Proceedings of the 2023 International Conference on Machine Learning and Automation
DOI: 10.54254/2755-2721/41/20230755

Weather-adaptive slam using unmanned ground vehicles

Yixing Long

Vanke Meisha Academy, No. 33, Dameishahuanmei Rd, Yantian District, Shenzhen, Guangdong

edward219810@outlook.com

Abstract. This paper presents a weather-adaptive SLAM system for Unmanned Ground Vehicles (UGVs) equipped with lidar, radar, and RGBD-camera sensors. These sensors and functionalities are built in Robot Operating System (ROS) Noetic. The system dynamically selects the best sensor combination along with the corresponding slam strategies based on real-time weather data acquired through the Python web crawler. Our work ensures accurate mapping and localization in various and fast-changing weather conditions, enhancing SLAM reliability and accuracy.

Keywords: Weather-adaptive SLAM, UGV, ROS, RGBD-camera, LiDAR, Radar, Web Crawler.

1. Introduction

Simultaneous Localization and Mapping (SLAM) plays a critical role in autonomous navigation, enabling a robot or vehicle to simultaneously create a map of its environment and localize itself within the map. The generated map is essential in various robotic applications, such as autonomous driving and warehouse automation, and hence has being the focus of the research frontier.

The objective of this project is to develop a weather-adaptive SLAM system for a multi-sensor robotic platform equipped with lidar, radar, and camera sensors. Traditional SLAM algorithms often struggle to adapt to different weather conditions, leading to decreased performance and potential hazards. For example, in extreme weather conditions, such as heavy rain or thick fog, SLAM accuracy may be compromised, potentially resulting in accidents. Additionally, these inaccuracies in SLAM can lead to incorrect vehicle trajectories, causing extended travel times and traffic congestion. As a countermeasure to these shortcomings and risks, we propose a robust SLAM system that relies on real-time weather subscription and automatically selects the most appropriate SLAM strategies with the necessary sensor settings.

Existing SLAM methods typically assume a constant and known environment, which becomes a challenge in scenarios with fast changing weather conditions. Different weather conditions, such as sunny/rainy, bright/foggy, and even different time in a day, such as daytime/nighttime, can significantly impact the performance of specific sensors in SLAm. For instance, cameras might struggle to perceive surroundings in low-light conditions, and lidar sensors may face challenges in rainy or foggy weather due to extremely noisy signals.

To achieve reliable navigation and mapping across different weather conditions, our project focuses on three candidate slam solutions:

DOI: 10.54254/2755-2721/41/20230755

1. RGBD-Camera SLAM for Sunny and Daytime Conditions:

During daytime and sunny weather, the camera sensor can provide rich visual information to the SLAM system. By leveraging visual features and landmarks, the camera-based SLAM can accurately map the environment, While the DEPTH SENSOR provides reliable depth information and localize the robot within it.

2. RGBD-Camera + Lidar SLAM for Clear Nighttime Conditions:

In clear nighttime conditions, the combination of camera and lidar sensors offers a complementary advantage. While the camera utilizes available light to perceive the environment, the Lidar sensor can provide accurate depth information, enhancing the SLAM's performance even when in low-light situations.

3. Radar SLAM for Rainy and Foggy Weather:

In adverse weather conditions, such as rain and fog, the radar sensor proves to be more robust than camera and *lidar*. Radar waves can penetrate through wintry mix, allowing the SLAM system to maintain localization and mapping capabilities despite reduced visibility.

To implement weather-adaptive SLAM, the robotic platform is designed to work under a wide variety of weathers. We have already taken measures to waterproof the system, seal electronic components, and ensure port insulation. The robot works under Ubuntu 20.04 with ROS Noetic and subscribes to real-time weather data through wifi or cellular data. Based on the received weather information, the SLAM system dynamically selects the appropriate sensor combination from the three options mentioned above. This real-time adaptation ensures optimal performance in varying weather conditions, enhancing the overall reliability and accuracy of the mapping results.The Python-based web crawler for fetching real-time weather data is seamlessly integrated into the ROS nodes, allowing the robotic platform to continuously receive and response to weather updates.

The ROS Noetic serves as the fundamental software platform for our project. ROS Noetic provides a flexible and modular framework for developing robotic applications, making it an ideal candidate for integrating the diverse sensors and implementing the weather-adaptive SLAM algorithm. All related code and implementations are accomplished through ROS Noetic, facilitating seamless communication and integration between the robot's sensors and the SLAM system.

2. Related Research

In the domain of Simultaneous Localization and Mapping (SLAM) for Unmanned Ground Vehicles (UGVs), various sensor modalities, including LIDAR, cameras, and radar, have demonstrated their utility. However, each of these sensors faces inherent limitations when confronted with adverse weather conditions. LIDAR systems encounter inefficiencies during rainy and low-visibility weather, cameras struggle with image degradation. Even while radar sensors exhibit resilience to environmental variations, their lower angular accuracy compared to LIDAR and camera sensors can limit their precision. These limitations underscore the critical necessity for a weather-adaptive SLAM system, the focus of this paper, which dynamically selects the most suitable sensor combination based on real-time weather data to ensure robust and accurate SLAM performance.

Weather Impact on Camera-based SLAM: Camera-based SLAM is widely adopted and was validated in many studies. But camera-based slam also has its limitations. For instance, Hong et al. (2021) demonstrated that image quality significantly degrades in adverse weather conditions such as snow, fog, rain, and night, rendering vision-based odometry and SLAM algorithms highly challenging. In particular, the study observed that camera sensors could be completely covered by snow, leading to a complete loss of visual information. To address these challenges, the authors proposed the use of radar sensors as an alternative sensing modality for SLAM, emphasizing the need for robust SLAM systems capable of performing effectively in extreme weather conditions. This research underscores the vulnerability of camera-based SLAM to adverse weather and the importance of developing weather-adaptive alternatives.

Challenges of LIDAR-based SLAM: In the study of Farzadpour et al. (2018), the limitations of LIDAR-based (SLAM) in adverse weather conditions were highlighted. The research found that LIDAR

DOI: 10.54254/2755-2721/41/20230755

sensors are notably inefficient during rainy seasons and low-visibility weather**.** Additionally, the processing of extensive LIDAR data was identified as a time-consuming and resource-intensive task, highlighting the advantage of using cameras when weather conditions permit. The study introduced a coverage model for mechanical LIDAR sensors and discussed the impact of various parameters on their coverage performance, within geometric constraints.

Radar SLAM for Weather Localization and Mapping: Schoen et al. (2017) explored the use of radar sensors for SLAM, emphasizing their resilience to various weather conditions. While radar sensors were found to have lower angular accuracy compared to camera and LIDAR sensors, they exhibited robustness and independence from weather conditions. The research presented a cost-efficient and accurate SLAM method that utilized radar sensors for 360-degree coverage. The algorithm's evaluation in dynamically changing scenarios demonstrated its potential for achieving reliable SLAM even in extreme weather conditions. This study highlights the trade-offs between sensor accuracy and weather resilience and suggests that radar-based SLAM can be a viable solution for all-weather localization and mapping applications.

3. Method

In this section, we introduce how we identify and categorize weather situations. We explore methods for real-time weather data retrieval using Python, shedding light on how current weather information can be seamlessly integrated into SLAM frameworks. Our discussion encompasses different scenarios, including RGBD-Camera SLAM optimized for sunny and daytime conditions, the fusion of RGBD-Camera and LiDAR for clear nighttime operation, and the utilization of Radar-based SLAM for challenging rainy and foggy weather scenarios. By addressing the influence of weather on SLAM and presenting tailored solutions for varying conditions, we aim to enhance the adaptability and robustness of autonomous systems in dynamic environmental settings.

