Radar SLAM: Towards Robust Indoor SLAM with mmWave Radar

Andrew Wang **Jacopo Ferro** **Abdelmalek Ahmed** **Néstor Lomba**

Abstract

Millimeter wave radars have been increasingly adopted in the automotive sector due to their shorter wavelengths, which provide higher resolution imaging capabilities. In this work, we propose a comprehensive pipeline for Simultaneous Localization and Mapping (SLAM) utilizing Commercial Off-The-Shelf (COTS) components, specifically combining the AWR1843BOOST radar sensor with the DCA1000 data card and the Turtlebot 4 robot.

The mapping process determines the robot's location by leveraging odometry data from the Inertial Measurement Unit (IMU) and wheel encoders. Simultaneously, a custom OctoMap Node efficiently constructs a 2D occupancy map by projecting the 3D point cloud onto the ground plane.

This methodology harnesses the distinctive benefits of radar technology, such as its ability to operate effectively under adverse optical conditions, including fog, rain, dust, and smoke. Furthermore, it can detect transparent materials like glass and acrylic, ensuring reliable performance where other sensors such as cameras and LiDAR may fail. Our system provides a robust and versatile SLAM solution, enhancing environmental perception and navigation in challenging conditions.

1 Introduction

1.1 FMCW Radar Background

Frequency-Modulated Continuous Wave (FMCW) millimeter-wave radar is a sophisticated radar system that transmits linear sweeps of frequency over time, known as chirps. The frequency of the transmitted signal increases or decreases linearly during a period called the chirp duration.

When the transmitted signal encounters an object, the reflection is received back by the radar. The radar then mixes the received signal with the transmitted signal, performs initial signal processing such as amplification and filtering, and finally converts the signal into digital form using an analog-to-digital converter (ADC), resulting in what is typically referred to as *Raw ADC Data*.

The fundamental principle behind FMCW radar involves the constant and finite speed of electromagnetic waves. There is a time delay between the transmission and reception of the signal. Due to the linear frequency sweep of the chirps, this time delay translates into a frequency difference that is directly proportional to the time delay and, consequently, to the distance of the object.

By mixing the transmitted and received signals, a new signal is generated whose frequency (Δf) is the frequency difference between the two. Using the following linear relationship, the distance (R) to the object causing the reflection can be determined:

$$R = \frac{c \cdot \Delta f}{2 \cdot S}$$

where R is the range, c is the speed of light, Δf is the frequency difference of the signal, and S is the slope of the chirp.

If the object is in motion, the reflected signal will also experience a Doppler shift, which affects the frequency of the received signal. By analyzing frequency shifts over multiple chirps, the radar can determine the relative velocity of the object.

The total frequency shift observed in the received signal is a combination of the beat frequency and the Doppler shift. By separating these components, the radar can extract both the range and velocity information of the object.

Additionally, using multiple antennas enables the estimation of the direction of arrival by exploiting the phase differences in the received signals. Beamforming techniques, such as those detailed in (18), allow for the construction of a beam in the direction of interest. This is achieved by summing the received signals with appropriate phase shifts, resulting in constructive interference from the desired direction and destructive interference from other directions. The power output is then computed for every possible angle to determine the object's direction.

Overall, FMCW radar's ability to measure distance, velocity, and direction makes it a powerful tool for various applications, including automotive radar systems, autonomous navigation, and environmental mapping. This technology's robustness in adverse conditions, such as fog, rain, and dust, further enhances its utility in real-world scenarios.

1.2 Project Proposal

The objective of our project was to design and implement a robust Simultaneous Localization and Mapping (SLAM) system utilizing the AWR1843BOOST radar (10) and the Turtlebot 4 robot. Our initial proposal aimed to leverage the advanced capabilities of the AWR1843BOOST radar to generate both odometry and mapping data, guided by methodologies outlined in key research papers such as (3) and (4), along with additional insights from (7), (9), (19), (8) and (15).

