Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

A Multi-Modal SLAM Cooperative System for UVs operating in GNSS-denied Environments

Karim Walid
German University in Cairo
Omar M. Shehata
German University in Cairo
Elsayed I. Morgan
German University in Cairo
Catherine M. Elias
catherine.elias@guc.edu.eg

German University in Cairo

Research Article

Keywords: Disaster Response, SLAM, Co-Loc, UWB, UAVs

Posted Date: March 25th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-9023546/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License. Read Full License

Additional Declarations: No competing interests reported.

A Multi-Modal SLAM Cooperative System for UVs operating in GNSS-denied Environments

Karim Walid[1,2], Omar M. Shehata[1,2], Elsayed I. Morgan[1], Catherine M. Elias[3*]

[1]Mechatronics Engineering Department, Faculty of Engineering and Materials Science, German University in Cairo, New Cairo, Cairo, 11835, Egypt.

[2]Multi-Robot Systems (MRS) Research Group, New Cairo, Cairo, 11835, Egypt.

[3*]Computer Science and Engineering Department, Faculty of Media Engineering and Technology, German University in Cairo, New Cairo, Cairo, 11835, Egypt.

*Corresponding author(s). E-mail(s): catherine.elias@ieee.org;
Contributing authors: karim.ahmedmahmoud@student.guc.edu.eg; Omar.shehata@ieee.org; elsayed.morgan@guc.edu.eg;

Abstract

Rapid, infrastructure-free situational awareness indoors is critical for disaster response but is hampered by GNSS and fixed infrastructure; key research gaps remain: (i) infrastructure-free cooperative localization techniques are under-explored, (ii) common simulator studies rely on idealized, stationary Ultra-Wideband (UWB) noise models that overestimate field performance, and (iii) there is limited integration and repeatable evaluation of SLAM-enabled leader–follower pipelines under realistic conditions. We present a simulation-driven leader–follower framework where a SLAM-equipped leader supports low-cost followers using an empirically derived, distance-dependent LOS/N-LOS UWB measurement model and a dual-Monte-Carlo Localization with measurement-driven proposals. Across three SLAM methods, Hector SLAM yielded the lowest mapping RMSE; the proposed follower algorithm achieved the best LOS localization trade-off (RMSE = 0.206 m; CPU $\approx$ 58.7%; RAM $\approx$ 401 MB) and degraded to RMSE = 0.630 m under NLOS, comparing favorably with VINS-Mono. Simulator tests show CoppeliaSim demands far less CPU

than Gazebo, and the results indicate that measurement-aware, infrastructure-free cooperative localization combined with lightweight SLAM offers a practical, resource-aware path toward deployable UV teams for search-and-rescue.

Keywords: Disaster Response, SLAM, Co-Loc, UWB, UAVs

1 Introduction

Disasters—whether natural or man-made—cause significant human, economic, and infrastructural losses, requiring rapid and well-coordinated response efforts. Consequently, these multifaceted impacts of disasters demand integrated management approaches that address actions for both pre-disaster and post-disaster. Traditional disaster management frameworks improve organizational structure and communication; however, they remain constrained by reliance on fixed infrastructure, delayed information flow, and human limitations, particularly in dynamic and hazardous environments. A major challenge in modern disaster response is the provision of timely situational awareness and scalable medical aid in environments that are often structurally unstable, inaccessible to human responders or deny Global Navigation Satellite Signals (GNSS) access.[1, 2]

1.1 Disaster Management

Disaster management refers to the comprehensive process of planning, coordinating, and implementing actions designed to mitigate the adverse effects of catastrophic events. According to the literature, disaster management encompasses a range of activities aimed at reducing mortality, protecting property, and restoring essential services following a disaster [2]. It involves not only the immediate response to an incident but also extensive pre-disaster planning and post-event recovery efforts. A critical component of this process is the establishment of structured response systems. For example, the Incident Command System (ICS) and its hospital-specific adaptation, the Hospital Emergency Incident Command System (HEICS), are frameworks developed to ensure rapid, coordinated, and effective action during mass casualty incidents [3]. As illustrated below, these systems break down the management process into several key phases: preparation (which includes developing detailed response plans, conducting simulation drills, and establishing clear communication protocols), the active response phase (marked by the mobilization of resources, execution of command structures, and real-time decision-making), and recovery (which



Fig. 1: Disaster Management Phases[1]

focuses on restoring normalcy and improving resilience against future incidents). Ultimately, disaster management aims to achieve the best possible outcomes for affected populations by minimizing losses—whether in lives, economic resources, or social stability—and by facilitating a seamless transition from crisis to recovery. Despite advancements in SAR technology, several challenges remain. Ensuring rescue worker safety is paramount, as operations can be severely compromised if responders themselves become victims[3]. Strict entry restrictions and training protocols are necessary to ensure that only authorized and adequately prepared personnel enter disaster zones. The use of photo-identification badges and color-coded uniforms helps manage personnel and prevent unauthorized individuals from interfering with SAR efforts. Additionally, controlling traffic flow and securing access routes are critical to maintaining smooth operations for emergency responders.

1.2 SLAM

SLAM enables a robot to build a map of an unknown environment while estimating its pose. Typical stages are initialization, tracking, and mapping [4]. Visual methods (monocular, stereo, RGB-D) remain popular due to low cost, but suffer from scale/depth issues unless fused with inertial or depth sensors (visual–inertial, RGB-D). Graph-based optimization with loop-closure detection has largely supplanted earlier filtering approaches for robust, real-time localization in complex scenes [5]. Multi-sensor fusion (e.g., Inertial Measurement Units (IMUs), cameras, 3D LIDAR) and factor-graph techniques improve accuracy and resilience to challenging conditions [6, 7]. Quantitative surveys report that most indoor UAV studies rely on cameras (54%) or Inertial Navigation (29%), and that fusing UWB with visual odometry can yield sub-decimetre alignment in small test volumes [8, 9]. Collectively, these advances—sensor fusion, efficient optimization, and application-driven design—define current SLAM practice across domains from precision agriculture to search and rescue.

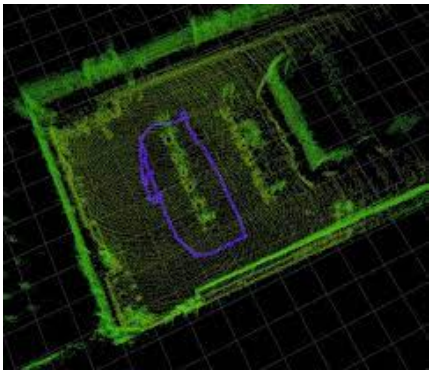


Fig. 2: Output Map sample.

1.3 Cooperative Localization

Cooperative localization is an advanced, collaborative approach that enables multiple UVs or Multi-Robot Systems (MRSs) to share local sensor data—such as measurements from GNSS, IMU, UWB sensors, and bearing-only devices to jointly estimate their positions, velocities, and attitudes. By fusing these heterogeneous data sources using nonlinear estimation techniques such as the Extended Kalman Filter (EKF), or particle filters, cooperative localization can overcome the limitations of individual

dead-reckoning methods, especially in GPS-denied or cluttered environments. Architecturally, cooperative localization can be implemented either centrally, where a fusion center aggregates data from all nodes, or in a decentralized manner, where each agent computes its state using local measurements and relative observations from its neighbors. Such contributions are critical for enabling coordinated behaviors, and collaborative missions in both urban and indoor settings.

1.4 Simulation Software

Simulation software provides a safe, repeatable testbed for designing and validating robotic systems for disaster response, enabling evaluation of sensor suites, communication protocols, control strategies, and swarm behaviours before field deployment [10]. By reproducing diverse failure modes (collapsed buildings, flooding, smoke, GPS-denied areas) and allowing rapid parameter sweeps, simulators help reduce risk and development cost while supporting iterative algorithm tuning and digital-twin workflows.

Comparative studies highlight simulators such as Gazebo and CoppeliaSim for their realistic physics, detailed sensor models, and native integration with middleware like ROS—features that make them well suited for large-scale urban and GPS-denied disaster scenarios and for long-running UAV swarm experiments [11, 12]. In practice, simulator choice is a trade-off between fidelity and computational budget: high-fidelity platforms excel for SLAM, perception and mission-critical validation, while lower-cost engines accelerate algorithm development and large-population experiments.

This paper presents a simulation-driven framework that integrates SLAM, UWB-aided cooperative localization, and mission coordination for leader–follower unmanned robotic teams operating in GNSS-denied scenarios. The main contributions of this work are threefold:

1. A modular simulation pipeline enabling reproducible evaluation in realistic disaster environments.
2. An enhanced cooperative localization implementation based on a dual Monte Carlo Localization (MCL) framework, where the commonly used Gaussian-based simulated UWB ranging model is replaced with an empirically derived measurement model, providing more realistic sensor behavior and improving the fidelity of localization performance evaluation.
3. An experimental assessment of a framework combining a SLAM-powered leader vehicle with UWB-based cooperatively localized followers under representative operational conditions.

The remainder of this paper is organized as follows. Section II reviews related work. Section III presents the proposed system architecture and simulation setup. Section IV describes the experimental methodology. Section V discusses the obtained results, and Section VI concludes the paper with limitations and future research directions.

2 Literature Review

2.1 Evolution of Disaster Response Technologies

Disaster response has evolved from manual coordination (radios, phones, ground transport) to systems that leverage satellite communications, GIS, GPS, remote sensing, mobile/cloud platforms and social media for improved situational awareness and faster damage assessment. UAVs and robotics now extend access to hazardous zones and enable rapid mapping, but challenges remain in infrastructure resilience, data fusion, interoperability and cybersecurity [2]. In recent years, thermal cameras, optical sensors, LiDAR, multispectral cameras, and particle sensors have been used to provide critical information about the environment, helping to detect heat signatures from survivors, assess air quality, and navigate through smoke-filled or low-visibility conditions. The DJI Phantom 4 PRO is one example of a widely used UAV in SAR operations, equipped with high-resolution imaging and real-time data transmission capabilities[10]. Furthermore, projects like ICARUS[1] and INGENIOUS[13] projects have demonstrated the integration of robotic systems in SAR, validating their effectiveness through field exercises in marine and land-based disaster scenarios.

