Resource-Aware Simulator for Decentralized Active Perception with Multiple Robots

Sandilya Sai Garimella, Zirui Xu, and Vasileios Tzoumas

***Abstract*— We present a simulation environment for evaluating decentralized multi-robot coordination algorithms in active perception tasks. Multi-robot coordination algorithms enable teams of robots to collaboratively achieve tasks such as map exploration, persistent surveillance, and target tracking. However, current simulators do not model realistic resource constraints on communication, computation, and data storage. Our architecture instead enables the following resource-aware capabilities: modeling of physical resource constraints, such as communication delays, Simultaneous Localization And Mapping (SLAM) with selective landmarks for efficient active perception, and resource-aware operation such as action coordination with near-minimal communication, computation, and data storage. More broadly, communication delays, line-of-sight communication, and computational limitations are be simulated to reflect real decentralized systems.**

I. Introduction

Multi-robot teams provide advantages over individual robots for collaborative tasks through decentralized coordination [1]. Simulation environments are critical for advancing decentralized multi-robot research through rapid prototyping and benchmarking algorithms.

Although existing multi-robot simulators such as ARGoS [2] have useful capabilities, such as scalability, and can model robots with limited sensing, they do not account for cyber-physical limitations emerging from robots' limited resources for communication, computation, and data storage.

In this paper, we present a simulator that accounts for such resource constraints, focusing on tasks of active perception.[1] We account for communication delays, line-of-sight communication limitations, and onboard computational constraints, implementing processes for sensing, map reconstruction, and coordination with reduced resource requirements.

Related Work. Several simulators have been developed for multi-robot active perception [3], [4], [5], [6], [7], [8]. We next discuss representative examples and their limitations.

Atanasov et al. [4] presented simulations of decentralized active SLAM using MATLAB. The simulated multi-robot system relied on a globally known map, facing no computation or communication constraints. Also, the SLAM front-end assumed perfect data association, and there was no obstacle avoidance. In contrast, our simulator incorporates communication delays, computational constraints, and line-of-sight communication constraints; it enables the robots to operate using only local maps; and it handles collision avoidance constraints.

Fig. 1. Gazebo simulation of 3 Turtlebot3 robots in house environment. Our simulator accounts for realistic physical constraints such as communication delays, line-of-sight limitations, and computational constraints, and implements processes for sensing, map reconstruction, and coordination with reduced resource requirements.

Chang et al. [5] developed Kimera-Multi, a multi-robot system for metric-semantic SLAM. The system can operate in a fully distributed fashion through peer-to-peer communication. In contrast to Kimera's focus on the estimation problem of metric-semantic SLAM, our focus is on active SLAM, that is, multi-robot action coordination and execution for collecting measurements that enable effective estimation.

Cao et al. [6] focused on multi-robot exploration under limited communication where robots can only communicate within a certain distance range. They propose maintaining local high-resolution and global low-resolution maps to balance detailed coverage planning with directed exploration. Their "pursuit" coordination strategy has robots opportunistically meet to communicate and explore faster. Their pursuit

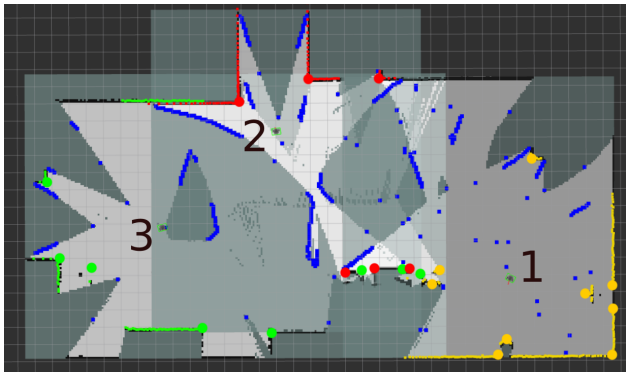

Fig. 2. RViz visualization of 3 numbered robots (in green boxes) with local costmaps, of the same scene as in Fig. 1. Grey regions show discovered occupancy grid map (OGM) cells. Blue points indicate OGM frontiers on each local map. Laserscans for robots 1-3 are yellow, red, and green dots. Larger matching dots are robot landmark features. Emerald-gray and dark grey denote unexplored local and global costmap regions.