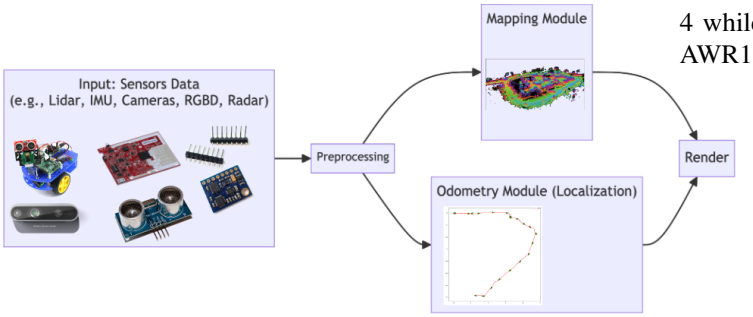

Figure 1: Simultaneous Localization And Mapping (SLAM)

Nonetheless, we quickly realized that this approach required substantial computing power, specifically a powerful GPU, to estimate odometry from the radar sensor accurately. This high computational demand posed significant challenges for real-time processing, which is a critical aspect of our project.

Therefore, our final implementation shifted focus from relying solely on the radar sensor for odometry to utilizing the highly accurate odometry provided by the Turtlebot 4. This decision allowed us to maintain reliable localization while leveraging the radar's strengths in mapping.

Initially, our system involved attaching a physical computer to the top of the Turtlebot 4. However, this setup proved impractical due to space constraints and the cumbersome nature of having both the computer and radar sensor mounted on the robot. Consequently, we transitioned all processing tasks to the Turtlebot 4 itself, which is powered by a Raspberry Pi running Ubuntu.

Leveraging the Robotic Operating System 2 (ROS 2) communication framework, we connected an external computer to the same local network as the Turtlebot 4 to receive the system's outputs. By utilizing the ROS2 driver provided by Texas Instruments, we were able to obtain a continuous 3D Point Cloud from the radar frames.

This Point Cloud data was then processed through a custom map-generating node built on the OctoMap library. Using the odometry data from Turtlebot 4, the system iteratively produced a 2D occupancy map by projecting the 3D Point Cloud onto the ground plane.

Finally, we employed ROS's 3D Robot Visualizer (rviz) to observe the Point Cloud data, robot localization, and the resultant map in an intuitive and interactive manner. This visualization was crucial for debugging and refining our SLAM pipeline, ensuring the system met our initial objectives of robust localization and accurate mapping.

In summary, while our project faced significant challenges in terms of computational requirements and hardware constraints, we adapted by integrating the Turtlebot's odometry with the radar's mapping capabilities, resulting in a versatile and effective SLAM system.

2 System Overview

2.1 Hardware Mount

The general setup for our SLAM system involves using the Turtlebot 4 robot as the mobile platform and mounting the AWR1843BOOST radar on top of it. This configuration allows us to leverage the mobility of the Turtlebot 4 while utilizing the advanced sensing capabilities of the AWR1843BOOST radar.

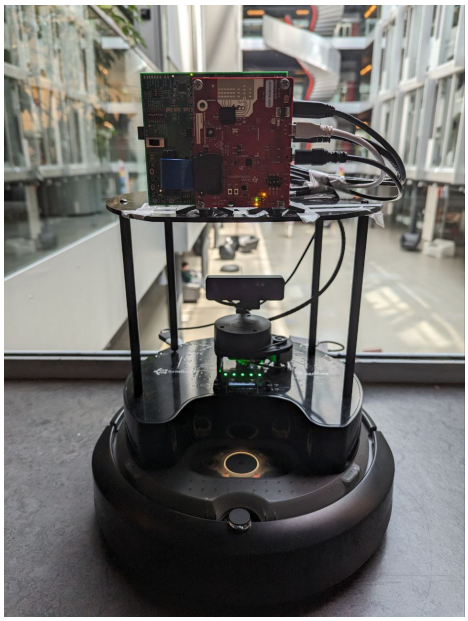

Figure 2: Turtlebot 4 with radar mounted on top

The Turtlebot 4 serves multiple critical functions in this setup. Firstly, it provides a stable and level platform for the radar, ensuring that the data collected is accurate and reliable. Secondly, the Turtlebot 4 supplies power to the radar, eliminating the need for separate power sources and simplifying the overall hardware setup.

