontier (green) where R is the number of robots and S & A denote synchronous and asynchronous approaches. It can be observed that Our approach covers an average of 10% and 7.5% more area in W.G and HOS compared to MAGS and Frontier approaches and the agents controlled asynchronously were

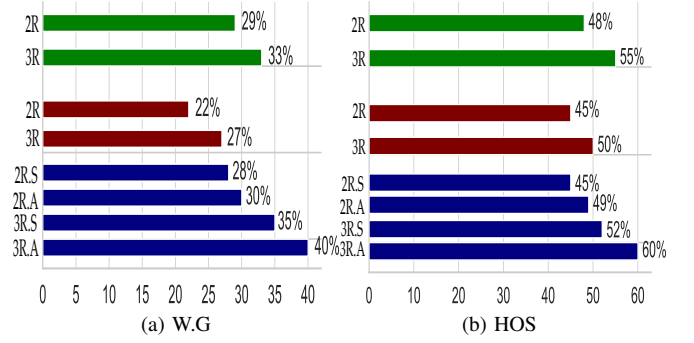


Fig. 5: % of map discovered in W.G and HOS Environments.

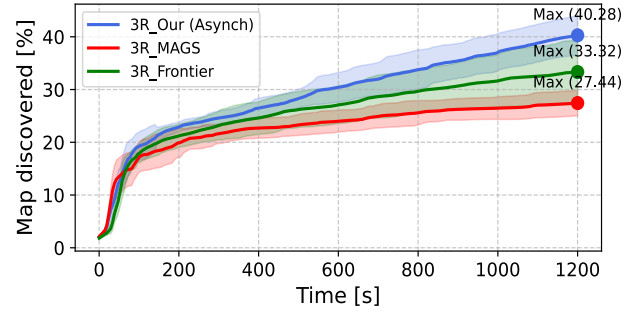


Fig. 6: % of map explored using Our, MAGS and Frontier approaches using 3 robots on W.G environment.

able to discover a higher percentage of the map concerning the ones controlled synchronously. Furthermore, we observe that Frontier outperforms MAGS as it performs frontier exploration without any uncertainty quantification, resulting in more exploration. Figure 6 shows the average rate of exploration for W.G along with the standard deviation using 3 robots. We can conclude that Our approach manages to explore 6.7% and 13% more area than Frontier and MAGS.

In Figure 7 we can deduce that in both W.G and HOS the synchronous and asynchronous approach the number of points is reduced significantly, with a reduction of 80%, 78%, 80%, and 65% for all the cases respectively in W.G. And that of 85%, 84%, 72% and 83% for HOS. Consequently reducing the computational cost required by the reward processing on the server side with the adoption of frontier management strategies in Section III-A to limit the number of global frontiers. Furthermore, the average detected points are lower in synch as the agents have to wait for others, consequently lowering the overall number of points but with the cost of less exploration as evident from Figure 5.

Table I and Table II shows the usage of PER_UNK i.e the percentage of unknown cells to be considered in the radius for computing the information gain of a frontier candidate and RAD showing the radius values changed when the list of points is recomputed using 3 robots with the async approach. We can observe that in W.G, PER_UNK remains at $\leq 40\%$ indicating the re-computation of the list because W.G has fewer obstacles than HOS resulting in less frontier neighbour percentage. Furthermore, we observe that the percentage of

³<https://github.com/aws-robotics>.

⁴<https://github.com/marinaKollmitz/gazebo>.

⁵http://wiki.ros.org/frontier_exploration.

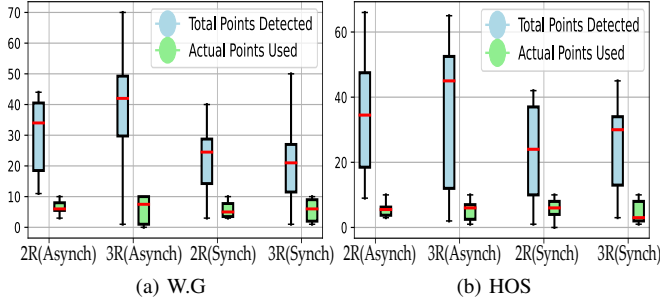


Fig. 7: Number of Points vs the applied approach.