2.2 SLAM

SLAM enables an agent to build a map of an unknown environment while estimating its own pose. Since early formulations and EKF-based solutions, SLAM has matured into two dominant paradigms: filter-based methods (e.g., EKF, particle filters/FastSLAM) and optimization-based approaches (graph optimization, bundle adjustment). Sensor advances have driven a shift from 2D to real-time 3D mapping and from single-sensor systems to multi-sensor fusion (cameras, LiDAR, IMUs), improving robustness and reducing drift in large or dynamic environments [14, 15].

2.2.1 Core components

A typical SLAM pipeline includes:

- **Localization:** estimating the platform's pose via sensor fusion (vision, IMU, wheel encoders, etc.).
- **Mapping:** building sparse or dense representations (features, occupancy grids, point clouds).
- **Odometry:** short-term motion estimation (e.g., visual odometry) that supplies priors to SLAM.
- **Loop closure:** recognizing revisited places to correct accumulated drift.
- **Data association:** matching observations across time to maintain map consistency.

2.2.2 Categories of SLAM

SLAM techniques can be broadly categorized based on the sensors used and the methods for data processing. The main categories include:

1. **Visual SLAM (V-SLAM)** provides high-resolution environmental information useful for victim detection and mapping, integrates naturally with computer-vision-based recognition, and is cost-effective for many UAV and ground-robot applications. However, it benefits substantially from inertial or depth fusion to resolve scale and robustness issues. V-SLAM is further categorized into *Mono-SLAM*, *Stereo-SLAM* and *RGBD-SLAM*.[16–18]
2. **LiDAR SLAM**, which uses laser scanners to generate precise distance measurements. These systems excel in creating accurate 2D or 3D maps, especially in outdoor and unstructured environments.LiDAR systems support feature extraction, scan matching, loop closure and back-end optimization (e.g., LOAM, LIO-SAM). They are particularly effective where range accuracy and geometric fidelity are required [6, 14].
3. **Multi-Sensor Fusion SLAM**, these systems combine data from multiple sensors—such as cameras, LiDAR, and IMUs—to overcome the limitations of any single sensor type. By leveraging complementary sensor modalities, multi-sensor fusion SLAM systems can achieve greater robustness and accuracy, making them well-suited for challenging and dynamic environments.[15] Fusion approaches are commonly classified as loosely or tightly coupled. Loosely coupled systems combine outputs from separate estimators, while tightly coupled systems jointly optimize measurements (e.g., image–point cloud associations, factor graphs) for improved accuracy and resilience in dynamic, cluttered environments [6, 18].

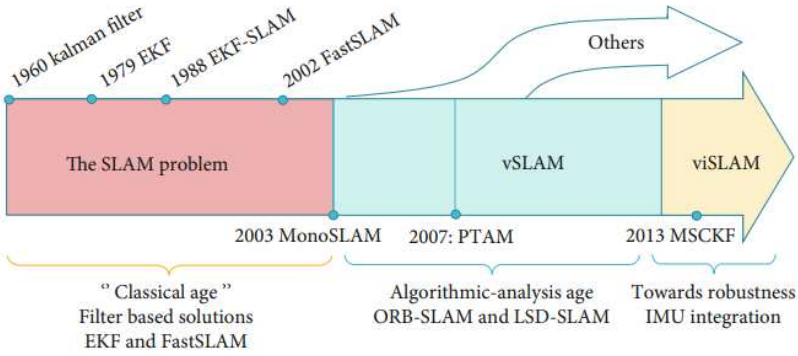


Fig. 3: Overview of V-SLAM history with milestones and the three ages: from the SLAM problem to V-SLAM, V-SLAM algorithmic development, and the emergence of Visual-Inertial SLAM.[16]

2.3 SLAM in Disaster Management

SLAM enhances disaster response by enabling accurate victim localization, persistent situational awareness, and autonomous traversal of hazardous areas. Applications that integrate detection (computer vision) with mapping (e.g., FastSLAM-based workflows)

can track victims and update maps in changing scenes, supporting faster and safer rescue decisions. Despite progress, many solutions still fall short of operational readiness for SAR teams: they often require better system-agnostic data outputs, more robust multimodal coordination, and validation in realistic, time-constrained disaster scenarios [17, 18].

2.4 Cooperative Localization

Tightly coupled factor-graph solutions and multi-agent fusion improve robustness in dynamic and visually degraded environments. Cooperative localization lets agents share observations (relative ranges, poses) to increase observability and reduce single-node failures [19, 20]. UWB ranging is frequently used for pairwise distances in GNSS-denied settings, providing decimeter-level relative accuracy and resilience to non-line-of-sight effects when combined with redundancy and anomaly detection [21, 22].
When executed correctly, cooperative localization significantly enhances accuracy, robustness, and observability of the overall system. For example, integration of UWB ranging with IMU data has demonstrated the potential to reduce indoor localization errors from over 20 cm to below 10 cm [23], while UWB-based systems can offer communication ranges up to 60 m with accuracies in the order of tens of centimeters [22]. Moreover, distributed algorithms have shown improvements in localization accuracy by as much as 83.3% in GPS-denied scenarios over 180-meter test runs [24].

2.4.1 Sensor Data Modelling

Many simulation studies adopt idealized sensor noise models (e.g., Gaussian noise), which risk overestimating algorithm performance. Recent works advocate empirical or black-box UWB models derived from real measurements to better capture nonidealities (biases, multipath, packet loss) and thus provide more representative evaluations. There is a clear gap: systematic comparisons between idealized and empirically derived measurement models—especially within cooperative SLAM and leader–follower pipelines—remain limited, hindering confident transfer from simulation to field deployment.

2.5 Leader–follower architectures

Leader–follower control is a pragmatic, scalable paradigm for heterogeneous teams: a map-building leader guides simpler followers that rely on shared map data and relative measurements for localization and task execution. This architecture balances onboard cost and autonomy while facilitating coordinated search, mapping, and logistics in disaster scenarios [25, 26].

2.6 Simulation for development and validation

High-fidelity simulation is essential given the cost and risk of field tests. Robotics simulators such as Gazebo and CoppeliaSim support realistic physics and sensor models, enabling repeatable evaluation of SLAM, multi-agent coordination, and UWB/communication behaviors before deployment [11, 12]. Comparative studies

favor these platforms for their ROS integration, sensor fidelity, and extensibility in disaster-relevant scenarios.

2.7 Summary

Existing literature shows mature SLAM algorithms and growing interest in cooperative, multi-sensor teams for disaster response, but practical deployment is impeded by (i) limited realism in simulated sensors, (ii) insufficient studies on how measurement model choice affects localization robustness, and (iii) a need for integrated leader–follower evaluations under representative GNSS-denied conditions. The present work addresses these gaps by combining SLAM, UWB-based cooperative localization (with an empirical measurement model), and leader–follower mission orchestration within repeatable simulation experiments.

3 Methodology

This section summarizes the design, implementation, and validation of a leader-follower framework for unmanned vehicles conducting search-and-rescue in GNSS-denied environments. A single leader -equipped with UWB modules- performs SLAM to produce an operational map; low-cost followers use UWB ranging to cooperatively localize within that map and concentrate on victim detection. The remainder of the section describes the leader-follower architecture, the simulation hardware/-software setup, rationale for SLAM algorithm selection, implementation details for the empirical-model based UWB dual Monte Carlo Localization filter used and implementation challenges for both components.

3.1 System Architecture

We adopt a leader–follower architecture for cost-efficiency, modularity, and scalability. The leader, equipped with high-fidelity sensors and computational resources, performs continuous mapping; followers leverage the leader's map and UWB-based cooperative localization to perform task-specific duties (e.g., victim search). This separation reduces per-unit hardware requirements and allows the team size to scale without large increases in system complexity. The overall workflow is illustrated in Fig. 4.

3.1.1 Squad Deployment

The framework is designed for rapid deployment following incidents (e.g., earthquakes). An operator launches the squad into the search area; followers are added or reallocated as needed. Telemetry and mapping data stream to the command center to support coordination, improve situational awareness, and accelerate response decisions.

3.1.2 SLAM

The leader's SLAM subsystem is central: upon entering the designated area, it must produce an accurate, up-to-date map with low computational resource consumption.

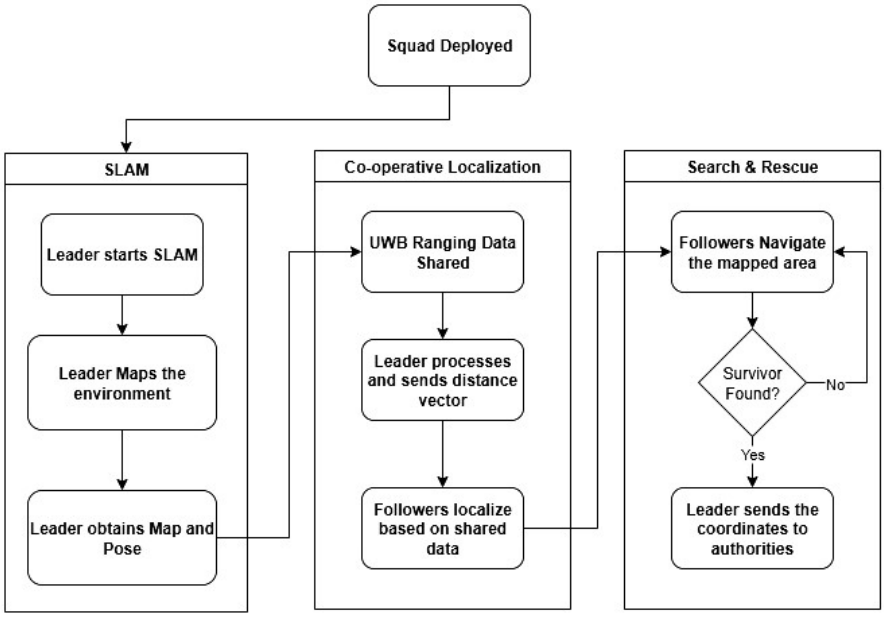


Fig. 4: Proposed system workflow.