3.1. Weather Identification Process

Our weather-adaptive SLAM system employs a Python-based web crawler to gather real-time weather data. We assert the superiority of this method based on the accuracy and timeliness. Among the assortment of methods evaluated, this algorithm consistently outperformed alternatives by delivering the most precise and up-to-date weather information. For instance, some methods suffer from slow data update frequencies, resulting in the potential use of outdated information. Additionally, historical detection accuracy may be lower in other approaches. The system subscribes to various online sources providing weather information. To exemplify, we use Google Search to fetch weather updates for a specific region. By sending a request to the appropriate URL, we retrieve the HTML content containing weather details.

The web crawler utilizes the Beautiful Soup library to parse the HTML content and extract the relevant weather information. Specifically, it retrieves the following data:

- Location: The name of the region for which the weather data is being retrieved.
- Current Time: The timestamp indicating the current time when the weather data was fetched.
- Temperature: The current temperature in degrees Celsius.
- Weather Description: A brief description of the current weather conditions (e.g., "Sunny," "Partly Cloudy," etc.).
- Precipitation: The likelihood of precipitation as a percentage.
- Humidity: The current relative humidity as a percentage.
- Wind Speed: The current wind speed in kilometers per hour.

The collected weather data is then preprocessed to categorize weather conditions. For example, different levels of rain or snow are grouped together. This categorization allows us to simplify the weather status and create a more manageable and informative dataset for the SLAM system.

DOI: 10.54254/2755-2721/41/20230755

Sunny	Clear
Clear	Clear
Partly Cloudy	Clear
Mostly Cloudy	Cloudy
Overcast	Cloudy
Light Rain	Rain
Moderate Rain	Rain
Heavy Rain	Rain
Light Snow	Snow
Moderate Snow	Snow
Heavy Snow	Snow
Foggy	Fog
Windy	Windy
Drizzle	Rain
Sleet	Snow
Freezing Rain	Rain
Mist	Fog

Figure 1. Classification of different weather conditions.

For light rain and light snow:

Wind speed >15	Clear
Wind speed <15	Rain / Snow

Additionally, our system performs a nuanced analysis that considers not only the official weather reports but also specific combinations of weather conditions. For instance, when the system detects “light snow" or "light rain" in conjunction with a "high wind speed," it categorizes the weather as "clear" rather than simply labeling it as "rain" or "snow." This approach allows us to offer a more accurate representation of the current weather scenario, considering factors such as visibility and road conditions, which are crucial for our SLAM system's decision-making process.

By continually subscribing to real-time weather updates and integrating the acquired information, our weather-adaptive SLAM system remains informed about the current weather conditions, ensuring accurate adaptation to changing weather scenarios.

DOI: 10.54254/2755-2721/41/20230755

3.2. RGBD-Camera SLAM for Sunny and Daytime Conditions

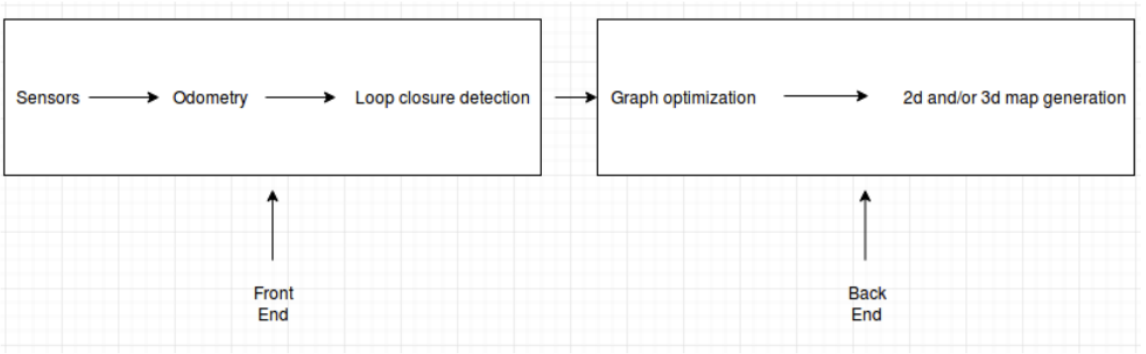

Figure 2. RTAB-Map Front end and Back End block diagram.

RGBD-Camera SLAM, or Simultaneous Localization and Mapping using RGB and Depth cameras, is a robust solution tailored for sunny and well-lit daytime conditions. This method leverages natural light to enhance its effectiveness in mapping and localizing within the environment. In this discussion, we will explore the fundamental characteristics, working principles, map generation strategies, and odometry techniques associated with RGBD-Camera SLAM.

In the realm of characteristics, RGBD-Camera SLAM is inherently adapted to environments characterized by abundant natural sunlight. It thrives in well-lit conditions, capitalizing on the ample illumination provided during the daytime.

The core component of RGBD-Camera SLAM is the RGBD camera, which captures rich visual data. These cameras provide not only color information but also depth data, enabling a comprehensive perception of the surroundings.

The essence of RGBD-Camera SLAM lies in its reliance on visual features for mapping and localization. These visual features encompass various elements, including key points, edges, and distinctive patterns. These features serve as the foundation for both building maps and determining the robot's location within the environment.「1」

In terms of its working principles, RGBD-Camera SLAM operates by capturing high-resolution RGB images using the camera sensor. These RGB images are rich in color and texture, forming the basis for the SLAM process. They are acquired at a high frame rate, allowing for real-time analysis of the environment.

Visual feature extraction is a pivotal step in RGBD-Camera SLAM. The system extracts key visual features, such as corners and edges, from the RGB images. These features are then tracked across frames as the robot moves, involving the matching of features and estimation of their relative positions.

The estimation of robot motion, often referred to as odometry, plays a critical role. Visual odometry techniques are applied to track how the visual features move between frames. This tracking process enables the system to estimate both the robot's position and orientation changes as it navigates the environment.「5」

Simultaneously, a map is constructed. This map represents the robot's perception of its surroundings and how it has evolved over time. The map generation process involves accumulating the visual features that have been tracked over time. These features are stored along with their corresponding 3D positions.

Depending on the specific SLAM algorithm employed, the map can be either sparse or dense. A sparse map includes only a subset of tracked features, while a dense map retains all tracked features, providing a more detailed representation.

To enhance map accuracy, many RGBD-Camera SLAM systems incorporate loop closure detection. This involves identifying when the robot revisits a previously explored area and aligning the new data with the existing map.

DOI: 10.54254/2755-2721/41/20230755

The generated map can be stored in various formats, such as a point cloud, a grid map, or a graph structure, depending on the requirements of the application.「3」

Visual odometry techniques are crucial for accurate localization in RGBD-Camera SLAM. These techniques encompass various stages:

Feature tracking: Visual odometry begins with the tracking of visual features from one frame to the next, relying on accurate feature matching and correspondence.

Triangulation: Once features are tracked across multiple frames, their 3D positions can be estimated through triangulation. This involves intersecting the lines of sight from the camera to each feature in each frame.

Bundle Adjustment: To refine odometry estimates, bundle adjustment techniques are often applied. Bundle adjustment optimizes the entire trajectory of the robot and the 3D positions of the mapped features to minimize errors.

Scale Estimation: Addressing scale drift is a challenge in visual odometry since cameras do not provide depth information directly. Scale estimation techniques may incorporate data from other sensors or exploit known distances in the environment.