TABLE I: PER_UNK usage

Env.	PER_UNK	Used
W.G	60 %	34.5 %
	50 %	1.4 %
	≤ 40 %	64.0 %
HOS	60 %	67.3 %
	50 %	4.3 %
	≤ 40 %	28.2 %

TABLE II: RAD usage

Env	RAD	Used
W.G	1.00	87.0 %
	1.25	1.8 %
	≥ 1.50	9.7 %
HOS	1.00	76.2 %
	1.25	5.1 %
	≥ 1.50	8.5 %

RAD and PER_UNK remain at 1.00 and $\leq 40\%$ respectively indicating less re-computation of the list on the server, consequently lowering the computational cost.

Regarding the visual analysis of the maps and debating on map quality matrices, the results on average appear promising as shown in (Table III) using 3 robots. In almost all cases, using Our method rendered reduced RMSE, AE, and increased SSIM as compared to MAGS and Frontier methods. We further conclude that the Frontier method when compared to MAGS explores the environment more as shown in Figure 5, but has higher RMSE, AE, and lower SSIM.

B. Experimental Results

Experiments in a real environment were performed using two ROSBot 2R robots⁵ with RPLidar A2 (Figure 8a) with ROS on Ubuntu 20.04.6 (LTS). The robots are equipped with an Intel Xeon® W-2235 CPU 3.80GHz x 12, with 64Gb RAM and Nvidia Quadro RTX 4000 GPU. The environment consists of a room and two corridors measuring $81m^2$ in total as shown in Figure 8b. Figure 9 shows the resultant O.G map along with karto SLAM pose graphs using two robots. From Figure 9a We can observe that using Our approach each agent effectively spreads and explores the environment with a total explored area of $70.31m^2$ compared to that of only $55.80 m^2$ for MAGS from Figure 9b. We performed

⁵<https://husarion.com/manuals/roswbot/>.

TABLE III: MAP QUALITY METRICES.

Env	Method	SSIM	RMSE	AE
W.G	Our (Asynch)	0.74	5.43	25.68
W.G	MAGS	0.86	6.34	28.39
W.G	Frontier	0.20	10.04	40.89
hospital	Our (Asynch)	0.74	4.89	25.39
HOS	MAGS	0.72	6.39	29.98
hospital	Frontier	0.35	12.67	42.89



Fig. 8: Robot and experimental environment used.

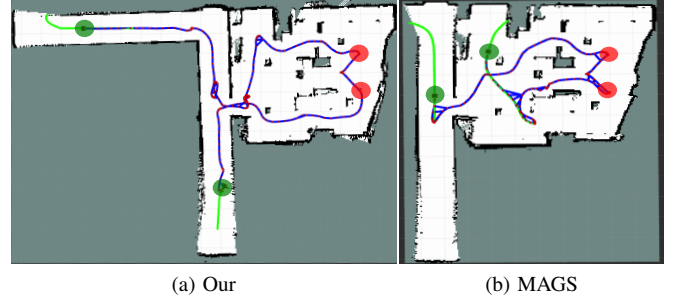


Fig. 9: Final O.G map using Our and MAGS methods indicating initial (red) and final positions of agents (green)

four experiments two using Our and two with MAGS with an experimental time of 20 minutes for each.

Figure 10 shows the average rate of percentage of the global map discovered by the robots with Our (2 experiments) and MAGS (2 experiments) methods respectively. It is evident that using Our approach we manage to cover 26% more map area than MAGS.

