Arizona State University

EGR 530: Final Project Presentation

Comparative Analysis of SLAM Algorithms

Team 5
Jayaram, Pavan, Pranav, Swapnil, Yoganandam

What's in this presentation

1. Problem Definition
2. SLAM Algorithms Overview
3. Methodology
4. Results
5. Conclusion
6. Challenges Faced
7. Future Scope

Scope of work

Proposed

- Comparing ORB SLAM and Cartographer SLAM
- Metrics for Comparison
- Proposing New SLAM Algorithm

Actual

- Comparing Cartographer SLAM and GMapping SLAM, overview of ORB-SLAM
- Metrics for Comparison (Error in mapping, pose accuracy)
- Proposed New SLAM Algorithm

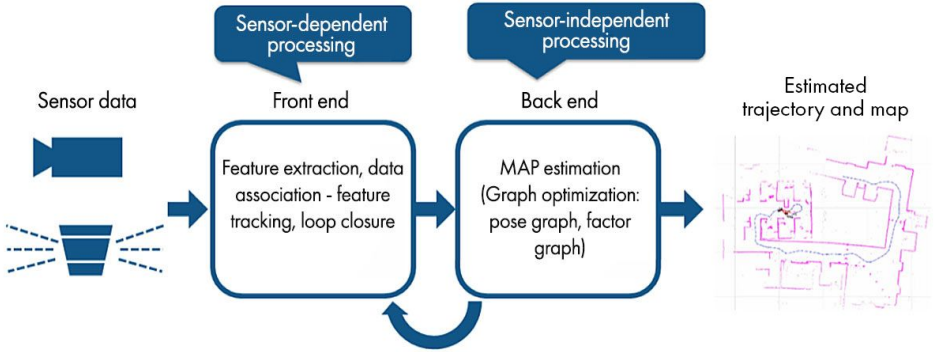
Reasons for deviation:

- Challenges faced in comparing ORB SLAM(3D map) with Cartographer SLAM(2D map)
- Could not save the ORB SLAM Map to calculate any sort of Accuracy or Error with the ground truth. ORB SLAM could only be evaluated on certain pre existing datasets.

Problem Definition

- SLAM stands for Simultaneous Localization and Mapping, which is widely used by Autonomous Vehicles to build maps and localize the vehicle in that map.
- SLAM algorithms allow the vehicle to map out unknown environments. Engineers use the map information to carry out tasks such as path planning and obstacle avoidance.
- We tried to compare two SLAM algorithms, Cartographer SLAM and GMapping SLAM, for an indoor environment.

How SLAM Works?

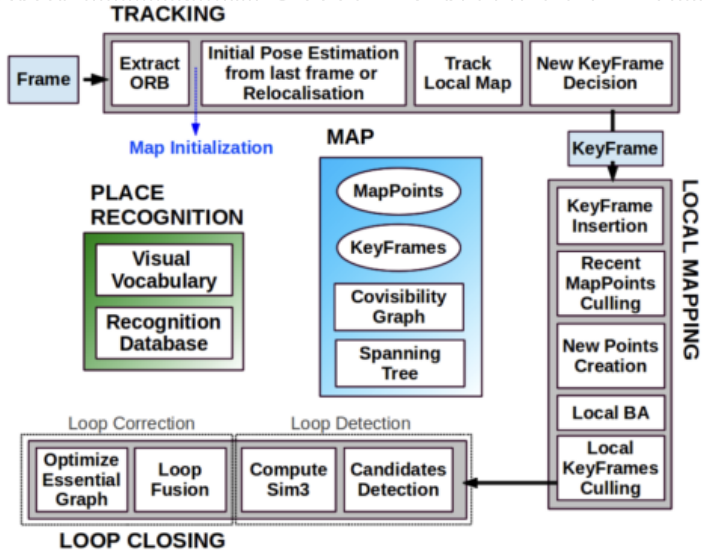


ORB-SLAM

- ORB-SLAM (**O**riented FAST and **R**otated **B**RIEF **SLAM)** is a feature based SLAM that operates in real-time,in small and large indoor and outdoor environments.

The system consists of 3 main components -

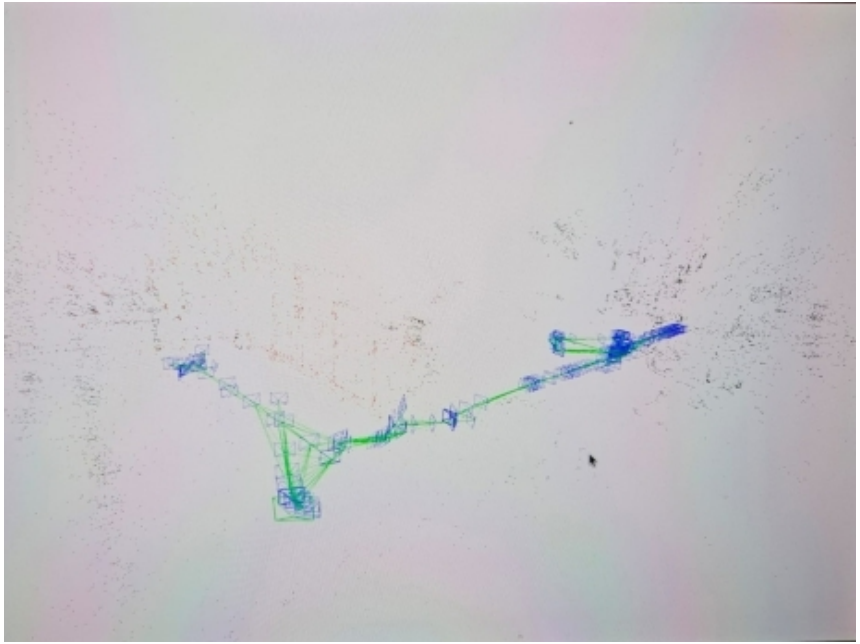
1) Tracking -The tracking is in charge of localizing the camera with every frame and deciding when to insert a new keyframe.
2) Local mapping - The local mapping processes new keyframes and performs local BA to achieve an optimal reconstruction in the surroundings of the camera pose.
3) Loop closing - The loop closing thread takes Ki , the last keyframe processed by the local mapping, and tries to detect and close loops.