Key requirements are spatial precision, and RAM/CPU usage to allow for as much room as possible for the vehicle's endurance and flight duration. These metrics guided the selection and evaluation of the integrated SLAM algorithms.

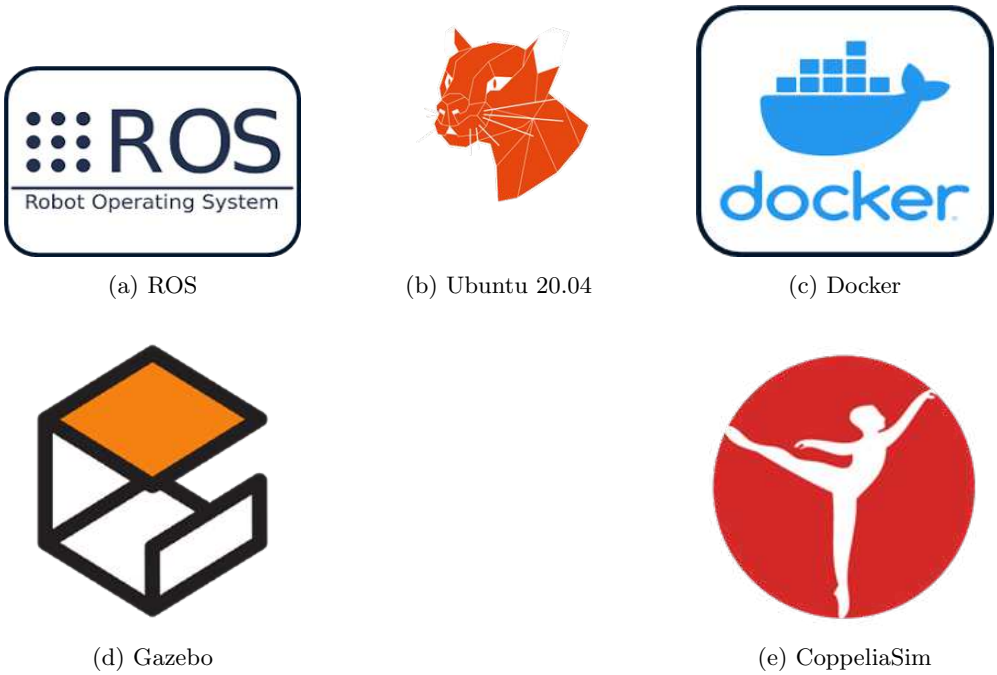
3.1.3 Cooperative Localization

Cooperative localization enables followers to estimate their poses within the leader's SLAM-generated map using UWB ranging. The leader fuses range measurements to each follower and computes relative positions in its coordinate frame before sending it back for processing, allowing navigation and task execution with reduced onboard sensing. Having a common global frame is key in this case considering located victim coordinates should not be inconsistent in terms of their reference of origin. The followers' navigation is guided by the leader's trajectory, which serves as a spatial reference; however, this work focuses on mapping and localization performance rather than on leader path-planning algorithms.

3.2 Software Tools

The experimental stack integrates ROS-based middleware, containerized deployment, and two complementary simulators to enable reproducible, realistic testing of SLAM and cooperative-localization. The primary components are ROS, Docker, Ubuntu 20.04 LTS, Gazebo, and CoppeliaSim. The stack supports running multiple,

resource-constrained Docker containers (each able to host different ROS distributions), orchestrating SLAM nodes, and validating algorithms across physics-faithful and flexible multi-robot scenarios.



(a) ROS (b) Ubuntu 20.04 (c) Docker

(d) Gazebo (e) CoppeliaSim

Fig. 5: Software stack used for development and evaluation.

3.3 Hardware Tools

A single Lenovo Ideapad L340 15IRH (model 81LK00YDED) served as the primary host for local development and for running Docker-based representations of vehicle subsystems. Its key specs (Intel Core i7, 16 GB RAM, 1 TB SSD, NVIDIA GeForce GTX 1650) provide a cost-effective balance of compute, memory, storage, and GPU-assisted rendering—sufficient to run multiple simulator instances and the evaluated stacks under realistic loads.

3.4 Environment Modeling

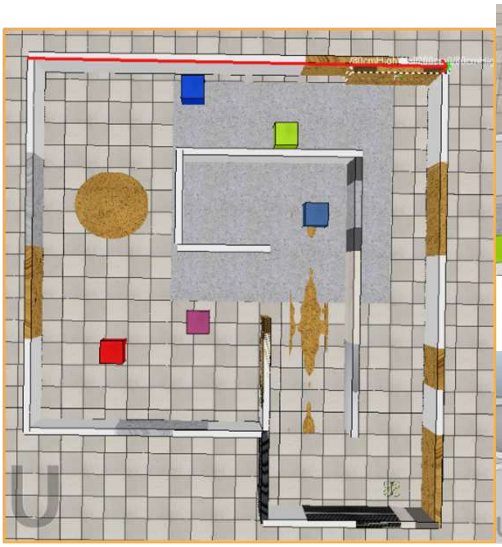
GNSS-denied, earthquake-damaged buildings present a complex environment for robotic mapping and exploration. These environments are characterized by structural compromise and a variety of hazards. Key aspects of these environments include[27]:

- **Structural Damage:** Buildings may suffer significant structural damage, including collapse, which results in inaccessibility for ground robots due to debris or clutter.

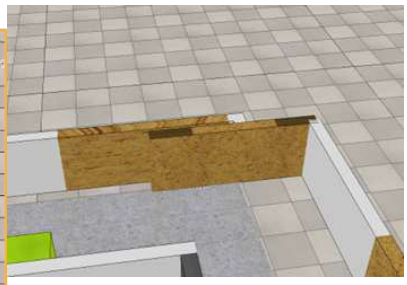
- **Clutter and Debris:** Interiors are typically filled with fallen debris and disarrayed furniture.

3.4.1 Simulated Environment

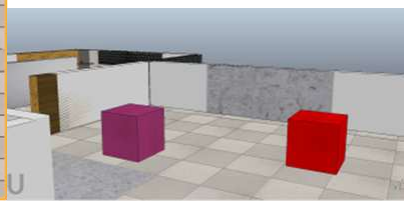
The simulated environment was developed to replicate the complex conditions of GNSS-denied, earthquake-damaged buildings, and it was modeled identically in both Gazebo and CoppeliaSim to enable a direct comparison between the two platforms, however the used vehicles were not identical yet were scaled to be similar in frame size. The simulation faithfully incorporates key environmental aspects, including structural damage with overlapping elements, clutter from disarrayed interiors, randomly displaced obstacles, and a highly varying array of textures and materials. These features provide a realistic testbed for assessing the performance of SLAM and cooperative localization under adverse conditions. The identical modeling across both simulators ensures that any performance differences can be attributed to the simulation platforms themselves rather than discrepancies in the environment. The figures below further elaborate on these aspects, visually demonstrating the fidelity and complexity of the simulated disaster scenarios.



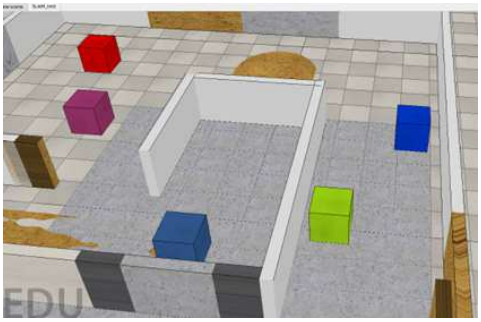
(a) Environment Top View



(b) Overlapping Structure.



(c) Varying Textures.



(d) Randomly Displaced Obstacles including sand clutters.



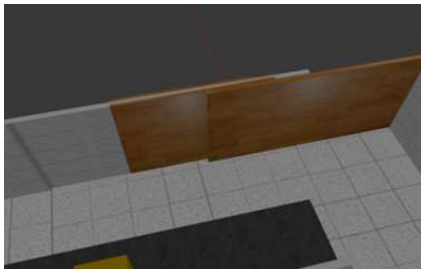
(e) UAV used.

Fig. 6: CoppeliaSim Simulation Environment.

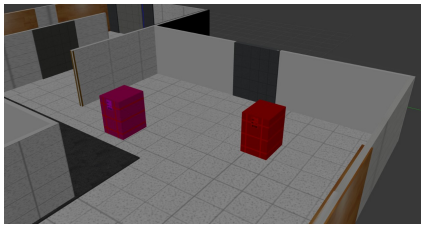
Figure 6 showcases the simulation environment used in CoppeliaSim while the following figures illustrate the environment modeled in Gazebo.



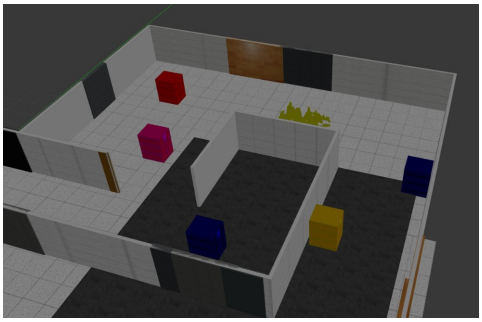
(a) Environment Top View



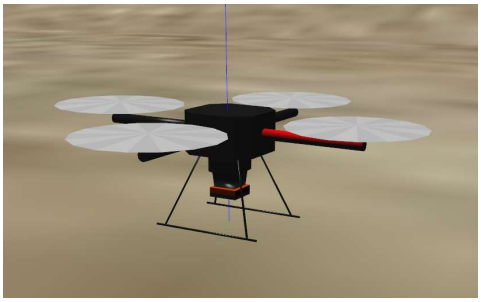
(b) Overlapping Structure.