The box plot graph shown in Figure 11 shows a reduction in the number of processed points used for exploration using the asynchronous and synchronous methods using Our approach. We can observe that the average number of points reduces both methods from 6 to 5 and from 3 to 2 points respectively. We also observe that the number of points detected in the synchronous approach is less because the robots wait for each other to reach the goal before processing new points. Thus we observe more points in the asynchronous approach. We observe very less points in total as compared to the simulated environment because of the large difference in environment size.

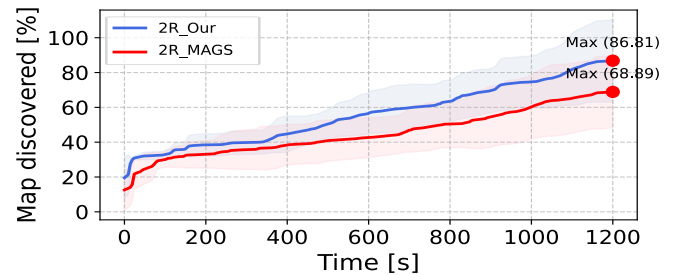


Fig. 10: % of explored map evolution in experiments

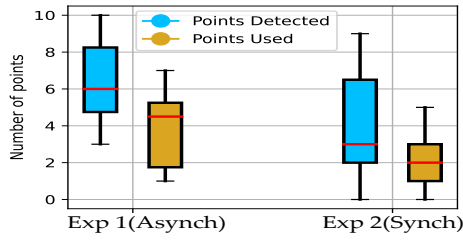


Fig. 11: Points reduction in experiments

V. CONCLUSIONS

We proposed a method for the coordination of multiple robots in a collaborative exploration domain performing AC-SLAM. We proposed a strategy to efficiently manage the global frontiers to reduce the computational cost and to spread the robots in the environment. Two different coordinating approaches were presented for efficient exploration of the environment. We presented extensive simulation analysis on publicly available datasets and compared our approach to similar methods using ROS and performed experiments to validate the efficiency and usefulness of our approach in the real-world scenario. Possible future works can explore strategies to implement the proposed architecture in a decentralized way, thus dividing the computational weight among all the agents and using visual sensors for extracting features as potential frontier candidates.

ACKNOWLEDGMENT

This work was conducted within the framework of the NExT Senior Talent Chair DeepCoSLAM, funded by the French Government through the program "Investments for the Future" administered by the National Agency for Research (ANR-16-IDEX-0007). We also extend our gratitude to the Région Pays de la Loire and Nantes Métropole for their invaluable support in facilitating this research endeavour.

REFERENCES

- [1] J. A. Placed and J. A. Castellanos, "Fast autonomous robotic exploration using the underlying graph structure," in *2021 IEEE/RSJ (IROS)*, Sep. 2021, pp. 6672–6679.
- [2] H. Carrillo, I. Reid, and J. A. Castellanos, "On the comparison of uncertainty criteria for active SLAM," in *2012 IEEE ICRA*, 2012, pp. 2080–2087.
- [3] H. J. S. Feder, J. J. Leonard, and C. M. Smith, "Adaptive Mobile Robot Navigation and Mapping," *IJRR*, vol. 18, no. 7, pp. 650–668, Jul. 1999.
- [4] T. Cieslewski, S. Choudhary, and D. Scaramuzza, "Data-Efficient Decentralized Visual SLAM," in *2018 IEEE ICRA*, May 2018, pp. 2466–2473, iSSN: 2577-087X.
- [5] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G2o: A general framework for graph optimization," Jun. 2011, pp. 3607–3613.
- [6] W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun, "Collaborative multi-robot exploration," in *ICRA*, vol. 1, 2000, pp. 476–481.
- [7] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-Time Single Camera SLAM," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052–1067, Jun. 2007.
- [8] Z. Xuexi, L. Guokun, F. Genping, X. Dongliang, and L. Shiliu, "Slam algorithm analysis of mobile robot based on lidar," in *2019 Chinese Control Conference (CCC)*, 2019, pp. 4739–4745.
- [9] E. Olson, J. Leonard, and S. Teller, "Fast iterative alignment of pose graphs with poor initial estimates," in *ICRA*, 2006, pp. 2262 – 2269.