- **Front End of Graph SLAM:**
  - **Data Interpretation:** In the front end, the robot processes the sensory data it collects, such as camera images, LIDAR scans, or other sensor readings. This data is used to extract relevant information about the environment, including features like landmarks, obstacles, or other distinctive points.
  - **Graph Construction:** The robot constructs a graph representation of the environment. This graph typically consists of nodes (representing robot poses) and edges (representing constraints between poses). The nodes represent where the robot has been, and the edges encode the relationships between these positions.
  - **Data Association:** One significant challenge in the front end is solving the data association problem. This involves determining whether the features detected in the current sensory data have been seen before. This is crucial for maintaining a consistent map and estimating the robot's trajectory accurately.
- **Back End of Graph SLAM:**
  - **Optimization:** The back end of graph SLAM is responsible for optimizing the graph created by the front end. It aims to find the most probable configuration of robot poses and map features that minimizes the overall error in the graph. This optimization process typically involves solving a nonlinear least squares problem.
  - **Constraint Fusion:** The back end takes into account all the constraints in the graph, including those related to odometry, sensor measurements, and loop closures. By adjusting the robot poses and map features, it refines the map to best fit the collected data.
  - **Consistency:** The back-end's optimization process ensures that the map and trajectory are consistent with the observed data, improving the accuracy of the SLAM solution.

RTAB-Map (Real Time Appearance Based Mapping) is an exemplary graph-based SLAM technique that utilizes vision sensors, such as cameras, to create maps and determine the robot's location. It heavily relies on visual data for feature extraction and loop closure recognition. In dealing with the computational complexity of loop closure, especially as maps expand, RTAB-Map implements efficient strategies to handle these situations in real-time, making it suitable for large-scale and long-term mapping projects. This method adopts a front end for data interpretation and map construction, coupled with a back end responsible for graph optimization, collaborating iteratively to enhance map accuracy and robot pose estimations.

We have a set of nodes, each represented by a parameter vector x, where xi describes the pose of node i. These nodes are connected by virtual measurements zij, and each measurement comes with an associated information matrix Ωij. These virtual measurements represent transformations that align observations from one node with those from another.

DOI: 10.54254/2755-2721/41/20230755

We predict these virtual measurements as zˆij(xi, xj), typically representing the relative transformation between nodes i and j. We measure the agreement between these predictions and the actual observations zij collected by the robot using an error function:

$$e_{ij}(x_i, x_j) = z_{ij} - \hat{z}_{ij}(x_i, x_j) \tag{1}$$

Our goal is to find the optimal configuration of nodes, denoted as x*, that minimizes the negative log-likelihood of all observed measurements. This log-likelihood is calculated as a weighted sum of squared errors, with each error term scaled by the corresponding information matrix:

$$F(x) = \sum_{\langle i,j\rangle \in} \underbrace{e_{ij}^T \Omega_{ij} e_{ij}}_{F_{ij}} \tag{2}$$

To solve this problem, we seek the configuration x* that minimizes F(x):

$$x^* = \underset{x}{\operatorname{argmin}}\, F(x) \tag{3}$$

3.3. RGBD-Camera + Lidar SLAM for Clear Nighttime Conditions

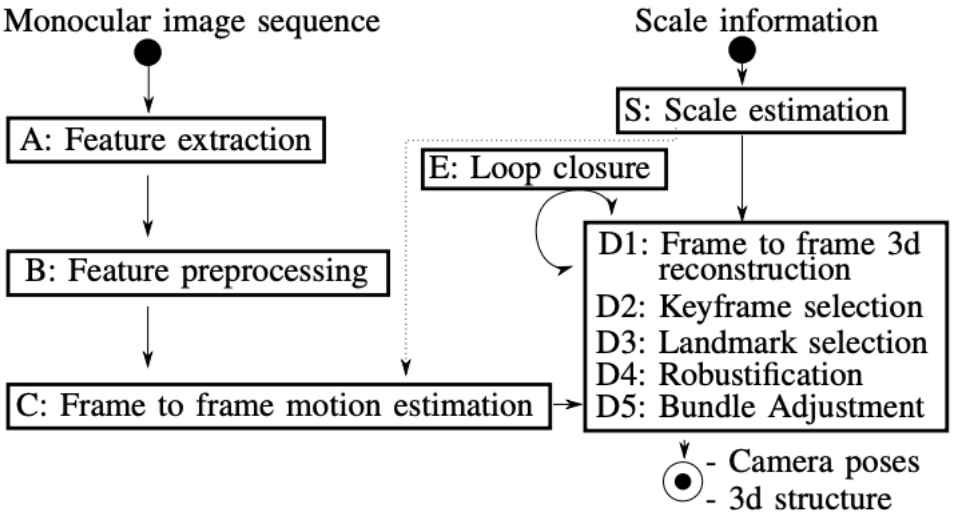

Figure 3. Structure of the VSLAM pipeline.

The camera module of our LiDAR-camera SLAM system performs several vital functions. Firstly, it engages in Feature Extraction, where the camera identifies crucial feature points, such as corners or feature descriptors like SIFT and SURF, within the image frames. These feature points are subsequently used in the matching and tracking processes. Secondly, it undergoes Feature Preprocessing, which includes noise reduction and image distortion correction on the extracted feature points, enhancing the accuracy and stability of subsequent matching. Thirdly, the module executes Frame to Frame Motion Estimation, estimating the camera's movement by comparing feature points between consecutive frames, typically employing techniques such as optical flow or other motion estimation algorithms.

Within the LIDAR module, referred to as LIDAR1, it plays a pivotal role in Scale Estimation. Using LIDAR data, it estimates the scale information within the scene, a critical factor for SLAM, as LIDAR provides accurate distance information aiding in the determination of the camera's absolute position in space.

Additionally, common steps are carried out collaboratively: step 1 involves Frame to Frame 3D, which entails estimating the camera's 3D motion, often in conjunction with frame-to-frame motion estimation from the camera module to obtain the global camera pose. Step 3 is Landmark Selection, selecting a stable set of landmarks for subsequent pose estimation and map construction. Step 2,

DOI: 10.54254/2755-2721/41/20230755

Keyframe Selection, improves SLAM system efficiency by identifying and using keyframes for map updates and loop closure detection. Step 4, Robustification, enhances system robustness against sensor noise and exceptional situations. Lastly, step 5 is Bundle Adjustment, optimizing camera poses and landmark estimates, reducing errors and enhancing system consistency after the SLAM process is complete.

Finally, loop closure detection is emphasized as a critical component ensuring map consistency and accuracy. It identifies previously visited landmarks or keyframes, correcting pose estimates and facilitating accurate and robust simultaneous localization and mapping in dynamic environments.

RGBD-Camera + Lidar SLAM is a robust solution tailored for low-light nighttime conditions, where natural lighting may be limited. This method adapts to such scenarios by combining the strengths of RGBD cameras and lidar sensors to achieve accurate mapping and localization.

In terms of characteristics, this approach excels in low-light conditions, making it a suitable choice for nighttime environments with minimal natural lighting. It leverages sensor fusion, harnessing the capabilities of RGBD cameras to capture visual data and lidar sensors to generate precise 3D point clouds. The fusion of these two sensor modalities results in enhanced 3D mapping capabilities.

The working principle of RGBD-Camera + Lidar SLAM involves the simultaneous collection of data from RGBD cameras and lidar sensors. RGBD cameras capture high-resolution RGB images, from which visual features like corners and edges are extracted. Lidar sensors, on the other hand, provide depth information by measuring the time taken for laser pulses to reflect off objects in the environment, creating a 3D point cloud.

The mapping process relies on feature-based mapping using visual features extracted from RGB images. These features are tracked over time, facilitating the construction of a map that includes the positions of these visual landmarks. What sets this method apart is the fusion of visual features from RGB images with precise depth information from lidar sensors, resulting in a more accurate and detailed map.

Odometry estimation plays a critical role in tracking the robot's motion accurately. Visual odometry estimates motion based on changes in visual features, while lidar-based odometry provides precise 3D movement information. The fusion of visual and lidar-based odometry enhances accuracy, particularly in low-light conditions where visual odometry alone may be less reliable.