In terms of data collection, the Turtlebot 4 interfaces directly with the radar to gather raw data (6). This data is then either broadcasted to a ROS2 topic or stored in a rosbag for later processing (14). Broadcasting the data in real-time to a ROS2 topic allows for immediate processing and visualization while storing the data in a rosbag enables offline analysis and debugging.

The data collected from the radar undergoes a series of processing steps within the pipeline to produce an accurate mapping of the environment. This processing may include noise reduction, data normalization, and feature extraction, depending on the specific requirements of the SLAM algorithm being used. By integrating the radar data with the Turtlebot's odometry information, we can generate detailed and accurate maps that are essential for effective localization and navigation.

The hardware setup, with the radar mounted on the Turtlebot 4, allows us to combine the strengths of both platforms. The Turtlebot's mobility and onboard processing capabilities complement the radar's ability to operate in various environmental conditions. This synergy is crucial for developing a robust and versatile SLAM system that can function reliably in real-world scenarios.

Overall, the integration of the AWR1843BOOST radar with the Turtlebot 4 robot provides a powerful and flexible platform for our SLAM system. This setup not only simplifies the hardware configuration but also enhances the system's capability to perform accurate and reliable mapping and localization.

2.2 Optimizing Radar Parameters

To optimize the performance of the AWR1843BOOST radar for our SLAM system, we configured its parameters to enhance mapping accuracy and efficiency. The horizontal field of view was set to range from -60 to 60 degrees and the vertical field of view from 0 to 30 degrees. These specific angular settings were chosen to reduce the impact of reflections from the ceiling and floor, which are common sources of noise in indoor environments. Additionally, we set the range resolution to 0.15 meters, ensuring high detail and resolution in the captured data.

Beyond these angular adjustments, we also optimized other critical radar parameters to maximize efficiency. The frequency range and sweep time were fine-tuned to balance between capturing detailed information and maintaining real-time performance. The sampling rate and ADC configurations were carefully selected to improve the fidelity of the captured signal, with a higher sampling rate ensuring more accurate data representation. Chirp configuration parameters, including start and end frequencies for multiple chirps, were adjusted to capture a wide range of frequencies.

By tuning these parameters, we effectively minimized unwanted reflections and enhanced the quality of the occupancy maps. This comprehensive optimization process allowed the radar to focus on the most relevant areas of the environment, ensuring that critical features were captured accurately and consistently.

2.3 Offline Pipeline

The offline pipeline we developed is based extensively on the Radarize (17) pipeline, with certain modifications to enable inference without requiring a ground truth (11). The Radarize approach leverages raw data from the radar and, after preprocessing, estimates the radar's velocity using a deep learning model. This methodology allows for both localization and mapping to be performed solely using the radar's raw data, similar to techniques discussed in (1).

The major challenge with this pipeline is the computationally intensive preprocessing of the data, which necessitates significant processing power typically provided by a Nvidia GPU. Unfortunately, we did not have access to such a GPU, which limited our ability to perform real-time model inference.

To collect raw radar data, we utilized the ConnectedSystemsLab/xwr_raw_ros repository (16) (17). This repository records raw radar data and broadcasts it to a ROS1 topic. The recorded data is then stored in a rosbag and sent to a server for preprocessing and model inference. The processed data from the model inference is then combined with heatmap data to produce a map.

In our pipeline, the preprocessing step involves several stages, including noise reduction, data normalization, and feature extraction. These steps are essential for ensuring the raw radar data is in a suitable format for the deep learning model to accurately estimate the radar's velocity and, subsequently, the robot's position.

After preprocessing, the data is fed into the deep learning model, which has been trained to recognize patterns in the radar data that correspond to specific movements and positions. The model outputs the estimated velocity and position of the radar, which is then used to update the robot's position on the map.