(c) Varying Textures.



(d) Randomly Displaced Obstacles including sand clutters.



(e) UAV used.

Fig. 7: Gazebo Simulation Environment.

4 SLAM Algorithm Selection

As noted in section 2, effective SLAM in collapsed buildings relies on sensor fusion, robust feature processing, loop-closure, and real-time operation. Representative approaches include feature-based, LiDAR-centric graph SLAM, RGB-D mapping, and multi-resolution occupancy frameworks such as OctoMap. For this study we selected three implementations that address complementary operational needs: 2D LiDAR Hector SLAM, RGB-D RTAB-Map, and 3D LiDAR HDL Graph SLAM.

4.1 Hector SLAM

Hector SLAM was chosen for its low computational footprint and fast, drift-resistant mapping in cluttered indoor scenes, making it suitable for real-time leader operation on constrained platforms. Key strengths are 3D motion estimation for LiDAR-equipped vehicles, scalability across deployments, and open-source availability.[7]

4.2 RTAB-Map (RGB-D)

RTAB-Map provides an online, appearance-based graph SLAM with bounded memory management to support long-term, large-scale mapping. Its practical features include loop-closure via bag-of-words, an explicit Working/Long-Term memory scheme, and support for RGB-D visual odometry and multi-sensor fusion.

4.2.1 Core concepts (RTAB-Map)

- **Graph-based SLAM:** poses are nodes; links encode spatial constraints.
- **Memory management:** Working Memory/Long Term Memory partitioning limits online graph size for fixed-time loop detection.
- **Loop detection:** bag-of-words visual vocabulary (TF–IDF + Bayesian filter) for hypotheses scoring.

- **Odometry & mapping:** supports F2F / F2M visual odometry, PnP RANSAC motion estimation, and 2D/3D occupancy generation from depth images.[28]

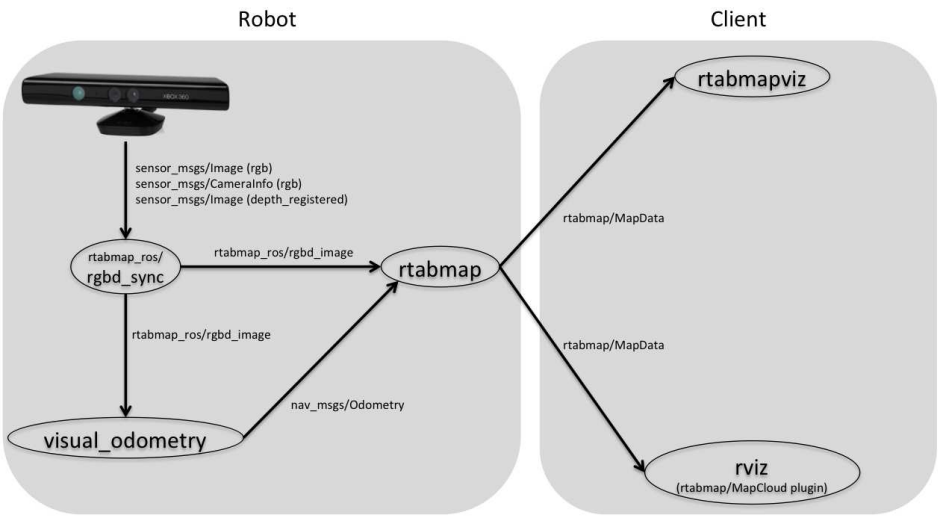


Fig. 8: Kinect-only RTAB-Map process flow. [28]

4.2.2 RGB-D SLAM with Visual Odometry

RGB-D systems produce dense 3D reconstructions useful for victim search in indoor disaster scenes. Visual odometry (F2F/F2M), feature detection/matching (ORB/SIFT/SURF), and PnP-based motion estimation are central to maintaining pose estimates and supporting loop-closure when revisiting areas.

4.3 HDL Graph SLAM (3D LiDAR)

HDL Graph SLAM targets high-accuracy 6-DOF mapping with 3D LiDAR. It combines NDT-based scan matching for odometry, graph optimization with multiple constraint types (odometry, loop closure, GPS, IMU, floor plane), and supports sensors such as Velodyne/RoboSense. Its modular nodelet design (prefiltering, scan-matching odometry, floor detection, graph optimization) facilitates tuning and constraint selection for varied deployment scenarios.

4.3.1 Supported constraints

- **Odometry:** Normal Distributions Transform (NDT)/scan-matching-derived pose increments.
- **Loop closure:** re-localization to reduce drift.
- **GPS / IMU:** optional 3D position and orientation priors (converted/weighted as needed).

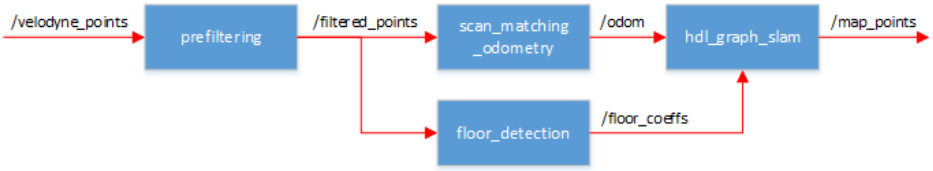


Fig. 9: 3D LiDAR HDL Graph SLAM process flow. [29]

- **Floor plane:** planar constraints to correct accumulated rotation errors indoors.

These three algorithms were selected to cover the trade-space between lightweight, fast 2D mapping (*Hector*), appearance-driven online graph SLAM with RGB-D (*RTAB-Map*), and high-fidelity 3D LiDAR mapping with rich constraint fusion (*HDL Graph SLAM*).

5 Cooperative Localization

This section covers the localization framework that enables followers to estimate their poses within the leader's SLAM map. As discussed, usual methods are classified into geometric/optimization and filtering approaches. Two deployment paradigms are considered: fixed-infrastructure (e.g., use of static anchors) and fully onboard. The proposed study emphasizes infrastructure-free, onboard cooperative localization based on UWB ranging integrated with the leader's global pose.
We adopt an infrastructure-free, two-robot scenario $M = \{R_0, R_1\}$ where R_0 (anchor robot) carries three UWB nodes forming a portable convex hull and R_1 (tag robot) carries one UWB node. The objective is to estimate the relative vector r of R_1 in the anchor frame F_0 using only onboard computation and UWB range readings. The primary estimator developed for this study is a dual Monte-Carlo Localization (dual MCL) variant, based on the implementation in [30], that uses measurement-driven proposals to remain robust under agile motion.

5.0.1 UWB System Modeling

We adopt the black-box UWB simulator approach of Paszek et al. in favor of the more common approach in which range measurements are modeled using a single Gaussian noise term with fixed parameters, independent of distance and environmental conditions. In practice, such simplified Gaussian representations fail to capture the nonlinear and context-dependent behavior of UWB sensors, particularly under Non-Line-of-Sight (NLOS) conditions where systematic bias, heavy-tailed errors, and distance-dependent variance become significant. These effects are especially pronounced in cluttered indoor disaster environments, where multipath propagation, metallic obstructions, and partial occlusions introduce non-stationary error characteristics that cannot be accurately represented by a stationary Gaussian model.
In contrast, the adopted black-box modeling approach [31] uses discrete measurements collected at reference distances under LOS/NLOS conditions, summarized by a mean-bias error (MBE) and standard deviation (σ), which are then modeled as

distance-dependent continuous functions for runtime sampling. This enables the simulator to reproduce realistic sensor degradation patterns and environmental sensitivity, providing a more faithful representation of UWB behavior required for validating localization algorithms intended for safety-critical and infrastructure-free search-and-rescue operations. Shown below is a sample of the data gathered by the aforementioned study:

Table 1: Sample LOS statistics (units: cm).

d_{ref}	d_{mbe}	d_{rmse}	d_{std}	ϵ_{min}	ϵ_{max}
50	2.0	2.7	1.8	-5	9
100	8.5	8.9	2.4	-6	18
200	9.2	9.5	2.3	0	18
500	13.5	13.6	2.2	3	22
1000	18.5	18.7	2.4	10	26
2000	15.6	15.6	1.7	10	21

Table 2: Sample NLOS statistics (units: cm).

d_{ref}	d_{mbe}	d_{rmse}	d_{std}	ϵ_{min}	ϵ_{max}
50	49.8	49.8	1.1	46	57
100	79.4	79.8	7.9	20	86
200	27.4	32.6	17.6	-4	79
500	48.7	49.8	10.5	18	82
1000	9.9	10.1	1.9	3	26
2000	14.9	23.3	18.0	-7	77

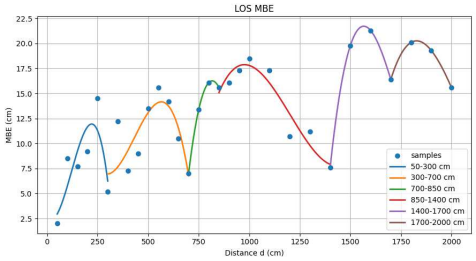
During simulation, we are now able to deduce simulated readings as follows:

$$d' = d + \text{MBE}(d) + \sigma(d)\, R, \qquad R \sim \mathcal{N}(0,1).$$

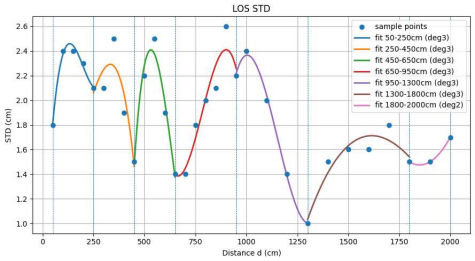
Sample estimators computed from measured samples $\{m_n\}$ at reference distance d_{ref} are

$$\widehat{MBE}(d_{\text{ref}}) = \frac{1}{N}\sum_{n=1}^{N}(m_n - d_{\text{ref}}), \qquad \widehat{\sigma}(d_{\text{ref}}) = \sqrt{\frac{1}{N-1}\sum_{n=1}^{N}\big((m_n - d_{\text{ref}}) - \widehat{MBE}\big)^2}.$$