The authors are with the Intelligent Robotics and Autonomy Lab (iRaL), University of Michigan, Ann Arbor, MI 48109 USA {garimell, ziruixu, vtzoumas}@umich.edu

[1] https://github.com/sandilyasg/resource_aware_decentralized_sim.git

strategy opportunistically shares information while we focus on simulating active decisions based on information gain. Such strategies could complement our coordination layer.

Contributions. We provide a resource-aware simulator for decentralized active perception with multiple robots. The simulator is based on the Robot Operating System (ROS) [9], and is executable both in simulated and real-world systems. The simulator accounts for resource limitations in both the cyber and physical layers of decentralized multi-robot systems. In more detail, respectively:

- *Physical layer*: The simulator models physical resource constraints that typically stress real multi-robot systems: (i) limited inter-robot communication speed due to communication delays, (ii) limited inter-robot connectivity due to line-of-sight constraints and limited onboard communication range, and (iii) limited information processing due to onboard computation constraints.
- *Cyber layer*: The simulator enables sensing, planning, and control capabilities with reduced requirements for onboard computation, communication, and data storage, by including (i) a dynamic map merging system that merges Occupancy Grid Map (OGM) and local pose graphs, discarding maps after they are used for coordination to conserve onboard resources for data storage, and (ii) an action coordination module for active perception that enables resource-awareness by implementing the Resource-Aware distributed Greedy (RAG) [1] algorithm which has near-minimal requirements for communication, computation, and data storage.

Additional cyber capabilities of our simulator include (i) active perception front-end using the Fast Adaptive Laser Keypoint Orientation-invariant library (FALKOlib) [10] keypoint detection integrated with Georgia Tech Smooth And Mapping (GTSAM) [11] back-end for landmark-based Pose Graph Optimization (PGO) in SLAM, (ii) multi-robot laser scan filter to mitigate robot-as-obstacle errors in SLAM, (iii) integrated an OGM frontier search algorithm to obtain frontiers on local or global OGMs, and (iv) inter-robot and environment object collision avoidance.

Approach. Our simulator is built on C++14, utilizing ROS Melodic and Gazebo 9.19.0. The modular architecture interconnects major components for active SLAM. For sensing, we used the OpenKarto library [12] as the SLAM front-end, which originally implements the Sparse Pose Adjustment (SPA) optimizer back-end. To make the software environment more accessible for computing marginal covariance of poses and landmarks needed for information gain [4], we replaced the SPA optimizer in our map reconstruction module with the GTSAM [11] library for optimizing with landmark-based pose graphs. We used FALKOlib [10] to detect keypoints representing landmarks in the environment — the detected keypoints are potential landmark observations. Newly discovered landmarks are added as new nodes in the GTSAM graph. For landmarks that have been previously observed, we add a pose-to-landmark constraint edge between the robot's current pose and the existing landmark node in the graph. Doing Pose Graph Optimization (PGO) with a landmark-based pose graph enables computing information gain, which is used by our coordination (action planning) module. Our control (action execution) module individually executes the motion of each robot based on the joint control actions selected by the coordination module.

II. Software Architecture of the Simulator

We describe the modular software architecture for simulating decentralized multi-robot tasks of active perception (Fig 3). The architecture consists of custom ROS packages and modified existing packages, and its modularity facilitates customizing the pipeline to suit simulation needs. We simulate multiple Turtlebot3 robots, along with realistic communication, computational, control, and sensing capabilities.

The modules in Fig. 3 cover the pipeline for autonomous active perception: map sensing, map reconstruction, action coordination (action planning), including collision avoidance, and control (action execution). We elaborate below.

A. Physics Engine

Gazebo Simulation Stack. The `turtlebot3` package provides a high-fidelity Gazebo simulation that emulates Turtlebot3 sensor data and physics. The simulator publishes sensor streams over ROS identical to real Turtlebot3s.

Each simulated robot runs the decentralized active SLAM system in Fig. 3. Robots are assigned unique namespace identifiers that prefix variable names We describe each software module in more detail below.