- [10] J. A. Placed, J. J. G. Rodríguez, J. D. Tardós, and J. A. Castellanos, "Explorb-slam: Active visual slam exploiting the pose-graph topology," in *ROBOT2022: Iberian Robotics Conference*, 2023, pp. 199–210.
- [11] H. Carrillo, P. Dames, V. Kumar, and J. A. Castellanos, "Autonomous robotic exploration using occupancy grid maps and graph SLAM based on shannon and rényi entropy," in *2015 ICRA*, pp. 487–494.
- [12] P. Whittle, "Some general points in the theory of optimal experimental design," *Journal of the Royal Statistical Society*, vol. 35, no. 1, pp. 123–130, 1973.
- [13] M. F. Ahmed, K. Masood, V. Fremont, and I. Fantoni, "Active slam: A review on last decade," *Sensors*, vol. 23, no. 19, 2023.
- [14] J. A. Placed and J. A. Castellanos, "A General Relationship Between Optimality Criteria and Connectivity Indices for Active Graph-SLAM," *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 816–823, Feb. 2023.
- [15] K. Khosoussi, M. Giamou, G. S. Sukhatme, S. Huang, G. Dissanayake, and J. P. How, "Reliable Graphs for SLAM," *IJRR*, vol. 38, no. 2-3, pp. 260–298, 2019.
- [16] K. Khosoussi, S. Huang, and G. Dissanayake, "Novel insights into the impact of graph structure on SLAM," in *2014 IROS*, Sep. 2014, pp. 2707–2714.
- [17] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *IEEE CIRA'97*, 1997, pp. 146–151.
- [18] A. Batinović, J. Oršulić, T. Petrović, and S. Bogdan, "Decentralized strategy for cooperative multi-robot exploration and mapping," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 9682–9687, 2020, 21st IFAC World Congress.
- [19] D. Trivun, E. Šalaka, D. Osmanković, J. Velagić, and N. Osmić, "Active slam-based algorithm for autonomous exploration with mobile robot," in *IEEE ICIT*, 2015, pp. 74–79.
- [20] D. Mammolo, "Active slam in crowded environments," Autonomous Systems Lab, ETH Zurich, March 2019.
- [21] P.-Y. Lajoie, B. Ramtoul, F. Wu, and G. Beltrame, "Towards collaborative simultaneous localization and mapping: a survey of the current research landscape," *Field Robotics*, vol. 2, pp. 971–1000, Mar. 2022.
- [22] S. Saeedi, m. trentini, M. Seto, and H. Li, "Multiple-Robot Simultaneous Localization and Mapping: A Review," *Journal of Field Robotics*, vol. 33, pp. 3–46, Jan. 2016.
- [23] G. Bresson, R. Aufrère, and R. Chapuis, "Consistent multi-robot decentralized slam with unknown initial positions," in *International Conference on Information Fusion*, 2013, pp. 372–379.
- [24] S. Saeedi, L. Paull, M. Trentini, and H. Li, "Occupancy grid map merging for multiple robot simultaneous localization and mapping," *IJRA*, vol. 30, no. 2, 2015.
- [25] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, "SVO: Semi-Direct Visual Odometry for Monocular and Multi-Camera Systems," *IEEE Transactions on Robotics and Automation*, Nov. 2016.
- [26] S. Gratton, A. S. Lawless, and N. K. Nichols, "Approximate gauss-newton methods for nonlinear least squares problems," *SIAM*, vol. 18, no. 1, pp. 106–132, 2007.
- [27] M. F. Ahmed, M. Maragliano, V. Frémont, and C. T. Recchiuto, "Entropy based multi-robot active slam," Sep. 2023.