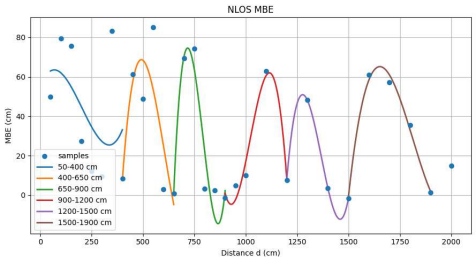
An adaptive, minima-based segmentation is used to fit cubic polynomials to $\widehat{MBE}(d)$ and $\widehat{\sigma}(d)$ on each subrange, producing smooth functions for LOS and NLOS conditions. Applying this methodology yields the following polynomial functions and corresponding plots for our study:



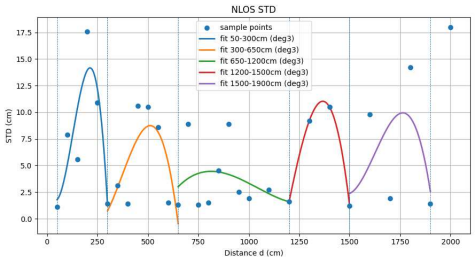
(a) LOS Mean Bias Error Function Plot



(b) LOS Standard Deviation Function Plot



(c) NLOS Mean Bias Error Function Plot



(d) NLOS Standard Deviation Function Plot

Fig. 10: UWB Polynomial Fit Plots

5.0.2 Dual-Particle Filter MCL

The problem is formulated for a two-robot system $M = \{R_0, R_1\}$ in which the robots estimate their relative state using only onboard sensors. The system architecture designates R_0 as the *anchor robot*, equipped with three UWB sensors, treated as if they were anchors and the robot formed its own *portable* convex hull, and R_1 is the *tag robot*, carrying a single UWB sensor. The primary localization goal is to estimate the relative position vector r of R_1 expressed in the anchor robot frame F_0. Crucially, the algorithm operates in a completely distributed manner, running solely on the onboard computer of the anchor robot. No external infrastructure, inter-robot communication, or ground station is required, preserving the fully infrastructure-free nature of the system.

To address the nonlinear and dynamic nature of the state-estimation problem, the authors of [30] employ a *dual Monte-Carlo Localization (MCL)* approach; a modified particle filter formulation. Traditional particle filters often suffer from the particle depletion problem in similar use cases, particularly when the variance of the state transition distribution is significantly smaller than that of the measurement distribution, which is illustrated by Figure 11.

Dual MCL mitigates this issue by inverting the conventional roles of the prediction and update phases. Instead of propagating particles via the motion model and then weighting them using measurement likelihoods, the dual MCL framework uses the

measurement model to construct the proposal distribution. Particles are therefore sampled around the measurement-derived estimate r_{meas}, obtained directly from the three UWB range measurements. The motion model is then used to compute the importance weights π_{aux}, required for the low-variance resampling used.
This inversion ensures that particles remain concentrated around the most recent and informative observations, allowing the system to preserve accuracy even under agile motion conditions, while keeping the particle count relatively small, thereby reducing computational load. Simulations as well as indoor and outdoor experiments with two aerial robots demonstrate that the dual MCL algorithm outperforms both standard particle-filter and extended Kalman filter (EKF) approaches, particularly during rapid and agile motion. The estimation accuracy achieved is shown to be sufficient for downstream tasks such as formation control, enabling the robots to maintain desired relative positions using only UWB-based relative measurements. Furthermore, the authors highlight that the algorithm typically requires fewer than 600 particles in simulation environments, underscoring its efficiency and suitability for onboard implementation in aerial robotics.[30]

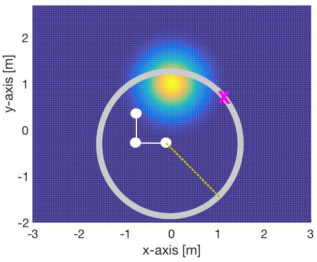


Fig. 11: The motion model output is represented by the gray circle, with three onboard UWB anchors shown as white dots and the true position of robot R1 marked by a magenta cross. If the proposal distribution relies only on the motion model, the particles cluster around the Gaussian peak (yellow), likely missing the robot's true location.[30]

6 Experiments

This section details the experimental evaluation protocols used to compare SLAM and cooperative-localization methods in GNSS-denied, post-earthquake indoor scenarios. Experiments are designed to (i) quantify mapping fidelity and computational cost for three SLAM systems, (ii) measure localization accuracy and resource usage for the Co-Loc approach and benchmark it against other methods, and (iii) compare simulation platforms' performance. All runs use the same environment models and logged ground truth for repeatable, statistically meaningful comparisons.
As discussed in section 3, the software stack used during experiments is containerized and reproducible: the middleware is ROS, containers are managed with Docker, and simulations are executed on two simulators: Gazebo and CoppeliaSim running on an Ubuntu host. Container images, spawn scripts, and recorded bagfiles are archived to permit result reproduction.

6.1 SLAM experiments

6.1.1 Objectives

Measure (a) mapping accuracy (metric agreement with ground truth) and (b) runtime/CPU footprint for:

- **Hector SLAM**
- **RTAB-Map**
- **HDL Graph SLAM**

6.1.2 Experimental setup

- **Environment:** single, identical Gazebo world instance (same map also ported to CoppeliaSim for cross-simulator tests) containing collapsed-structure geometry, displaced debris, and varied textures. Spawn points and waypoint trajectories are fixed.
- **Trajectories:** a pre-defined flight/drive path (Fig. 12) is executed by the vehicle under test. Each algorithm under test is executed on its own Docker container with identical CPU/memory quotas.
- **Repetitions:** each algorithm is executed for $R = 5$ independent runs to reduce run-to-run variability.
- **Logged data:** map outputs (occupancy grids / point clouds), estimated poses, and system resource traces (CPU%, memory) are recorded for offline analysis.

6.1.3 Mapping accuracy: metrics and procedure

For each mapping run we extract a set of n linear measurements from salient environment features (e.g., wall lengths, corner-to-corner distances). For measurement i we record the ground-truth value x_i (from the simulator model) and the SLAM estimate $\hat{x}_i$ (measured on the generated map using RViz/measure tools). The per-run root-mean-square error (RMSE) is:

$$\mathrm{RMSE}_{\mathrm{map}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2}.$$

The algorithm's reported mapping RMSE is the average across R runs:

$$\overline{\mathrm{RMSE}}_{\mathrm{map}} = \frac{1}{R}\sum_{r=1}^{R}\mathrm{RMSE}_{\mathrm{map}}^{(r)}.$$

6.1.4 Computational efficiency

CPU usage is sampled during each run using `docker stats` (container-level CPU%) and the host monitoring tools (system-level metrics). Reported numbers cover the average CPU% per core over the run and values are averaged across the R runs and presented together with standard deviation.

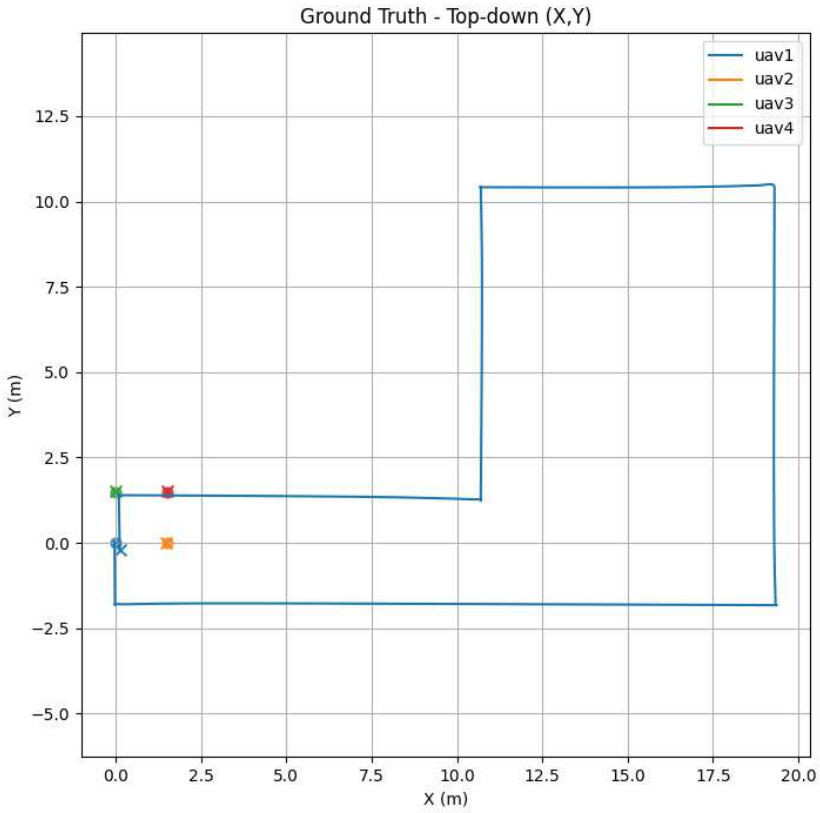


Fig. 12: Reference path followed during SLAM experiments (leader trajectory).

6.2 Cooperative localization experiments

6.2.1 Objectives

To contextualize performance, we compare the proposed algorithm with another method:

1. **VINS-Mono:** a tightly-coupled visual-inertial estimator providing an upper-bound baseline when rich sensors are available.
2. **Dual MCL:** the proposed/implemented particle-filter approach designed for a moving reference node using measurement-based proposal distributions.

The aim is to evaluate follower localization accuracy (in the leader's frame) and resource usage for all three.