B. Map Sensing Module

Multi-Robot Laser Scan Filtering. When robots are within laser scanner range, their scans interfere by detecting each other as obstacles. This causes inaccurate scan matching and motion estimates as dynamic robots appear static between scans. To address this, the filter module (`multirobot_laserscan_filter`) uses robot location data to remove interfering scan ranges before input to the OpenKarto SLAM layer. Additionally, our coordination module rejects motion plans colliding with other robots. This removed the need for a reactive local planner like the Timed Elastic Band (TEB) [13] that dynamically optimizes trajectories with obstacle avoidance constraints.

Pose-Landmark GTSAM Graph Optimization. A key contribution is using GTSAM's Levenberg-Marquardt optimizer to jointly estimate robot poses and landmark features. This provides the first ROS package, `slam_karto_gtsam_landmark` that enables KartoSLAM to leverage GTSAM's landmark-pose graph optimizer with FALKOlib. GTSAM facilitates obtaining marginal covariances for poses and landmarks, unlike Karto's Sparse Pose Adjustment. We currently assume no outliers.

We maintain the landmark graph with visited landmark locations. New and re-observed landmarks are added with range-bearing constraints reflecting real sensor uncertainty.

To reduce communication bandwidth, we used a custom ROS message which serializes robot pose, landmark, covariances, and graph key information. The RAG algorithm deserializes this message for decision-making.

GTSAM graph optimization occurs after 5 new nodes (robot poses or landmarks) have been added by KartoSLAM's front-end processing, balances the tradeoff between accuracy and computation time. Frequent optimiza-

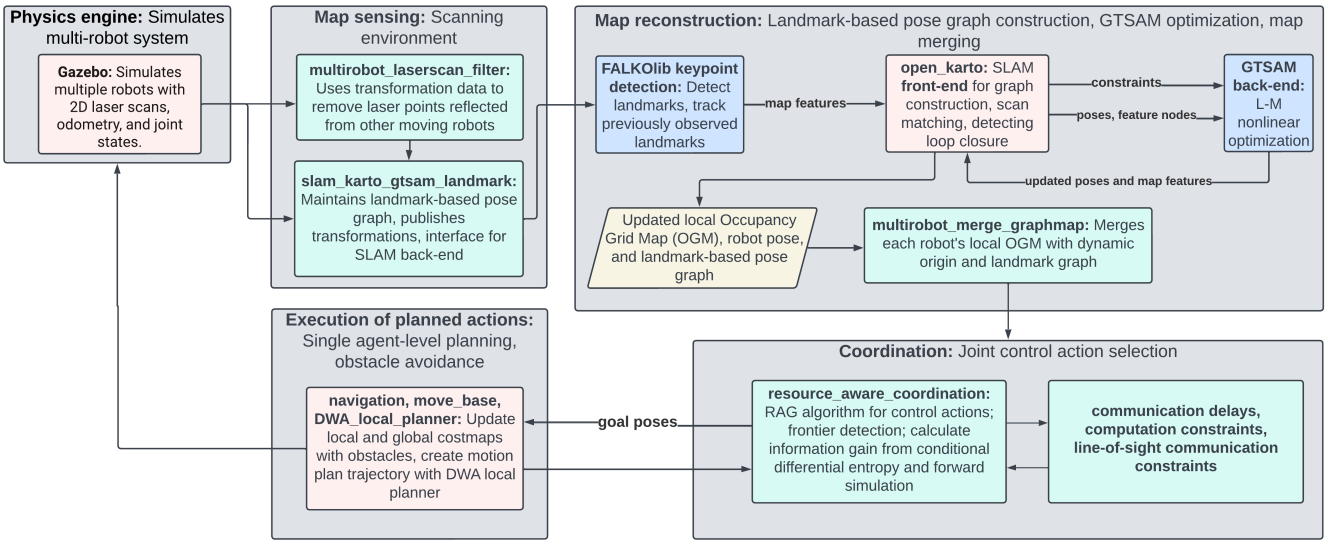

Fig. 3. **Simulator's architecture**. Green blocks are new modules that we created to enable RAG implementation. Pink blocks are existing packages that are used in the simulator after changes. Blue blocks are existing packages that are used in the simulator with no changes. Yellow blocks represent data.

tion introduces efficiency bottlenecks, while infrequent optimization accumulates error. We modified KartoSLAM to dynamically adjust the number of nodes required to trigger an optimization based on the robot's motion. For rotational motion, node thresholds are reduced to bound error during rapid uncertainty growth. For translational motion, node thresholds are increased to reduce unnecessary optimization.