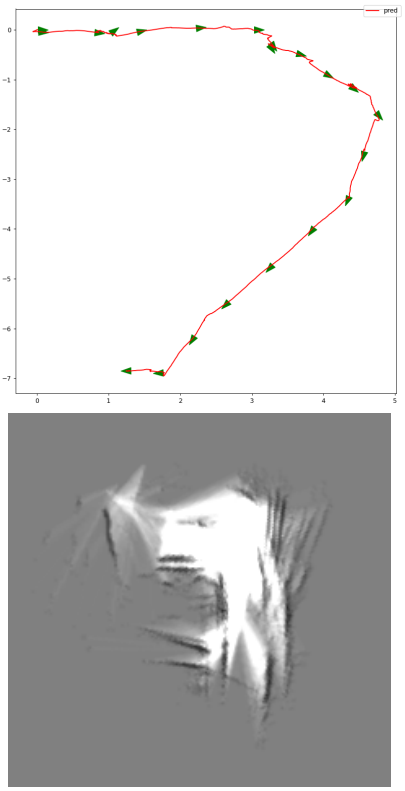

Figure 3: Estimated trajectory and map 1

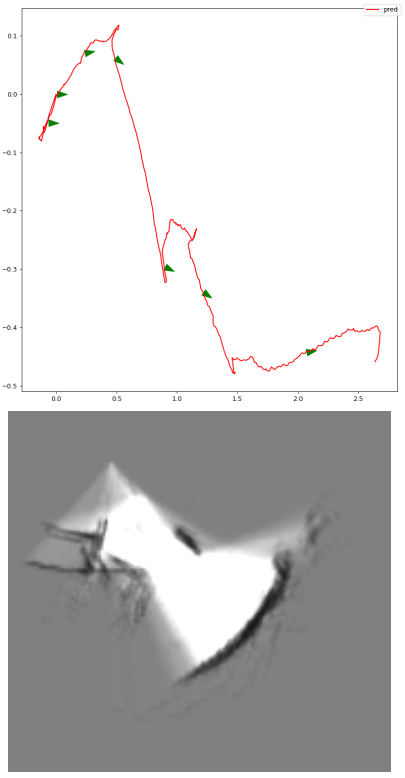

Figure 4: Estimated trajectory and map 2

The major issue, as mentioned earlier, is that the data processing takes too long to feasibly perform in real-time. This is evident in the recording lengths of figures 3 and 4, where intermittently dropped packets from the radar sometimes re-

sulted in errors during the preprocessing step. Consequently, we were unable to record for extended periods, as doing so would increase the risk of encountering a preprocessing error on the rosbag. This dropped packet issue did not appear to affect the team that originally wrote the package, suggesting it might be due to our specific hardware setup, possibly related to the USB Ethernet adapter.

After identifying and addressing the dropped packet issue, we believe it is feasible to make this pipeline run in real-time. However, achieving this will likely require GPU acceleration to handle the intensive preprocessing efficiently.

Future work could involve optimizing the preprocessing algorithms to reduce computational load and exploring alternative methods for data collection and transmission to mitigate the risk of dropped packets. Additionally, integrating GPU support for real-time processing would significantly enhance the system's performance and reliability.

2.4 Online Pipeline

As discussed previously, we built our online pipeline taking into consideration all of the knowledge we acquired so far, favoring simplicity, speed, and ease of implementation.

As seen in 5, our pipeline consists of three main blocks, the first of which is a ROS2 wrapper on the hardware implementation of the 3D pointcloud generation from the raw ADC data. First, the radar board performs a Fast Fourier Transform (FFT) to obtain the range profile from the raw ADC data. Then, it performs a second FFT on the sequence of chirps to extract the Doppler profile. With this 2D range-doppler matrix, the radar board executes a Constant False Alarm Rate filtering, which is a very common method for extracting the signal (real reflections) from the noise using an adaptive threshold. We decided to turn off the additional filtering implemented in the radar board (such as peak grouping) to get a more simple and manageable 3D pointcloud, which is itself just an intermediate step and not the final goal. This output of the radar board is then wrapped in a ROS2 environment thanks to the Texas Instruments ROS2 driver, which is a tool provided by them exactly for this purpose and we just had to make some small changes to the code to make it work as intended. Therefore, the output of this first block is the 3D Pointcloud frames perfectly inserted in the ROS2 framework.