6.2.2 Experimental setup

- **Agents:** leader and one follower. Both navigating the configured path where the leader executes SLAM and maintains global pose; the follower runs the tested localization stack.
- **Measurements:** ground-truth poses are logged by the simulator and the algorithm under test outputs its estimated poses.
- **Repetitions:** $R = 5$ runs per algorithm; identical initial conditions and trajectories explained.

6.2.3 Positional accuracy metric

For planar tracking we compute the trajectory RMSE over N time samples (Eq. (1), repeated here):

$$\mathrm{RMSE}_{\mathrm{traj}} = \sqrt{\frac{1}{N}\sum_{k=1}^{N}\|\mathbf{p}_k - \hat{\mathbf{p}}_k\|^2}, \tag{1}$$

and report the mean and standard deviation of $\mathrm{RMSE}_{\mathrm{traj}}$ across the R runs.

6.2.4 Computational/resource tracking

Per-run metrics recorded:

- Average CPU% per core.
- Average RAM usage (MB).

6.3 Simulation platform assessment

6.3.1 Goals

Compare Gazebo vs CoppeliaSim in two dimensions:

- **Computational efficiency:** CPU and memory overhead when running identical SLAM tasks (Hector SLAM used as control).
- **Functionality / usability:** qualitative evaluation along five axes: ROS integration, world modeling, model modification, programmatic control (scripting/APIs), and learning curve.

6.3.2 Computational experiment

- Run Hector SLAM in an identical environment in each simulator, $R = 5$ times.
- Record host CPU% and memory usage (System Monitor) and container usage (`docker stats`).

6.3.3 Functionality scoring

For each of the five functionality axes we assign a +/- indicating the authors' preference after practical review. The five functionality axes are as follows:

- **ROS Integration:** The ease of integrating ROS with the simulator.

- **World Modeling:** An evaluation of the comprehensiveness and user friendliness for building a desired digital environment.
- **Model Modification:** Assessing how modular existing models are and the accessibility offered by the simulator to modify them.
- **Programmatic Control:** Simulator control accessibility (e.g. APIs)
- **User-friendliness:** The overall experience with the simulator's GUI

7 Results

This chapter summarizes the experimental outcomes for the evaluated SLAM algorithms and simulation platforms. Results are split into SLAM performance (accuracy and CPU) and localization experiments (accuracy, resource use, trade-offs), followed by a short simulation assessment.

7.1 SLAM Performance

Each SLAM method was executed five times in Gazebo; reported values are averages.

7.1.1 Mapping accuracy

7.1.2 Hector SLAM

Average RMSE: **0.03** m (five runs).

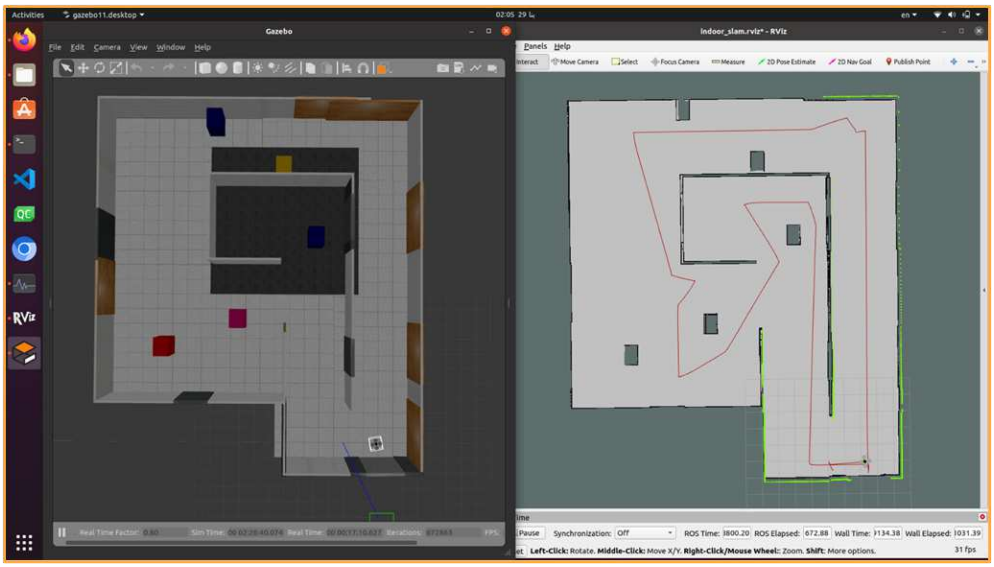


Fig. 13: Hector SLAM map.

7.1.3 RGBD RTAB-Map

Average RMSE: **2.12** m (five runs).

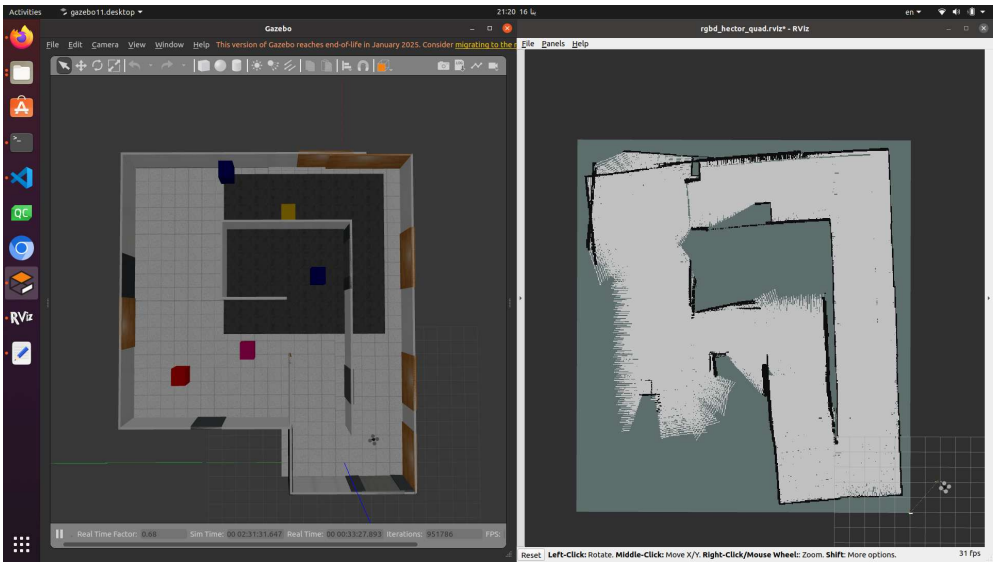


Fig. 14: RGBD RTAB-Map 2D output.

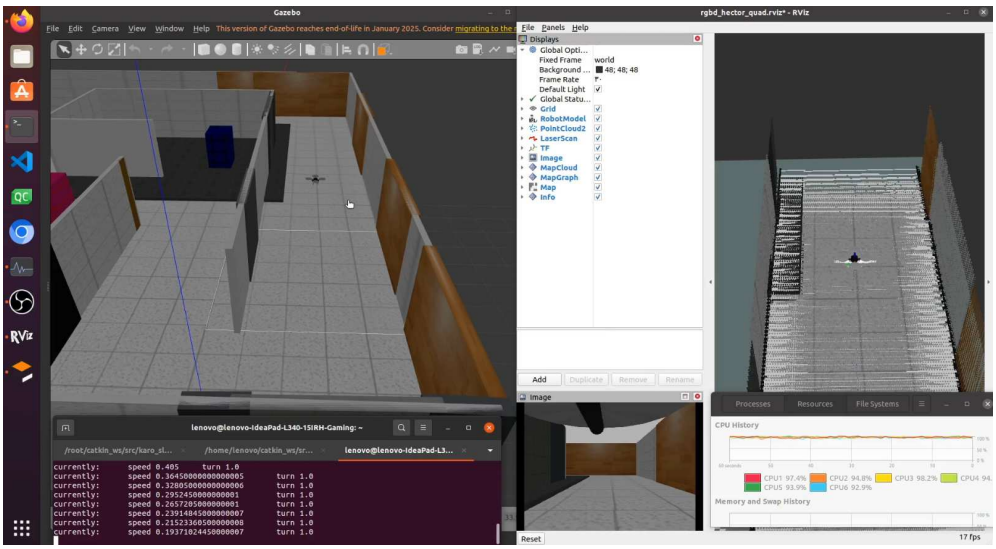


Fig. 15: RGBD RTAB-Map output.

7.1.4 3D LiDAR (HDL Graph SLAM)

Average RMSE: **0.20** m.

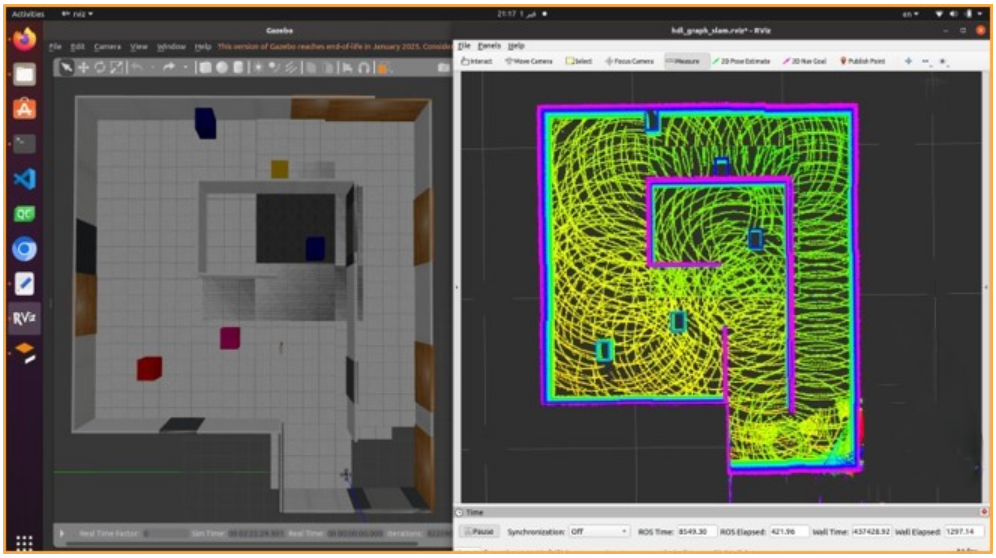


Fig. 16: 3D LiDAR (HDL Graph SLAM) map.