C. Map Reconstruction Module

OpenKarto SLAM Front-End and FALKOlib Keypoint Detection. This module (`open_karto`) extends the existing OpenKarto library, which handles (i) scan matching, (ii) loop closure detection, and (iii) landmark-based pose graph generation. We made modifications to enable 2D laser scan keypoint detection using FALKO, where keypoints are defined as regions of a laser scan with invariance to viewpoints and detection repeatability across frames. This enables tracking of visited landmark keypoints without re-adding them. Re-adding landmarks leads to denser graphs, higher optimization cost, and duplicated nodes. Using persistent landmarks enables efficient long-term operation. We also activate GTSAM optimization when accumulating 5 new nodes or on loop closure, rather than only on loop closure. This enables joint optimization over poses and landmarks, achieving higher accuracy than optimizing over poses only, maintaining the graph sparse.

Dynamic-Origin Occupancy Grid Map and Pose Graph Merger. This module takes in dynamic-origin Occupancy Grid Maps (OGMs) published by KartoSLAM with origins linked to the robot's changing pose. It outputs standardized 384×384 OGMs with fixed $(-10, -10)$ origins to enable multi-robot map merging. The original `multirobot_map_merge` [14] package required fixed OGM origins, but graph-based KartoSLAM outputs local OGMs with varying grid dimensions and shifting origins based on the robot trajectory. This prevented accurate merging. To address this, we created the `multirobot_merge_graphmap` node. It generates new enlarged OGMs and inserts the KartoSLAM occupancy data in the proper cells based on transforming each cell's indices from the dynamic frame to the fixed global frame. This properly resizes and standardizes the maps for merging. Additionally, it merges local landmark graphs into a unified global graph. RAG utilizes the merged landmark covariances for mutual information gain evaluations.

D. Coordination (Action Planning) Module

Decentralized Resource-Aware Coordination Algorithm. We present how the coordination module (`resource_aware_coordination`) enables the robots to select their next actions, *i.e.*, their trajectories at the next time step, in a decentralized resource-aware manner. The robots aim to select actions such that the accumulative reduced uncertainty over detected landmarks is maximized after the selected actions are executed. The metric used by the robots to evaluate the reduced uncertainty is the mutual information of the collected measurements and the landmarks' positions [4]. The objective function used in our coverage planning problem, the mutual information between the collected measurements and landmarks' positions, exhibits diminishing returns and is submodular [15]. Submodularity captures the intuitive notion of diminishing returns, meaning adding a new robot to a small team of robots gives more gain than adding it to a larger team.

Solving the coverage problem is NP-hard [16], and a near-optimal approximation ratio is the $1/2$, which is achieved by the Sequential Greedy algorithm [17]. However, Sequential Greedy cannot scale to many robots since it requires sequential decision-making, resulting in resource requirements that increase linearly with the number of robots [1].

We use instead the Resource-Aware distributed Greedy (RAG) algorithm, introduced in [1]. RAG enables resource-aware decentralized action selection for the robots by solving the coverage problem: *RAG enjoys near-minimal computation, communication, and data-storage resources via enabling parallel decision-making, instead of sequential.*

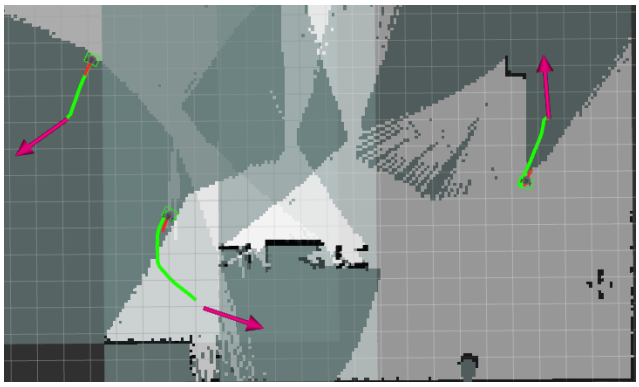

Fig. 4. **Goal pose setting using the RAG algorithm.** Three robots are depicted. The position of each robot is represented at each pink arrow's base. Orientation is indicated by the arrow's direction. Green lines represent planned trajectories. Red lines show the DWA local planner's motion plans.