The second block corresponds to the Turtlebot 4, which is a ROS2 node that runs in the Raspberry Pi of the robot and that regulates and synchronizes all the outputs of the robot, from the LiDAR scans to the Battery Status. In our case, we are interested in the odometry information, which although uses many inputs to be calculated, it is mainly the result of two inputs: the Inertial Measurement Unit and the Wheel Encoders. With this setup, the odometry data is extremely precise; even when making the robot do a loop of more than a hundred meters of perimeter, the location is only off by a few centimeters. Given this, we decided that a loop closure algorithm would not be necessary at the scale that we would be working at.

Finally, the Map Generator block takes the input from the other two (3D pointcloud and odometry) and generates a 2D occupancy map by doing a Euclidean projection of the 3D pointcloud and using a probabilistic model to remove possible false reflections. Every time a 3D point is projected on one of the occupancy cells, the probability of that cell being occupied is increased by a given amount. When the occupancy probability grows higher than a given threshold, the cell is set to occupied in the map, represented in the visualizer as a black square. The cells that are in the line of sight of the radar but whose occupancy probability is smaller than the threshold are set to non-occupied or white on the map. Finally, the cells that have not yet been in the line of sight of the radar are set to an unknown status, represented with a greyish-green color on the map. The odometry data is used to establish the position and rotation of the pointcloud in relation to the map, but since our implementation of the Map Generator does not directly access the linear transformation between the map frame and the robot frame given by the Robot itself, we wrote a naive Broadcaster Node that simply takes as input the transformation between the map frame and the robot frame and broadcasts it as the same transformation but between the map frame and the radar frame. Therefore, our implementation assumes that the robot and the radar have the same coordinates, which is very reasonable since one is mounted on top of the other.

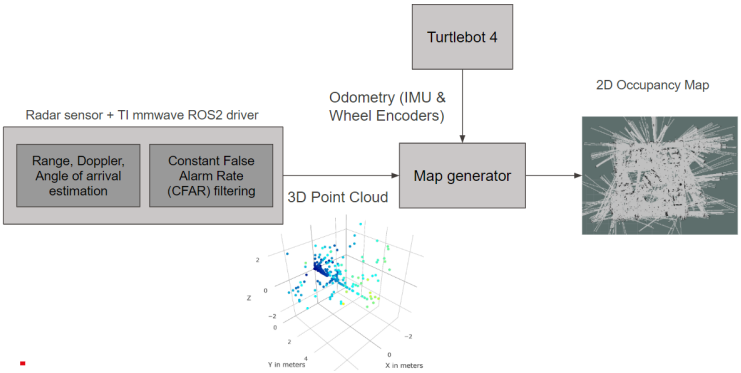

Figure 5: ros2 integration

The pipeline shown in the figure represents the integration of the radar sensor with the Turtlebot 4 to generate a 2D occupancy map. The radar sensor, equipped with the TI mmWave ROS2 driver, performs range, Doppler, and angle of arrival estimations. It also uses Constant False Alarm Rate (CFAR) filtering to enhance the signal processing. The processed data is converted into a 3D point cloud, which is then sent to the map generator along with odometry data from the Turtlebot 4's Inertial Measurement Unit (IMU) and wheel encoders. The map generator uses this data to produce a 2D occupancy map. This map can be visualized with standard tools such as ROS 3D Robot Visualizer (Rviz), providing a clear representation of the environment. By performing our own preprocessing on the raw data, we can achieve greater accuracy in range data, which is crucial for improving the overall SLAM performance (9).

2.5 Results

The results of our online SLAM pipeline demonstrate the effectiveness of integrating the AWR1843BOOST radar sensor with the Turtlebot 4 for real-time mapping and localization. The radar data combined with odometry from the Turtlebot's IMU allowed us to construct detailed occupancy maps of the environment.