7.1.5 CPU usage

CPU was measured via Ubuntu System Monitor during five runs per algorithm:

- **Hector SLAM**: **70.82**%
- **RGBD RTAB-Map**: **89.83**%
- **3D LiDAR (HDL)**: **85.72**%

7.2 Localization Experiments

All localization metrics are averages over five independent runs.

7.2.1 Positional accuracy

Table 3: Trajectory RMSE per run (m)

Algorithm	Run 1	Run 2	Run 3	Run 4	Run 5	Mean
VINS-Mono	0.417	0.835	0.667	0.180	0.636	0.547
Dual-Particle MCL (LOS)	0.296	0.070	0.240	0.179	0.245	0.206
Dual-Particle MCL (NLOS)	0.775	0.510	0.679	0.580	0.606	0.630

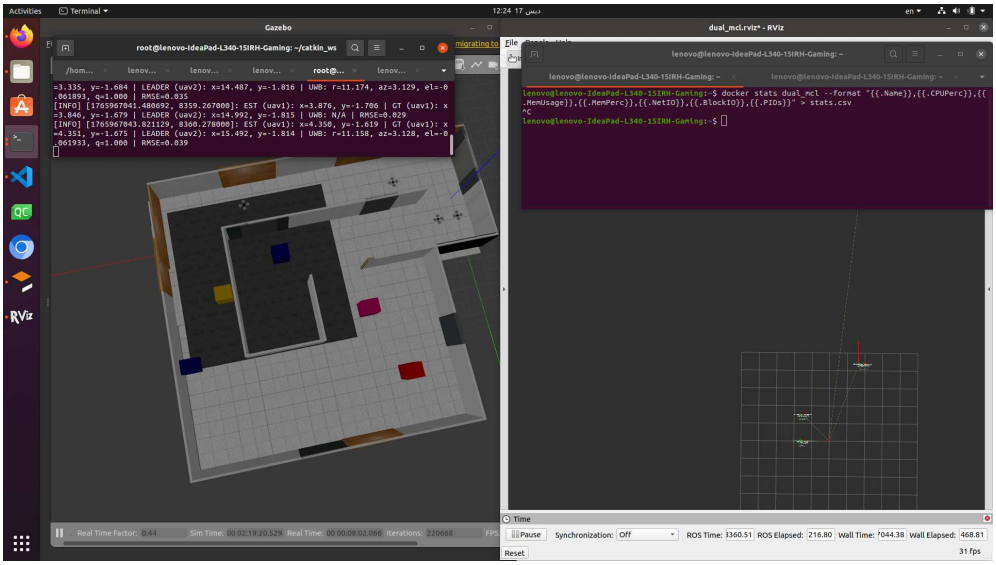


Fig. 17: Snapshot from a Dual-Particle MCL (LOS) run.

7.2.2 Computation consumption

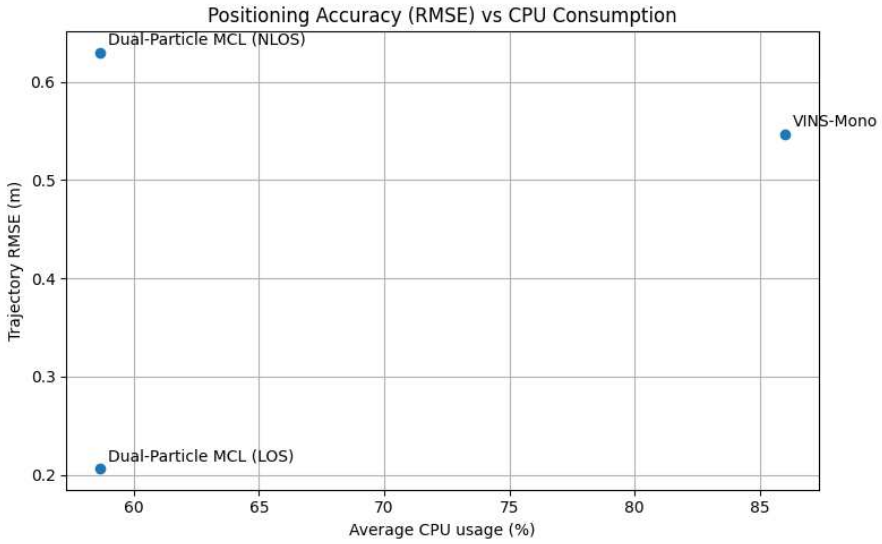
Table 4: CPU (%) — per run and average.

Algorithm	R1	R2	R3	R4	R5	Avg
VINS-Mono	81.08	85.76	84.16	89.10	89.90	86.00
Dual-Particle MCL	60.87	58.82	59.00	59.21	55.43	58.67

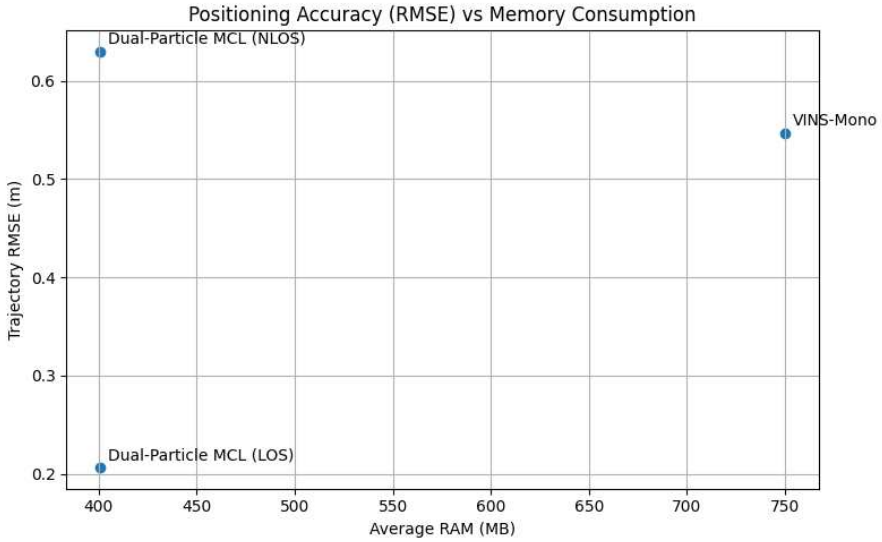
Table 5: Memory (MB) — per run and average.

Algorithm	R1	R2	R3	R4	R5	Avg (MB)
VINS-Mono	762.80	702.98	754.58	771.95	757.19	749.9
Dual-Particle MCL	375.00	375.40	376.00	425.50	452.70	400.92

7.2.3 Accuracy vs. resource trade-off



(a) Accuracy vs CPU



(b) Accuracy vs RAM

Fig. 18: Accuracy vs resource consumption for evaluated methods.

Key observations (concise):

- **VINS-Mono**: RMSE = 0.547 m, CPU ≈ 86%, RAM ≈ 750 MB — high compute cost without best accuracy.
- **Dual-Particle MCL (LOS)**: RMSE = 0.206 m, CPU ≈ 58.7%, RAM ≈ 401 MB — best accuracy/performance trade-off for onboard deployment.
- **Dual-Particle MCL (NLOS)**: RMSE = 0.630 m (degraded accuracy under poor measurements) with similar resource use to LOS.

7.3 Simulation Software Assessment

7.3.1 CPU usage

Hector SLAM mapped the same scene in both simulators (five runs). Average CPU:

- **Gazebo**: **70.82**%
- **CoppeliaSim**: **25.28**%

7.3.2 Functionality

Comparison across six criteria ("+" indicates preferred platform):

Table 6: Simulator functionality comparison

Criteria	Gazebo	CoppeliaSim
ROS integration	+	-
World modeling	-	+
Model modifications	-	+
Programmatic control	+	+
CPU usage	-	+
Usability / learning	-	+

Gazebo provides stronger ROS support; CoppeliaSim offers easier world/model editing, lower CPU, and faster prototyping for multi-robot scenarios.

7.4 Overall discussion

Table 7: SLAM summary: average RMSE and CPU

Algorithm	Avg RMSE (m)	Avg CPU (%)
Hector SLAM	0.03	70.82
RGBD RTAB-Map	2.12	89.83
3D LiDAR (HDL)	0.20	85.72

Hector SLAM combines the lowest RMSE with comparatively low CPU cost, making it attractive for real-time, resource-constrained search-and-rescue platforms. Dual-Particle MCL (LOS) offers the best localization accuracy vs resource trade-off among the evaluated estimators. CoppeliaSim is preferable for rapid multi-robot prototyping due to lower CPU and easier modeling, while Gazebo remains the choice for tight ROS integration.

8 Conclusion and Future Work

This thesis evaluated multiple SLAM algorithms and simulators for post-earthquake, GNSS-denied search-and-rescue scenarios. Results show that low-latency, robust SLAM with sensor fusion is essential for reliable mapping and localization in cluttered environments. In particular, Hector SLAM demonstrated excellent real-time suitability due to its low computational overhead and fast processing, while CoppeliaSim offered notable CPU efficiency and ease-of-use compared to Gazebo—important when scaling to multi-robot systems.

8.1 Conclusions

The main contributions are:

- Comparative evaluation of three SLAM methods (Hector SLAM, RGBD RTAB-Map, 3D LiDAR HDL Graph SLAM) in a simulated post-disaster environment.
- Quantitative assessment using RMSE and CPU metrics to highlight trade-offs between mapping precision and computational cost.
- Simulator comparison using CPU measurements and qualitative criteria, demonstrating Gazebo's stronger ROS integration and CoppeliaSim's advantages in CPU use and rapid prototyping.