In map generation, the accumulated visual features and integrated lidar point cloud create a comprehensive representation of the environment. Loop closure detection techniques enhance map accuracy by identifying revisited areas and ensuring proper alignment with the existing map. The map can be stored in various formats, such as combined point cloud and feature maps or graph structures, depending on specific application requirements.

To initialize the Bundle Adjustment (BA) process, we rely on a frame-to-frame motion estimate. This initialization step plays a crucial role in improving computation efficiency and accuracy in landmark selection for the subsequent BA optimization. Our frame-to-frame motion estimation builds upon the Perspective-n-Point (PnP) problem, a well-established technique in computer vision.

The objective of the PnP problem is to minimize a cost function, represented as:

$$\underset{x,y,z,\alpha,\beta,\gamma}{\operatorname{argmin}} \sum_{i} \|\varphi_{i,3d\to 2d}\|_2^2$$

$$\varphi_{i,3d\to 2d} = \bar{p}_i - \pi(p_i, P(x,y,z,\alpha,\beta,\gamma)) \tag{4}$$

Here, ϕ3d→2d represents the 3D-to-2D projection error, where p¯i is the measured feature point in the current frame, and pi is the corresponding 3D point corresponding to p¯i. The transformation from the previous to the current frame is described by six parameters: three for translation (x, y, z) and three for rotation (α, β, γ). The function π(. . .) handles the projection from 3D world coordinates to the 2D image domain. To estimate pi, we use feature correspondences p˜i from the previous frame along with their estimated depths.

DOI: 10.54254/2755-2721/41/20230755

While the PnP method excels in urban scenarios with rich structural information, it faces challenges in scenarios with low structure and high optical flow, such as highway environments. In such cases, the number of features with valid depth information may be insufficient for accurate estimation.

To enhance the stability of our algorithm, we introduce the epipolar error ϕ2d→2d:

$$\varphi_{i,2d\to 2d} = \bar{p}_i F\left(\frac{x}{z}, \frac{y}{z}, \alpha, \beta, \gamma\right)\tilde{p}_i \tag{5}$$

This error term incorporates the fundamental matrix F, which can be derived from the frame-to-frame motion and the intrinsic camera calibration parameters. These additional constraints contribute to improved accuracy in rotation estimation.

Here, a(s) is a fixed threshold for outlier rejection. This loss function helps reduce the influence of outliers, contributing to more robust estimation. For a more in-depth understanding of the fundamental matrix and the influence of the loss function, we refer to the work of Hartley et al. [3].

The final optimization problem for the frame-to-frame motion estimate can be formulated as:

$$\underset{x,y,z,\alpha,\beta,\gamma}{\operatorname{argmin}} \sum_i \rho_{3d\to 2d}\left(\|\varphi_{i,3d\to 2d}\|_2^2\right) + \rho_{2d\to 2d}\left(\|\varphi_{i,2d\to 2d}\|_2^2\right) \tag{6}$$

3.4. Radar SLAM for Rainy and Foggy Weather

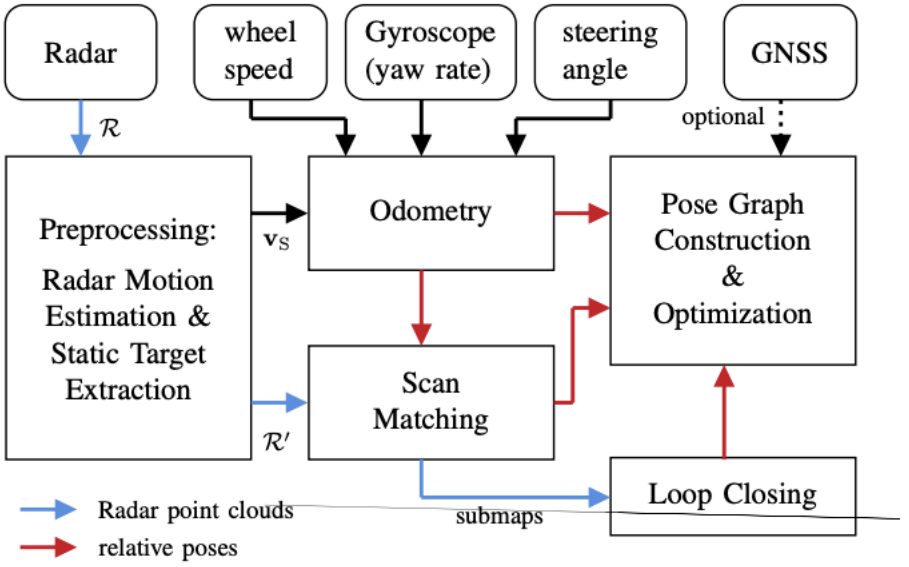

Figure 4. Components of the Radar SLAM system.

The integration of Radar sensors into the proposed SLAM system represents a significant advancement in the field of simultaneous localization and mapping. This system is equipped with several key components, each contributing to its robust performance in various aspects.

The Radar Preprocessing Component serves as the initial stage, responsible for handling the raw data obtained from the Radar sensors. It performs a crucial role by distinguishing between static targets, such as stationary objects in the environment, and moving targets, which include objects in motion. Furthermore, it effectively filters out clutter or unwanted noise present in the Radar scans. Additionally, this component estimates the velocity of the Radar sensor itself, providing essential data for motion estimation and precise pose tracking of the robot.

The Odometry Component is another vital element of the system, dedicated to estimating the relative transformations between consecutive robot poses, encompassing both position and orientation. To achieve this, it amalgamates data from a variety of sensors, including wheel speed measurements,

DOI: 10.54254/2755-2721/41/20230755

steering wheel angles, yaw rate measurements, and velocity information derived from the Radar sensor. These relative transformations play a pivotal role in monitoring the robot's movement within its surrounding environment, offering invaluable insights for localization and mapping.

Central to the system's functionality is the Scan Matching Component, which utilizes the renowned Iterative Closest Point (ICP) method. This component takes on the responsibility of aligning sequential Radar scans, a critical step in the SLAM process. By employing the ICP method, which excels at registering point clouds by determining the optimal transformation to align them, the system effectively estimates the relative pose changes, encompassing both position and orientation, between successive scans. This information serves as the foundation for constructing a consistent and accurate map of the environment.

The Loop Closing Component adds a layer of sophistication to the system by recognizing instances where the robot revisits previously explored locations. When a loop closure is detected, this component goes to work, calculating the relative transformations between the current robot position and those of its previous visits. These relative pose estimates are instrumental in rectifying any accumulated errors in the robot's trajectory, ensuring a more accurate representation of its path through the environment.

In the grand scheme of things, the system operates on the principles of graph optimization. It assembles a pose graph utilizing the relative pose estimates derived from both the scan matching and loop closing components. Graph optimization techniques are then applied, effectively determining the most likely trajectory of the robot that best aligns with the collected data. In this process, a point cloud map is generated, composed of Radar detections, providing a comprehensive and detailed portrayal of the environment's features and obstacles.

To further enhance its capabilities, the proposed SLAM system offers optional GNSS (Global Navigation Satellite System) integration. This allows for the incorporation of position measurements obtained from a GNSS receiver. By doing so, the system gains the advantage of improved localization accuracy, particularly in outdoor environments where GNSS data can serve as a valuable constraint for the SLAM solution.

In conclusion, this advanced SLAM system represents a pioneering approach that capitalizes on the strengths of Radar sensors while effectively addressing the challenges associated with Radar-based SLAM. Its ability to directly match point clouds using the ICP method and seamlessly integrate data from multiple sensor modalities positions it as a powerful solution for accurate localization and mapping. Moreover, the adoption of a graph-based SLAM approach aligns with industry standards, offering superior performance and user-friendly characteristics compared to traditional filtering-based methods.