Now we explain how the RAG algorithm is set up to maximize information gain from discovered landmarks. When landmarks are unavailable, it uses frontier points to encourage exploration and gain information about unexplored space until new landmarks are detected, similar to [4]. We adapted a frontier exploration [18] algorithm to only identify frontier points in the OGM. The module takes in: (i) locations and marginal covariance matrices of robot poses and landmark positions from graph optimization result, (ii) locations and pre-set covariances of frontier points in each robot's frame, and (iii) SE3 transformation matrices from the world frame to each robot's frame. Using these inputs, RAG forms a matrix of Gaussian priors containing the mean positions and covariances for current robot poses, discovered landmarks, and detected frontier points. RAG applies receding horizon control to plan optimal T-step trajectories by solving the active information acquisition problem. This is subject to constraints and objectives from the RAG formulation.

Line-of-Sight Communication. To coordinate communication, the module can identify in-neighbors and out-neighbors of each robot based on communication range and proximity flags provided by the base station. If the robots are within a communicable range, it merges the individual robot's OGMs and local landmark-based pose graphs. Once the sets of in/out-neighbors choose actions, their locally merged OGMs and landmark-based pose graphs are discarded. The base station saves each local OGM continuously and performs merging. It computes the submodular conditional differential entropy of the Gaussian priors through forward simulation [19] and pre-defined control action priors. This determines the mutual information gain of possible control actions. It decides optimal velocity inputs and publishes goal poses to each robot's `navigation` stack, as shown in Fig. 4, executing one step from the T-step trajectory.

Communication Delays and Computation Constraints. To simulate realistic communication delays, the module introduces time delays proportional to the size in bits of messages being sent based on configurable communication bandwidth constraints. Communication using Wi-Fi 802.11b, which utilizes the 2.4GHz frequency band, reaches a peak data rate of 26 Mb/s on the Raspberry Pi 3B+. Meanwhile, lower-data-rate protocols can lead to a communication bandwidth as low as 1 Kb/s. For instance, a GTSAM Pose Graph Optimization (PGO) result message, including covariances, averages about 1604 Bytes. Restricting communication to 1 Kb/s necessitates a publication frequency of approximately 0.623 Hz (1000/1604 $\approx$ 0.623Hz), resulting in an additional delay of 0.604 seconds beyond the 1-second interval. We can also add delays matching the loop execution time of modeled computational hardware to simulate limited onboard processing resources. The computation delay is established similarly to the communication delay, using the onboard processor's processing speed to calculate the proportional constant. These two constraints can be tuned independently, making it possible to simulate heterogeneous teams.

E. Control (Action Execution) Module

The `navigation` package takes in odometry, laser scans, and goals, outputting velocity commands using DWA local planning [20]. It connects the decentralized path planner on each robot to the motion controller, enabling obstacle avoidance. The coordination algorithm sets world frame goals for each robot. Robots navigate to these goals with decentralized collision avoidance, rejecting plans colliding with obstacles in local cost maps.

III. Capabilities and Usage

Simulated Tasks. The simulator currently enables decentralized exploration via multi-robot active SLAM. But modifying the simulator's coordination module can straightforwardly enable additional tasks such as multi-target tracking or persistent surveillance.

Transferring from Simulation to Real Multi-Robot System. The simulated base station emulates an on-site computer for real robot control via WiFi. Decision-making centralization is configurable for full decentralization. Our namespace and topic conventions match real ROS systems for easy transition. The resource-awareness constraints in simulation code are removed, as real onboard platform limitations manifest naturally.

IV. Conclusions

Summary. We provided a resource-aware simulator for decentralized active perception with multiple robots. The simulator is based on ROS, and is executable both in simulated and real-world systems. The simulator accounts for resource limitations in both the cyber and physical layers of decentralized multi-robot systems, integrating FALKOlib for feature detection, GTSAM for map reconstruction, and the RAG algorithm for coordination.

Future Work. We aim to improve the simulator's scalability and overall system performance:

a) Scaling to Larger Multi-Robot Teams. To simulate 30+ robots in large environments, we will transition from Gazebo to Unreal. For lower fidelity 100+ robots simulation, we plan to use ARGoS [2]. To enable efficient optimization for large teams with degraded odometry, we can implement LAGO (Linear Approximation for pose Graph Optimization) [21] instead of GTSAM.

b) Improving system performance. We will further optimize the algorithms and infrastructure, improving pose graph efficiency, and enabling semantic mapping [5].