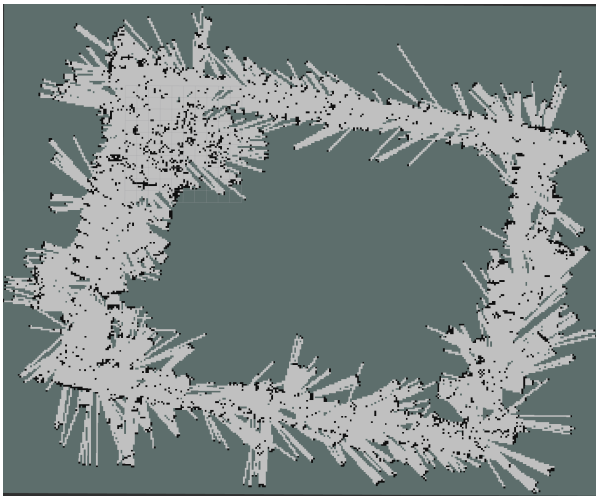

Figure 6: Map of the 1st floor of BC

Figure 6 shows a map of the 1st floor of the BC building. The map was generated using the radar point cloud data processed through a custom OctoMap Node. The map accurately represents the layout and spatial features of the environment including walls, obstacles, and open spaces.

Overall, our results demonstrate the viability of using millimeter wave radar in SLAM applications. The system's ability to produce accurate and reliable maps in real-time underscores the potential of radar technology in enhancing environmental perception.

2.6 Future Work

The development of our SLAM system, while successful, leaves ample room for future enhancements. Here, we outline several promising directions to further improve the accuracy, efficiency, and robustness of our system.

Firstly, the integration of a GPU for real-time processing of raw radar data is a critical enhancement. GPUs are well-suited for parallel processing tasks, and their integration would significantly speed up the complex computations required for radar data preprocessing and SLAM algorithms. This improvement would enable real-time mapping and localization, which is currently limited by the processing power of the onboard CPU.

Secondly, sensor fusion presents a valuable opportunity to enhance our SLAM system. By combining radar data with inputs from other sensors, such as LiDAR and cameras, we can create a more comprehensive and reliable mapping solution. LiDAR provides high-resolution depth information, while cameras offer texture details. Integrating these sensors with radar, which performs well in adverse conditions, can result in a robust SLAM system.

Moreover, exploring advanced filtering techniques and machine learning algorithms can further refine the system's performance in various environmental conditions (19) (5). Techniques such as Kalman filtering and deep learning-based approaches can help in reducing noise and improving the accuracy of the extracted features. Machine learning models, trained on diverse datasets, can learn to identify and correct systematic errors in the radar data, enhancing the overall performance of the SLAM system.

Another area for future work is the optimization of the SLAM algorithm itself. Implementing state-of-the-art SLAM frameworks, such as ORB-SLAM3 or Cartographer, can provide better accuracy and efficiency. Additionally, optimizing the existing OctoMap-based mapping node to handle larger environments and more complex scenes would be beneficial.

Furthermore, the system can be extended to support dynamic environments. Currently, our SLAM system is optimized for static scenes, but real-world applications often involve dynamic elements. Implementing dynamic SLAM techniques that can track and map moving objects in real-time would significantly broaden the applicability of our system.

Lastly, enhancing the user interface and visualization tools can improve the usability and interpretability of the SLAM system. Integrating advanced visualization tools, such as 3D reconstruction and augmented reality interfaces, can provide users with a more intuitive understanding of the mapped environment.

In conclusion, while our current SLAM system demonstrates the feasibility of using radar for mapping and localization, these proposed enhancements can significantly advance its performance, making it more versatile and capable of handling a wider range of applications.

3 Challenges

Throughout the development and implementation of our radar-based SLAM system, we encountered a myriad of challenges. These challenges can be broadly categorized into hardware-related issues, software integration difficulties, and knowledge gaps.