Radar SLAM is a robust solution tailored for challenging weather conditions, including scenarios marked by rain, fog, or low visibility due to darkness. It distinguishes itself through a set of defining characteristics and a unique working principle that enables effective mapping and navigation in adverse environments.

One of the standout characteristics of Radar SLAM is its exceptional weather resilience. Unlike traditional sensor modalities such as cameras or lidar, Radar SLAM thrives in conditions where visibility is severely limited, making it a reliable choice when environmental challenges are at their peak. Its ability to operate effectively in low-light or inclement weather scenarios ensures that critical mapping and localization tasks can continue without interruption.

A fundamental feature of Radar SLAM is its independence from visual cues. While many other SLAM methods heavily rely on visual information, which can be obstructed or distorted by adverse weather conditions, Radar SLAM remains unfazed. Its operation is based on radar sensors, which emit electromagnetic waves in the microwave frequency range. These waves penetrate through rain, fog, or darkness, providing consistent data acquisition even when other sensors might falter.

The working principle of Radar SLAM revolves around the ingenious use of radar sensors to perceive the environment and navigate in conditions where visibility is severely challenged. These radar sensors emit electromagnetic waves into the environment and measure the time it takes for these waves to bounce back after interacting with objects in the surroundings. This time delay is used to calculate the distance to objects, resulting in a point cloud representation of the environment. Furthermore, radar

DOI: 10.54254/2755-2721/41/20230755

sensors can detect the Doppler effect, which provides valuable velocity information about objects. This additional layer of data is crucial for motion estimation and odometry.

The map generated by Radar SLAM is a direct product of the radar sensor's perception of the environment. It consists of objects detected by the radar sensor, with each object represented as a point in three-dimensional space. As the radar sensor continuously detects objects within its field of view, these objects are mapped into the robot's coordinate frame. This dynamic mapping process ensures that the map remains synchronized with the real-world environment, allowing the robot to navigate effectively, even in challenging conditions.

To enhance map accuracy, some Radar SLAM systems implement loop closure detection. This involves recognizing when the robot revisits a previously explored area and correctly aligning the new radar data with the existing map. This mechanism ensures that the map remains consistent and accurate over time, even in scenarios where the robot's path may intersect.

Radar SLAM also relies on radar-based odometry techniques to estimate the robot's motion and position accurately. This odometry process leverages the Doppler effect to measure the velocity of objects in the environment. By analyzing changes in Doppler measurements as the robot moves, the system can estimate its own velocity and, consequently, its motion. Additionally, object tracking plays a crucial role in radar-based odometry. The positions of objects detected by the radar sensor are tracked over time, and this tracking information contributes to accurate motion estimation.[5]

In some instances, Radar SLAM systems incorporate data fusion techniques to combine radar-based odometry with information from other sensors, such as wheel encoders or inertial sensors. This fusion enhances the accuracy of motion estimation, especially in complex scenarios where multiple sources of information can provide a more comprehensive understanding of the robot's movement.

In conclusion, Radar SLAM shines as a formidable solution for mapping and localization in adverse weather conditions. Its capacity to operate independently of visual cues, coupled with its robust weather resilience, positions it as a valuable tool for scenarios where other sensors may struggle to provide reliable data. Its working principles, map generation strategies, and odometry techniques make it a reliable choice when environmental challenges are at their peak.[6]

The foundation of radar scan matching lies in the ICP algorithm, which operates as follows. Given initial guesses for translation (t0) and rotation (θ0) and two sets of points (P and Q), ICP iteratively refines these estimates to align P with Q. Each iteration transforms P according to the current θk and tk, producing a transformed point set P0k. This transformation involves applying a rotation matrix, Rk, to each point in P, allowing for translation and rotation adjustments to minimize the sum of squared Euclidean distances between point pairs {p0i, qj}. The algorithm continues these iterations until convergence is reached.

$$p'_k = \left\{p'_1, \cdots, p'_n\right\} \tag{7}$$

$$\text{with } p'_i = \mathrm{R}_{k\mathrm{P}_i} + \mathrm{t}_k, \quad \mathrm{R}_k = \begin{bmatrix} \cos\theta_k & -\sin\theta_k \\ \sin\theta_k & \cos\theta_k \end{bmatrix} \tag{8}$$

Radar scans pose unique challenges compared to Lidar sensors with high angular resolution and thousands of points per scan. Radar scans often consist of fewer than 100 detections, especially after removing clutter and accounting for moving targets. Attempting to match consecutive radar scans directly can yield suboptimal results. Instead, the scan matcher adopts a more effective approach.[4]

The scan matcher divides the radar scans into small partial maps, referred to as submaps. These submaps are created by merging several consecutive radar scans. To ensure consistency, the positions of static targets in each radar scan (R0) are transformed into a common Cartesian coordinate system using odometry pose estimates and known sensor position information. Once a submap accumulates a predefined number of scans (N, a design parameter), it is considered complete.

DOI: 10.54254/2755-2721/41/20230755

To further enhance data quality, a radius outlier filter is employed to reduce measurement noise. This filter removes points with fewer than a predefined number of neighbors within a specific radius, thus improving the robustness of subsequent processing.[4]

$$\underset{t,\theta}{\operatorname{argmin}} \sum_{(i,j)} \|\mathrm{R}(\theta)p_i + t - q_j\|^2 \tag{9}$$

After submap formation and noise reduction, the scan matcher proceeds to align the submap with a history of H previous submaps using the ICP algorithm. The odometry component provides initial values for translation and rotation, aiding in convergence. However, it's important to note that only point pairs with distances below a 2-meter threshold are considered in the ICP objective function. This constraint helps maintain the quality of the alignment.

$$\mathrm{Var}(\Phi^*) \approx \frac{\mathcal{J}(\Phi^*)}{\mathrm{n}-3}\left[\frac{1}{2}\mathrm{H}\right]_{\Phi}^{-1} \tag{10}$$

To assess the reliability of the resulting relative pose estimates, the scan matcher employs the Hessian method. This method provides an estimation of uncertainty (variance) in the relative pose (Φ*). The uncertainty estimation, denoted as Var(Φ*), is proportional to the Hessian of the ICP objective function (J) with respect to the pose parameters (Φ). It allows for a quantified measure of confidence in the estimated transformation.[3]

4. Implementation

4.1. Equipment

The robotic platform implemented is characterized by its compact dimensions of 484 mm in length, 378 mm in width, and 189 mm in height. this versatile robotic platform embodies a four-wheeled configuration with Mecanum wheels. The Mecanum wheels not only enable agile and omnidirectional movement but also enhance the platform's maneuverability in tight spaces and complex environments with a maximum achievable speed of 2 m/s.

Figure 5. The robotic platform implemented.

DOI: 10.54254/2755-2721/41/20230755

Considering the requirements for a mature technology, a large field of view, and low energy consumption for an RGBD camera, we have selected the Intel® RealSense™ Depth Camera D435i as the ideal choice. The Intel® RealSense™ Depth Camera D435i is a versatile imaging and sensing solution suitable for both indoor and outdoor use. With an ideal depth range spanning from 0.3 to 3 meters and highly accurate stereoscopic depth technology, it delivers precise spatial information. The camera captures high-quality RGB frames at a resolution of 1920 × 1080 pixels and offers a broad field of view at 69° × 42° with its rolling shutter RGB sensor technology. Additionally, it comes with a flexible mounting mechanism, featuring one 1/4-20 UNC thread mounting point and two M3 thread mounting points, making it adaptable to various setups. Furthermore, the integration of an Inertial Measurement Unit (IMU), specifically the BMI055 IMU, enhances its motion tracking capabilities.

Figure 6. The Intel® RealSense™ Depth Camera D435i.