These findings guide the selection of SLAM and simulation tools for real-time, resource-constrained SAR platforms: Hector SLAM for lightweight, fast mapping; Dual-Particle MCL (LOS) for best localization accuracy–cost trade-off; CoppeliaSim for multi-robot prototyping.

8.2 Future work

Priority directions for continued research:

- **Multi-robot cooperative localization:** Integrate UWB/IMU and advanced cooperative estimators to improve robustness in GNSS-denied settings.
- **Field validation:** Conduct extensive real-world trials to verify simulator-derived results under realistic operational conditions.
- **Hybrid SLAM:** Develop and test approaches that fuse visual and LiDAR modalities to reduce drift and increase map fidelity.
- **ML-assisted adaptation:** Apply learning methods for adaptive sensor fusion and on-the-fly strategy selection based on environment cues.
- **Richer simulation realism:** Add dynamic obstacles, occlusions, and disturbance models to better mimic post-disaster environments.

Addressing these areas will improve resilience and performance of autonomous SAR systems and help translate simulation gains into operational capability.

Supplementary information. Not applicable.

Acknowledgements. Not applicable.

Declarations

- **Funding:** Not applicable.
- **Conflict of interest:** The authors declare no competing interests.
- **Ethics approval and consent to participate:** Not applicable.
- **Consent for publication:** Not applicable.
- **Data availability:** Not applicable.
- **Code availability:** Not applicable.
- **Author contribution:** K.W. focused on the Conceptualization of this study, Methodology, Software, conceived and conducted experiments. O.M. and C.E. focused on Conceptualization of this study, Analysis of the results, and the manuscript revision. E. M. contributed to The manuscript revision.

References

[1] Jorge, V.A.M., Granada, R., Maidana, R.G., Jurak, D.A., Heck, G., Negreiros, A.P.F., Santos, D.H., Gonçalves, L.M.G., Amory, A.M.: A survey on unmanned surface vehicles for disaster. Sensors **19**(3), 702 (2019) https://doi.org/10.3390/s19030702

[2] Ibrahim, Z.T., He, J.: Slam technology on disaster response. World Journal of Engineering and Technology **12**, 695–714 (2024) https://doi.org/10.4236/wjet.2024.123043

[3] O'Neill, P.A.: The abc's of disaster response. Scandinavian Journal of Surgery **94**, 259–266 (2005)

[4] Macario Barros, A., Michel, M., Moline, Y., Corre, G., Carrel, F.: A comprehensive survey of visual slam algorithms. Robotics **11**, 24 (2022) https://doi.org/10.3390/robotics11010024

[5] Singandhupe, A., La, H.M.: A review of slam techniques and security in autonomous driving. In: Proceedings of the Third IEEE International Conference on Robotic Computing (IRC) (2019). https://doi.org/10.1109/IRC.2019.00122

[6] Xu, X., Zhang, L., Yang, J., Cao, C., Wang, W., Ran, Y., Tan, Z., Luo, M.: A review of multi-sensor fusion slam systems based on 3d lidar. Remote Sensing **14**, 2835 (2022) https://doi.org/10.3390/rs14122835

[7] Kohlbrecher, S., Stryk, O., Meyer, J., Klingauf, U.: A flexible and scalable slam system with full 3d motion estimation. In: Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2011)

[8] Pérez, M.C., Gualda, D., Vicente, J., Villadangos, J.M., Ureña, J.: Review of uav positioning in indoor environments and new proposal based on us measurements. In: Proceedings of an International Conference on Indoor Positioning and Navigation (2019). Work in Progress Paper

[9] Ebadi, K., Bernreiter, L., Biggie, H., Catt, G., Chang, Y., Chatterjee, A., Denniston, C.E., Deschênes, S.-P., Harlow, K., Khattak, S., Nogueira, L., Palieri, M., Petráček, P., Petrlík, M., Reinke, A., Krátký, V., Zhao, S., Agha-mohammadi, A.-A., Alexis, K., Heckman, C., Khosoussi, K., Kottege, N., Morrell, B., Hutter, M., Pauling, F., Pomerleau, F., Saska, M., Scherer, S., Siegwart, R., Williams, J.L., Carlone, L.: Present and future of slam in extreme underground environments. Unspecified (2022)

[10] Sivakumar, M., TYJ, N.M.: A literature survey of uav usage for civil applications. Journal of Aerospace Technology and Management **13**, 4021 (2021) https://doi.org/10.1590/jatm.v13.1233

[11] Nogueira, L.: Comparative analysis between gazebo and v-rep robotic simulators. Gyroscopy and Navigation **13**(4), 241–252 (2022). DOI not provided.

[12] Phadke, A., Medrano, F.A., Sekharan, C.N., Chu, T.: Designing uav swarm experiments: A simulator selection and experiment design process. Sensors **23**, 7359 (2023) https://doi.org/10.3390/s23177359

[13] Karam, S., Nex, F., Chidura, B.T., Kerle, N.: Microdrone-based indoor mapping with graph slam. Drones **6**(11), 352 (2022) https://doi.org/10.3390/drones6110352

[14] Abaspur Kazerouni, I., Fitzgerald, L., Dooly, G., Toal, D.: A survey of state-of-the-art on visual slam. Expert Systems With Applications **205**, 117734 (2022) https://doi.org/10.1016/j.eswa.2022.117734 . Available online 3 June 2022. Open access under the CC BY license.

[15] Taheri, H., Xia, Z.C.: Slam: Definition and evolution. Engineering Applications of Artificial Intelligence **97**, 104032 (2021) https://doi.org/10.1016/j.engappai.2020.104032

[16] Servieres, M., Renaudin, V., Dupuis, A., Antigny, N.: Visual and visual-inertial slam: State of the art, classification, and experimental benchmarking. Journal of Sensors, 26 (2021) https://doi.org/10.1155/2021/2054828 . HAL ID: hal-03187856

[17] Soni, B.: Detecting human victims in an urban search and rescue scenario. Master's thesis, University of New South Wales, Sydney, Australia (2013). https://doi.org/10.26190/unsworks/17101 . http://hdl.handle.net/1959.4/53922

[18] Delmerico, J., Mintchev, S., Giusti, A., Gromov, B., Melo, K., Horvat, T., Cadena, C., Hutter, M., Ijspeert, A., Floreano, D., Gambardella, L.M., Siegwart, R., Scaramuzza, D.: The current state and future outlook of rescue robotics. Journal of Field Robotics, 1–21 (2019) https://doi.org/10.1002/rob.21887

[19] Goel, S., Kealy, A., Gikas, V., Retscher, G., Toth, C., Brzezinska, D.-G., Lohani, B.: Cooperative localization of unmanned aerial vehicles using gnss, mems inertial, and uwb sensors. Journal of Surveying Engineering **143**(4), 1–18 (2017) https://doi.org/10.1061/(ASCE)SU.1943-5428.0000230

[20] Khelifi, M., Butun, I.: Swarm unmanned aerial vehicles (suavs): A comprehensive analysis of localization, recent aspects, and future trends. Journal of Sensors **2022**, 8600674 (2022) https://doi.org/10.1155/2022/8600674

[21] Salimpour, S., Torrico Morón, P., Yu, X., Westerlund, T., Peña-Queralta, J.: Exploiting redundancy for uwb anomaly detection in infrastructure-free multi-robot relative localization. Frontiers in Robotics and AI **10**, 1190296 (2023) https://doi.org/10.3389/frobt.2023.1190296 . Open Access

[22] Wang, S., Martínez Almansa, C., Peña Queralta, J., Zou, Z., Westerlund, T.: Uwb-based localization for multi-uav systems and collaborative heterogeneous multi-robot systems: A survey. IEEE Access (2020)

[23] Liu, Y., Yan, R., Lian, B., Zhao, H.: Hybrid imu/uwb cooperative localization algorithm in single-anchor networks. IEEE Geoscience and Remote Sensing Letters **19**, 3511605 (2022) https://doi.org/10.1109/LGRS.2022.3165933

[24] Hartzer, J., Saripalli, S.: Vehicular teamwork: Collaborative localization of autonomous vehicles. In: Proceedings of the IEEE Intelligent Transportation Systems Conference (ITSC) (2021). https://doi.org/10.1109/ITSC48978.2021.9564981

[25] Zhuang, H., Dong, K., Wang, N., Dong, L.: Multi-robot leader grouping consistent formation control method research with low convergence time based on nonholonomic constraints. Applied Sciences **12**, 2300 (2022) https://doi.org/10.3390/app12052300 . Open Access under CC BY 4.0

[26] Niculescu, V., Polonelli, T., Magno, M., Benini, L.: Ultra-Lightweight Collaborative Mapping for Robot Swarms. Submitted for publication, IEEE (2023). https://github.com/ETH-PBL/Nano-C-SLAM

[27] Michael, N., Shen, S., Mohta, K., Mulgaonkar, Y., Kumar, V., Nagatani, K., Okada, Y., Kiribayashi, S., Otake, K., Yoshida, K., Ohno, K., Takeuchi, E.,

Tadokoro, S.: Collaborative mapping of an earthquake-damaged building via ground and aerial robots. Journal of Field Robotics **29**(5), 832–841 (2012) https://doi.org/10.1002/rob.21436

[28] Labbé, M., Michaud, F.: Rtab-map as an open-source lidar and visual slam library for large-scale and long-term online operation. Journal of Field Robotics (2024). Preprint version. Final version available at https://doi.org/10.1002/rob.21831

[29] Koide, K.: hdl_graph_slam: 3D LIDAR-based Graph SLAM. https://github.com/koide3/hdl_graph_slam (2018)

[30] Guler, S., Abdelkader, M., Shamma, J.S.: Infrastructure-free localization of aerial robots with ultrawideband sensors. arXiv preprint arXiv:1903.05316 (2019). Available at: https://arxiv.org/abs/1903.05316

[31] Paszek, K., Grzechca, D., Becker, A.: Design of the UWB positioning system simulator for LOS/NLOS environments. Sensors **21**(14), 4757 (2021) https://doi.org/10.3390/s21144757