3.1 Hardware Challenges

One of the significant hardware challenges we faced was the inconsistency and occasional malfunctions of the AWR1843BOOST radar. The radar would randomly stop working, which disrupted our testing and data collection processes. This inconsistency was primarily due to the poor support for non-point cloud data in ROS2 (12), making it difficult to fully utilize the radar's capabilities.

As a result, for most of the project, we were reliant on the TI mmWave Studio Windows application to capture radar data. This approach posed its own set of problems, as ROS2 is not compatible with Windows, complicating the process of recording and transmitting radar data to other devices. This incompatibility forced us to find cumbersome workarounds to transfer data between systems.

3.2 Software Integration Difficulties

The software side of the project presented its own set of challenges, primarily revolving around the integration of various components and the compatibility of different operating systems. Initially, we used the TI mmWave Studio on Windows to capture raw radar data, but this approach quickly became untenable as it required significant manual intervention and custom code.

After extensive searching, we discovered the official TI mmWave ROS2 driver, which, although only compatible with Ubuntu 22.04 and limited to point cloud data, seemed sufficient for our needs. This discovery was a breakthrough as it allowed us to standardize our development environment using ROS2, which was also the communication protocol used by the Turtlebot 4.

However, the release of the Radarize (17) code provided the necessary tools to capture radar data and broadcast it to ROS1. This release came very late in our development cycle and was only compatible with Ubuntu 20.04, causing our group's attention to split. It was only after significant investigation that we discovered our hardware was not suitable for running the Radarize pipeline in real-time.

The constant switching between operating systems from Windows to Ubuntu 22.04 for ROS2 and Ubuntu 20.04 for ROS1 made it difficult to maintain a cohesive development workflow. This fragmentation led to numerous compatibility issues and significantly hampered our progress.

3.3 Knowledge Gaps

Our team also had to overcome a steep learning curve due to our initial lack of experience with both ROS2 and radar technologies. Understanding how the radar works and tuning its parameters for optimal performance required extensive research and experimentation (2) (13). The complexity of the ROS2 framework added to this challenge, as we had to familiarize ourselves with its various nodes, topics, and services to achieve effective integration and visualization of the SLAM system.

Additionally, the radar's low resolution posed a significant challenge in accurately mapping the environment. This limitation necessitated a deeper understanding of how other sensors like LiDAR and cameras work to compare their mapping capabilities and validate the accuracy of our radar-based system.

3.4 Adaptation and Solutions

Despite these challenges, we adapted by leveraging the Turtlebot's odometry for accurate localization, thereby reducing our dependency on the radar for odometry calculations. We developed a custom OctoMap Node that efficiently constructs a 2D occupancy map from the 3D Pointcloud data provided by the radar. This approach allowed us to maintain the advantages of radar technology, such as its ability to function in adverse conditions and detect transparent materials.

Furthermore, we utilized ROS2's 3D Robot Visualizer (rviz) to provide a comprehensive visualization of the PointCloud data, robot localization, and the resultant map (8). This visualization tool was instrumental in debugging and refining our SLAM pipeline.

In conclusion, the development of our radar-based SLAM system was fraught with challenges, from hardware malfunctions and integration issues to knowledge gaps. However, these challenges drove us to innovate and adapt, resulting in an effective SLAM system that leverages the unique capabilities of radar technology.

Acknowledgments

We extend our gratitude to the teaching team, whose guidance and expertise were valuable throughout the course of this project.

References

[1] et al., H.Z.: 4d radar-inertial slam based on factor graph optimization. SAE International Journal of Connected and Automated Vehicles p. 10 (2024). https://doi.org/10.4271/2024-01-2844

[2] et al., J.H.K.: Modeling point uncertainty in radar slam. arXiv preprint arXiv:2402.16082 (2023), https://arxiv.org/abs/2402.16082

[3] Barnes, D., Gadd, M., Murcutt, P., Newman, P.: Radarslam: Radar based large-scale slam in all weathers. arXiv preprint arXiv:2005.02198 (2019), https://arxiv.org/abs/2005.02198

[4] Cen, Z., Newman, P.: Maroam: Map-based radar slam through two-step feature selection. arXiv preprint arXiv:2210.13797 (2019), https://arxiv.org/abs/2210.13797