The choice of the Mid-360 LiDAR by Livox is driven by specific requirements for your application, including a relatively large sensing range, high sampling resolution, and low energy consumption. It uses a 950nm laser and is designed for low-speed robotics. This LiDAR module provides complete 360° 3D perception. Mid-360 is particularly powerful in object detection. Boasting a 360° * 59° field of view, a minimum detection range of 0.1 meters, and 40 meters range at 10% reflectivity, along with a dense 40-line point cloud, this LiDAR module measures just 65 x 65 x 60 mm (L × W × H) and weighs a mere 265 grams.

Figure 7. The Mid-360 LiDAR.

In selecting the ARS408-21 radar, key factors considered were its extensive detection range, impressive resolution, and robust data processing capabilities. This radar excels in applications like automotive safety and surveillance due to its unmatched precision and reliability.

This ARS408-21 radar boasts an impressive range, from 0.20 to 250 meters in the far range, with near range capabilities of 0.20 to 70 meters (extending to 100 meters at 0°) and 0.20 to 20 meters at ±60° azimuth. It offers exceptional precision with distance measurements accurate to ±0.40 meters in the far range and ±0.10 meters in the near range. Its azimuth angle augmentation spans from -9.0° to +9.0° in the far range and -60° to +60° in the near range, while elevation angle augmentation is 14° in the far range and 20° in the near range. With a fast cycle time of approximately 72 ms for near and far measurements, 24 antenna channels, and advanced digital beam forming, the ARS408-21 radar is a top-

DOI: 10.54254/2755-2721/41/20230755

tier solution for applications ranging from automotive safety to surveillance, offering unmatched precision and reliability.

Figure 8. The ARS408-21 radar.

The NUC11PAHi7, Powered by the **Intel® Core™ i7-1165G7 Processor with up to 4.70 GHz** of processing power, this compact kit offers remarkable performance in a small package. With features like **64GB** memory. **measuring 117 x 112 x 51mm**, makes it suitable for space-constrained environments**.**

Figure 9. The NUC11PAHi7.

All the above hardware connections are shown in the figure 10.

DOI: 10.54254/2755-2721/41/20230755

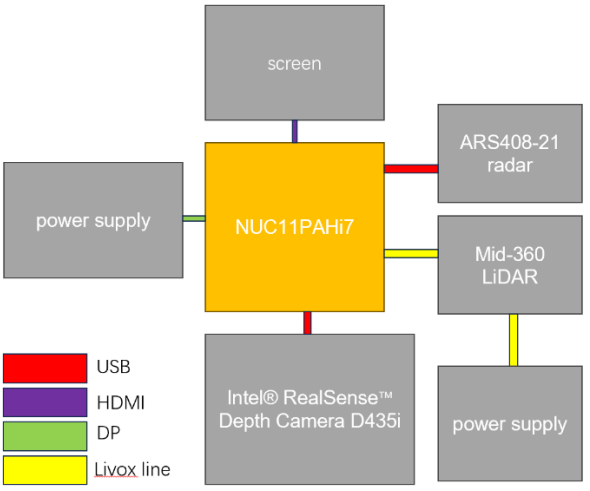

Figure 10. Hardware connection diagram.

4.2. RGB-D Camera SLAM for Sunny and Daytime Conditions

The RGB-D Camera SLAM method utilized in our weather-adaptive SLAM system is based on the "RGB-D Hand-Held Mapping" tutorial provided by the RTAB-Map ROS package[7].

The core algorithm driving the RGB-D Camera SLAM in this method is RTAB-Map, a real-time appearance-based mapping system for RGB-D cameras. RTAB-Map combines visual and depth data to create 3D maps while also performing real-time localization of the camera within these maps.

Here are the main steps:

Step 1: Create a ROS Workspace

Firstly, we need to create a ROS workspace if we haven't already. This workspace serves as a contained directory where all the necessary packages and dependencies for the project will be organized. The ROS workspace is typically created using the following command:

```
mkdir -p ~/catkin_ws/src
cd ~/catkin_ws/
catkin_make
```

This command structure establishes the workspace, including the source directory, and initializes the workspace with catkin_make.

Step 2: Install RTAB-Map ROS

To enable SLAM mapping, the RTAB-Map ROS package must be installed. RTAB-Map is a key component for SLAM and offers ROS integration for ease of use. The installation process involves several commands, as follows:

- Add the RTAB-Map ROS repository to the ROS sources list:

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
```

- Set up the ROS keys:

```
sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-key C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
```

- Update the package list:

```
sudo apt-get update
```

- Install RTAB-Map ROS:

```
sudo apt-get install ros-noetic-rtabmap-ros
```

Step 3: Prepare Sensor Data

DOI: 10.54254/2755-2721/41/20230755

Prior to initiating SLAM mapping, it is crucial to ensure that the sensor data is adequately prepared. In this RGB-D camera SLAM method, the sensory input is derived from the Camera, which provides both depth and color images. Preparation of sensor data may include:

- Verifying the proper connection of the camera to the system.
- Confirming the availability of depth and color image streams.
- Ensuring that the camera calibration data is accurate and up-to-date.

These preparations ensure that the sensor data is reliable and ready for use in the SLAM mapping process.

Step 4: Create a Launch File

A ROS launch file is a crucial component for configuring the SLAM mapping system. This launch file specifies various parameters, including sensor settings and the SLAM algorithm to be employed. Creating a custom launch file tailored to the specific requirements of the RGB-D camera SLAM method is essential. It typically includes:

- Setting sensor parameters such as camera topics, image resolution, and frame rates.
- Defining the SLAM algorithm to be used, such as RTAB-Map.
- Configuring other nodes and parameters related to SLAM mapping.

The launch file ensures that the SLAM mapping system is initialized with the correct settings for optimal performance.

Step 5: StepRun SLAM Node

Executing the ROS launch file launches the SLAM node, which is responsible for processing sensor data and performing SLAM mapping. The SLAM node applies the configured SLAM algorithm to the incoming data streams, generating mapping results in real-time.

Step 6: Save Maps

During the SLAM mapping process, the system provides the capability to save generated map data. This functionality is accessible through specific key commands that can be issued during runtime. For instance, pressing a designated key can trigger the system to save the current map or certain map data for later analysis or reference.

RTAB-Map excels in real-time map creation, allowing the system to continuously update and refine the map as the robot or vehicle navigates its environment. This is particularly valuable for dynamic scenarios.

4.3. Lidar and RGB-D Camera Fusion SLAM for Clear Nighttime Conditions

The second method employed in our weather-adaptive SLAM system is derived from the "LiMo: Lidar and Monocular Vision Fusion SLAM" package, developed by Johannes Graeter. This innovative method addresses a critical need by harnessing the combined power of camera and LIDAR sensors. It introduces a novel depth extraction algorithm from LIDAR data, enabling robust visual localization through keyframe-based Bundle Adjustment. By incorporating semantic labeling for outlier rejection and emphasizing vegetation landmarks, the method enhances accuracy and reliability.

Step 1: Install Ceres Solver

Before using LiMo, we need to install Ceres Solver, which is a library for solving large, nonlinear optimization problems. Ceres Solver is a prerequisite for LiMo. we can follow the installation instructions for Ceres Solver based on our system's requirements. Make sure to install the required dependencies as well.

Step 2: Install LiMo

Once Ceres Solver is installed successfully, we can proceed to install LiMo. LiMo is the package responsible for LiDAR and monocular camera fusion SLAM. we can clone the LiMo repository from its GitHub page at https://github.com/johannes-graeter/limo.

To install LiMo, follow these general steps:

- Clone the LiMo repository to ourr local machine using git clone https://github.com/johannes-graeter/limo.git.
- Navigate to the LiMo directory using the cd limo command.

DOI: 10.54254/2755-2721/41/20230755

- Build LiMo by running catkin_make or catkin build. This step compiles the LiMo package and its dependencies.

Step 3: Compile the Workspace

After successfully installing LiMo, we should compile our ROS workspace to ensure that all packages, including LiMo, are built and ready for use.

Step 4: Run LiMo

Now that LiMo is installed and our workspace is compiled, we can run LiMo for SLAM mapping. To do this, we'll need to create launch files and configure LiMo parameters based on our specific hardware setup and mapping requirements. These launch files will specify sensor parameters, LiMo algorithm settings, and other parameters necessary for SLAM mapping. Once the launch files are set up, we can run LiMo by executing the launch files using the roslaunch command.

4.4. Radar SLAM for Rainy and Foggy Weather

The method is known as Navtech Radar SLAM and it is developed and maintained by Giseop Kim. The radar odometry component is based on the Yeti open source project, incorporating motion distortion considerations and enabling seamless connection with the later place recognition module through ROS. The radar place recognition module utilizes a Cartesian 2D feature point cloud extracted from the cen2019 method and applies it to the Scan Context method for loop detection. Lastly, pose-graph optimization is implemented using iSAM2 in GTSAM, enhancing the overall SLAM performance.