REFERENCES

[1] Z. Xu and V. Tzoumas, "Resource-aware distributed submodular maximization: A paradigm for multi-robot decision-making," in *IEEE Conference on Decision and Control (CDC)*, 2022, pp. 5959–5966.

[2] C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, F. Ducatelle *et al.*, "Argos: a modular, parallel, multi-engine simulator for multi-robot systems," *Swarm intelligence*, vol. 6, pp. 271–295, 2012.

[3] G. Best, O. M. Cliff, T. Patten, R. R. Mettu, and R. Fitch, "Dec-mcts: Decentralized planning for multi-robot active perception," *The International Journal of Robotics Research*, vol. 38, no. 2-3, pp. 316–337, 2019.

[4] N. Atanasov, J. Le Ny, K. Daniilidis, and G. J. Pappas, "Decentralized active information acquisition: Theory and application to multi-robot SLAM," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 4775–4782.

[5] Y. Chang, Y. Tian, J. P. How, and L. Carlone, "Kimera-multi: a system for distributed multi-robot metric-semantic simultaneous localization and mapping," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 11 210–11 218.

[6] C. Cao, H. Zhu, Z. Ren, H. Choset, and J. Zhang, "Representation granularity enables time-efficient autonomous exploration in large, complex worlds," *Science Robotics*, vol. 8, no. 80, p. eadf0970, 2023.

[7] M. Peti, F. Petric, and S. Bogdan, "Decentralized coordination of multi-agent systems based on pomdps and consensus for active perception," *IEEE Access*, 2023.

[8] R. Tallamraju, E. Price, R. Ludwig, K. Karlapalem, H. H. Bülthoff, M. J. Black, and A. Ahmad, "Active perception based formation control for multiple aerial vehicles," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4491–4498, 2019.

[9] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng *et al.*, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[10] F. Kallasi, D. L. Rizzini, and S. Caselli, "Fast keypoint features from laser scanner for robot localization and mapping," *IEEE Robotics and Automation Letters (RAL)*, vol. 1, no. 1, pp. 176–183, 2016.

[11] F. Dellaert and M. Kaess, "Square root sam: Simultaneous localization and mapping via square root information smoothing," *The Inter. Jour. Rob. Res. (IJRR)*, vol. 25, no. 12, pp. 1181–1203, 2006.

[12] K. Konolige, G. Grisetti, R. Kümmerle, W. Burgard, B. Limketkai, and R. Vincent, "Efficient sparse pose adjustment for 2d mapping," in *IEEE/RSJ Inter. Conf. Intell. Robo. Sys. (IROS)*, 2010, pp. 22–29.

[13] C. Rösmann, W. Feiten, T. Wösch, F. Hoffmann, and T. Bertram, "Trajectory modification considering dynamic constraints of autonomous robots," in *ROBOTIK 2012; 7th German Conference on Robotics*. VDE, 2012, pp. 1–6.

[14] J. Hörner, "Map-merging for multi-robot system," 2016.

[15] A. Krause and D. Golovin, "Submodular function maximization." *Tractability*, vol. 3, no. 71-104, p. 3, 2014.

[16] U. Feige, "A threshold of $ln(n)$ for approximating set cover," *Journal of the ACM (JACM)*, vol. 45, no. 4, pp. 634–652, 1998.

[17] M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey, "An analysis of approximations for maximizing submodular set functions–II," in *Polyhedral combinatorics*, 1978, pp. 73–87.

[18] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97.'Towards New Computational Principles for Robotics and Automation'*, 1997, pp. 146–151.

[19] M. Lauri and R. Ritala, "Planning for robotic exploration based on forward simulation," *Robotics and Autonomous Systems*, vol. 83, pp. 15–31, 2016.

[20] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[21] L. Carlone, R. Aragues, J. A. Castellanos, and B. Bona, "A fast and accurate approximation for planar pose graph optimization," *The Inter. Jour. Robo. Res. (IJRR)*, vol. 33, no. 7, pp. 965–987, 2014.