[5] Chen, D., Wang, N., Xu, R., Xie, W., Bao, H., Zhang, G.: Rnin-vio: Robust neural inertial navigation aided visual-inertial odometry in challenging scenes. In: 2021 IEEE International Symposium on Mixed and Augmented Reality (ISMAR). pp. 275–283 (2021). https://doi.org/10.1109/ISMAR52148.2021.00042

[6] Dokmanic, I., Daudet, L., Vetterli, M.: From acoustic room reconstruction to slam. In: 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 6345–6349 (2016). https://doi.org/10.1109/ICASSP.2016.7472854

[7] Guo, H., Sie, E., Wu, X., Vasisht, D.: Radarize: Large-scale radar slam for indoor environments. arXiv preprint arXiv:2311.11260 (2024), https://arxiv.org/abs/2311.11260

[8] Hess, W., Kohler, D., Rapp, H., Andor, D.: Real-time loop closure in 2d lidar slam. In: 2016 IEEE International Conference on Robotics and Automation (ICRA). pp. 1271–1278 (2016). https://doi.org/10.1109/ICRA.2016.7487258

[9] Hong, Z., Petillot, Y., Wallace, A., Wang, S.: Radar slam: A robust slam system for all weather conditions. arXiv preprint arXiv:2104.05347 (2021), https://arxiv.org/abs/2104.05347

[10] Instruments, T.: Single-chip 76-GHz to 81-GHz industrial radar sensor integrating DSP, MCU and radar accelerator. https://www.ti.com/product/IWR1843

[11] Kwon, S.Y., Kwak, S., Kim, J., Lee, S.: Radar sensor-based ego-motion estimation and indoor environment mapping. IEEE Sensors Journal **23**(14), 16020–16031 (2023). https://doi.org/10.1109/JSEN.2023.3295324

[12] Li, Y., Yang, S., Xiu, X., Miao, Z.: A spatiotemporal calibration algorithm for imu–lidar navigation system based on similarity of motion trajectories. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 7637–7644 (2020). https://doi.org/10.1109/IROS45743.2020.9340921

[13] Lim, H., Kim, D., Kim, B., Myung, H.: Adalio: Robust adaptive lidar-inertial odometry in degenerate indoor environments. IEEE Robotics and Automation Letters **5**(4), 6674–6681 (2020). https://doi.org/10.1109/LRA.2020.3013291

[14] Lu, C.X., Rosa, S., Zhao, P., Wang, B., Chen, C., Stankovic, J.A., Trigoni, N., Markham, A.: See through smoke: Robust indoor mapping with low-cost mmwave radar. In: Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services (MobiSys '20). pp. 14–27 (2020). https://doi.org/10.1145/3386901.3388912

[15] Mur-Artal, R., Montiel, J.M.M., Tardos, J.D.: Orb-slam: A versatile and accurate monocular slam system. IEEE Transactions on Robotics **31**(5), 1147–1163 (2015). https://doi.org/10.1109/TRO.2015.2463671

[16] Sie, E., Liu, Z., Vasisht, D.: Batmobility: Towards flying without seeing for autonomous drones. In: The 29th Annual International Conference on Mobile Computing and Networking (ACM MobiCom '23) (2023). https://doi.org/10.1145/3570361.3592532

[17] Sie, E., Wu, X., Guo, H., Vasisht, D.: Radarize: Enhancing radar slam with generalizable doppler-based odometry. The 22nd Annual International Conference on Mobile Systems, Applications and Services (MOBISYS '24) (2024). https://doi.org/10.1145/3643832.3661871, https://radarize.github.io

[18] Van Trees, H.L.: Detection, Estimation, and Modulation Theory, Part IV: Optimum Array Processing. Wiley-Interscience (2004)

[19] Zhao, Y., Lu, X., Ye, T.: One robust loosely coupled 4d millimeter-wave image radar slam method. In: SAE 2023 Intelligent and Connected Vehicles Symposium. p. 9. SAE International (2023). https://doi.org/10.4271/2023-01-7051