Step 1: Install GTSAM (Graphical Models Toolkit)

GTSAM is a crucial library for implementing SLAM algorithms. To install it, you can follow these detailed steps:

- Visit the GTSAM website: GTSAM.
- Navigate to the "Get Started" or "Installation" section of the website.
- Choose the installation method that corresponds to your operating system and package manager. Typically, GTSAM can be installed using CMake, Homebrew, or other package managers. Follow the provided installation instructions for your chosen method.

Step 2: Install OpenCV

OpenCV is an essential computer vision library for image processing. You can install it with the following steps:

- Visit the OpenCV website: OpenCV.
- Go to the "Download" or "Installation Guide" section.
- Choose the installation method suitable for your operating system. Common methods include using package managers like apt for Ubuntu or brew for macOS, or building from source. Follow the provided instructions for your chosen method.

Step 3: Install Navtech Radar SLAM

Clone the Navtech Radar SLAM repository to your local machine using Git

- Clone the Navtech Radar SLAM repository to your local machine using Git
- Navigate to the cloned directory

5. Experiment

We conducted three groups of related experiments, and the related results are as follows:

First, nodes of the graph generated by RTAB-Map after 1 rounds of global loop closure,the results are as follows:

DOI: 10.54254/2755-2721/41/20230755

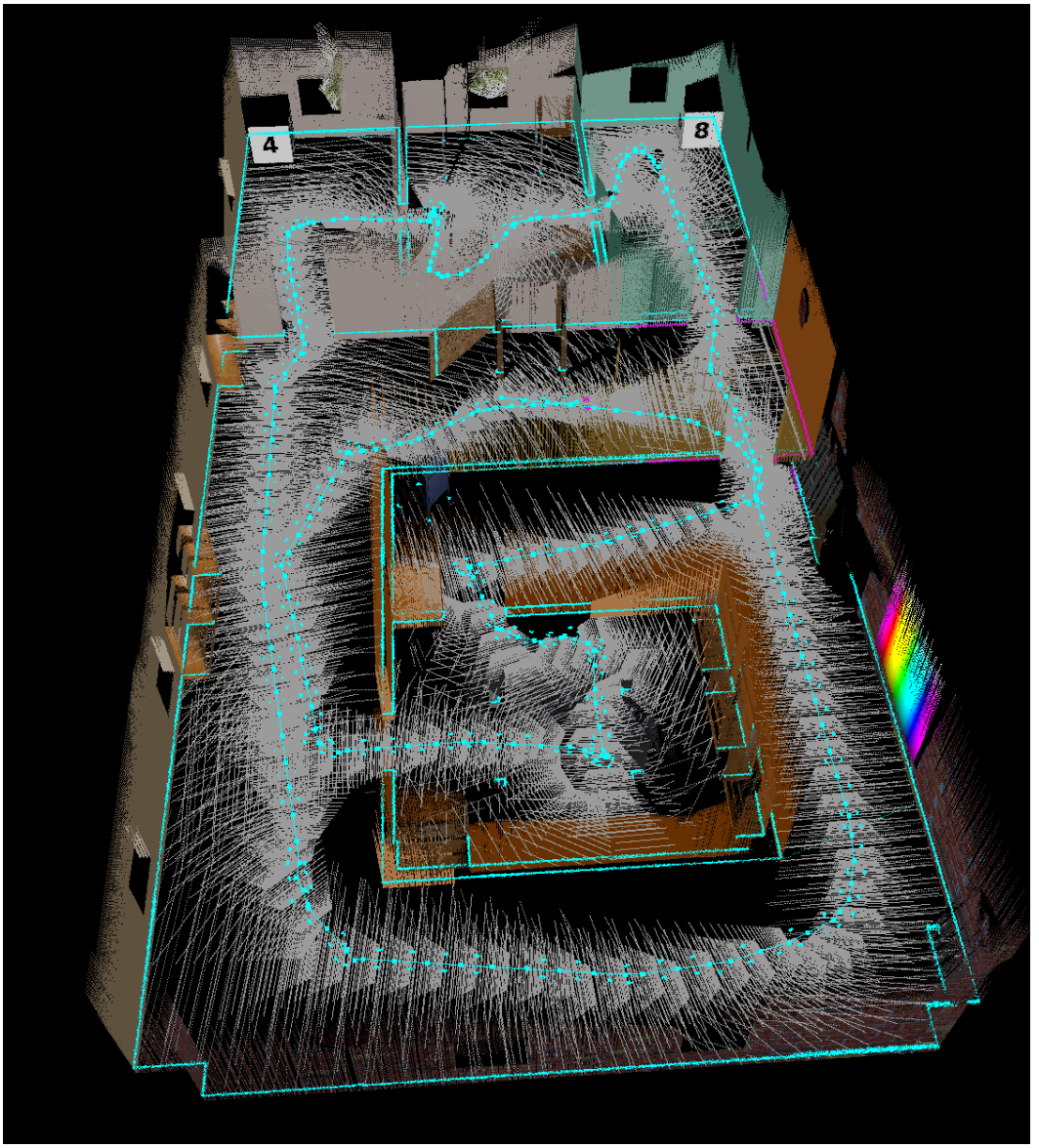

Figure 11. Camera simulation image generated by RTAB-Map algorithm.

Second, the occupancy grid generated by RTAB-Map after 1 rounds of global loop closure,the results are as follows:

DOI: 10.54254/2755-2721/41/20230755

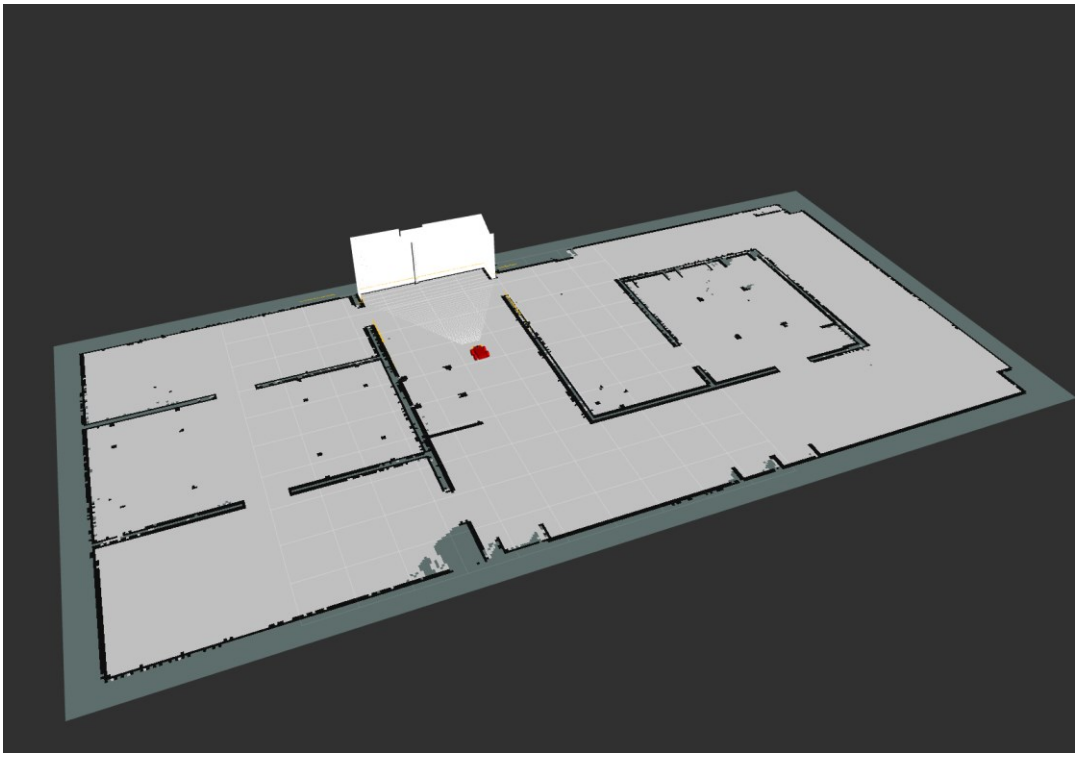

Figure 12. Simulation scenario.

Depth estimation using LIDAR in real environments, the blue line is LIMO estimate, The blue line is LIMO estimate, thick - inside optimization window, thin - outside optimization window. The red line is Ground Truth. Landmarks: Black - not selected for optimization, Yellow - with LIDAR depth estimate, Green - moni camera only, Bright color - near bin, Medium bright color - middle bin, Dark color - far bin, the results are as follows:

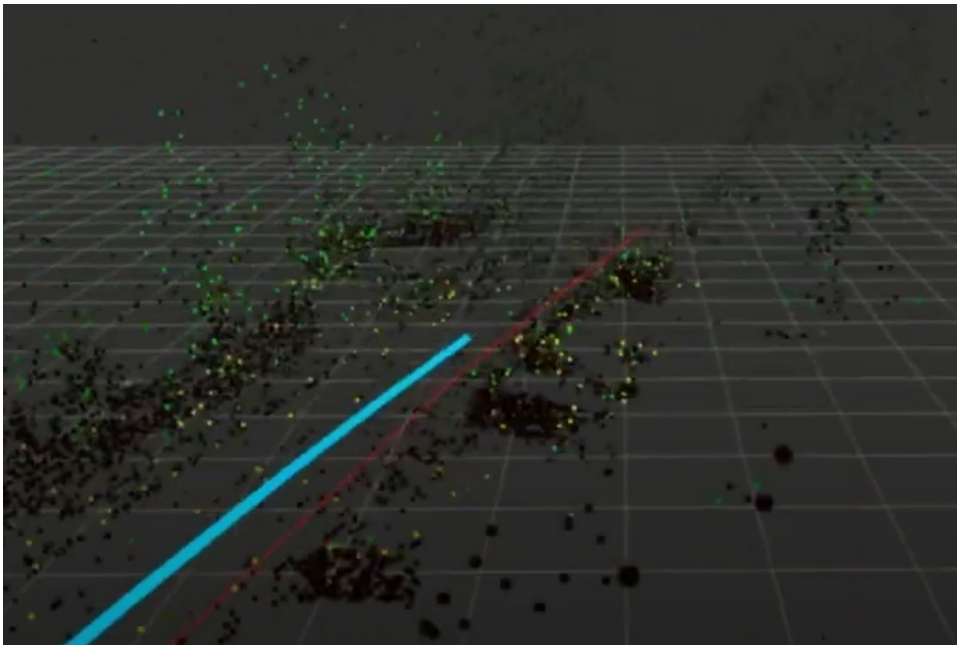

Figure 13. A simulated image generated using the LiMo algorithm, utilizing data from both a camera and a lidar sensor.

DOI: 10.54254/2755-2721/41/20230755

Testing using the MulRan dataset, which is suitable to evaluate the radar odometry or SLAM algorithm in complex urban sites. The results of odometry and loop closing are as follows.

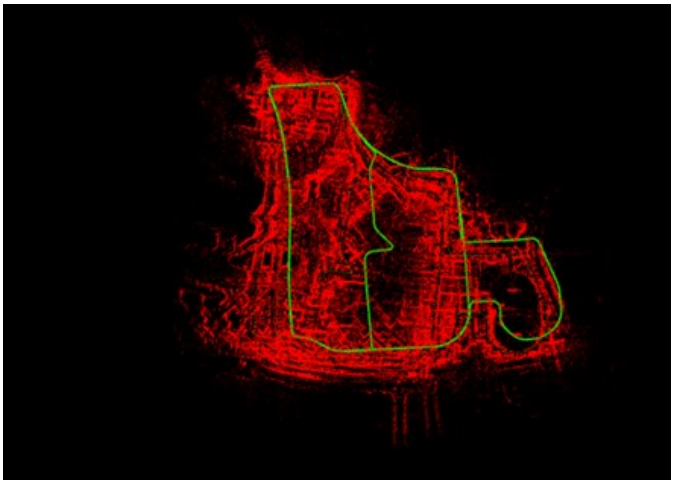

Figure 14. A simulated scene generated from radar data.

6. Conclusion

Our weather-adaptive SLAM system, incorporating RGBD cameras, Lidar, and radar, dynamically selects the optimal sensor combination based on real-time weather data, ensuring precise mapping and localization in diverse conditions. This research highlights the importance of adaptable SLAM solutions, offering a robust framework for enhancing the reliability and accuracy of autonomous vehicles and robotic systems across varying weather scenarios, thus advancing the field of robotics and autonomous navigation.

In terms of future work, we plan to test our method in real track scenarios. Also, we want to explore further methods that are resilient to changing weathers.

References

[1] Das, Sagarnil. "Simultaneous localization and mapping (SLAM) using RTAB-MAP." *arXiv preprint arXiv:1809.02989* (2018).

[2] M. Holder, S. Hellwig and H. Winner, "Real-Time Pose Graph SLAM based on Radar," 2019 IEEE Intelligent Vehicles Symposium (IV), Paris, France, 2019, pp. 1145-1151, doi: 10.1109/IVS.2019.8813841.

[3] Hong, Ziyang, et al. "Radar SLAM: A robust SLAM system for all weather conditions." *arXiv preprint arXiv:2104.05347* (2021).

[4] "Rtabmap_ros - ROS Wiki." *Wiki.ros.org*, wiki.ros.org/rtabmap_ros.

[5] Graeter, Johannes, Alexander Wilczynski, and Martin Lauer. "Limo: Lidar-monocular visual odometry." *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2018.

[6] Grisetti, Giorgio, et al. "A tutorial on graph-based SLAM." *IEEE Intelligent Transportation Systems Magazine* 2.4 (2010): 31-43.

[7] "Intel® NUC 11 Performance Kit - NUC11PAHi7 - Product Specifications." *Intel*, www.intel.com/content/www/us/en/products/sku/205073/intel-nuc-11-performance-kit-nuc11pahi7/specifications.html. Accessed 15 Sept. 2023.

Acknowledgments

Authors wishing to acknowledge assistance or encouragement from colleagues, special work by technical staff or financial support from organizations should do so in an unnumbered Acknowledgments section immediately following the last numbered section of the paper.