For detailed explanation, Refer here:
https://drive.google.com/file/d/1qGu41LyW4bNLZ1f4ECC-FPPVYdBFU5T0/view?usp=sharing

ORB-SLAM Result Overview

- We conducted our initial exploration on ORB SLAM, mapped it and tried to store it, but were unsuccessful in trying to save the generated map.
- We mapped one layout and here is the map created



SLAM Algorithms Overview

CARTOGRAPHER SLAM

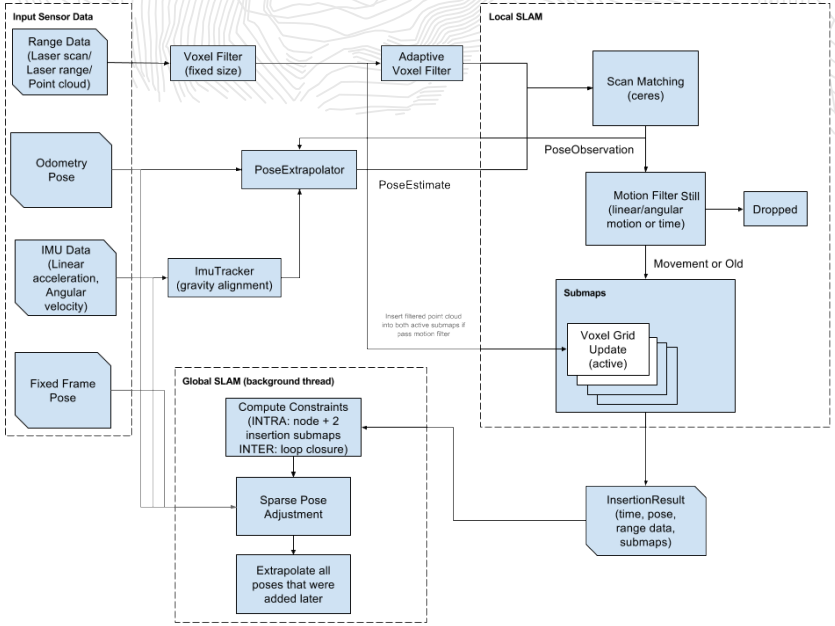
- Cartographer is an open-source SLAM library developed by Google.
- Supports both 2D and 3D mapping
- It is a graph-building algorithm based on graph optimization (multi-threaded backend optimization, problem optimization of ceiling construction).
- Data from multiple sensors (such as LIDAR, IMU, and cameras) can be combined to simultaneously calculate the sensor's position and map the environment around the sensor.

GMAPPING SLAM

- GMapping is short for Grid-Based Mapping, it only supports 2D mapping.
- It is a commonly used open source SLAM algorithm based on the filtered SLAM framework.
- It is particularly well-suited for mapping environments using LIDAR sensors, which provide range measurements in the form of point clouds.
- GMapping is based on the Rao-Blackwellized Particle Filter (RBPF) algorithm, which separates the real-time positioning and mapping processes.

Introduction to Cartographer SLAM

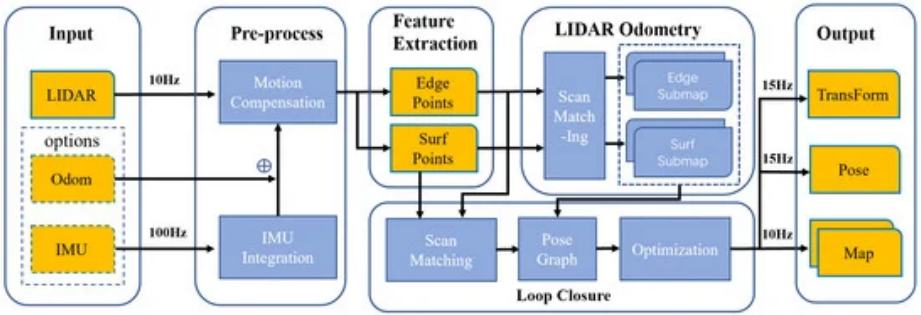
- Core Components:
  - Probability Grid Maps: Utilizes grid-based maps where each cell represents the likelihood of occupancy, which dynamically updates based on sensor data.
  - Sensor Data Integration: Integrates multiple types of data (LIDAR, IMU, odometry) through sensor fusion to enhance accuracy and reduce localization errors.
- Technological Foundations:
  - Built on advanced algorithms for scan matching and real-time optimization.
  - Employs techniques such as loop closures and pose graph optimization to maintain global consistency across large datasets.



Cartographer high level system

Technical Processes in Cartographer

1. Initialization and Sensor Integration: Start with initial position and integrate sensor data (LIDAR, IMUs, odometry) for coherent mapping.
2. Scan Matching and Pose Estimation: Align new sensor data with maps using scan-matching techniques like ICP and optimize pose estimates using methods like Levenberg-Marquardt.
3. Loop Closure and Map Optimization: Detect when revisiting mapped areas, refine map and trajectory estimates, and continuously optimize the map in real-time.



Real-Time Lidar Odometry and Mapping with Loop Closure

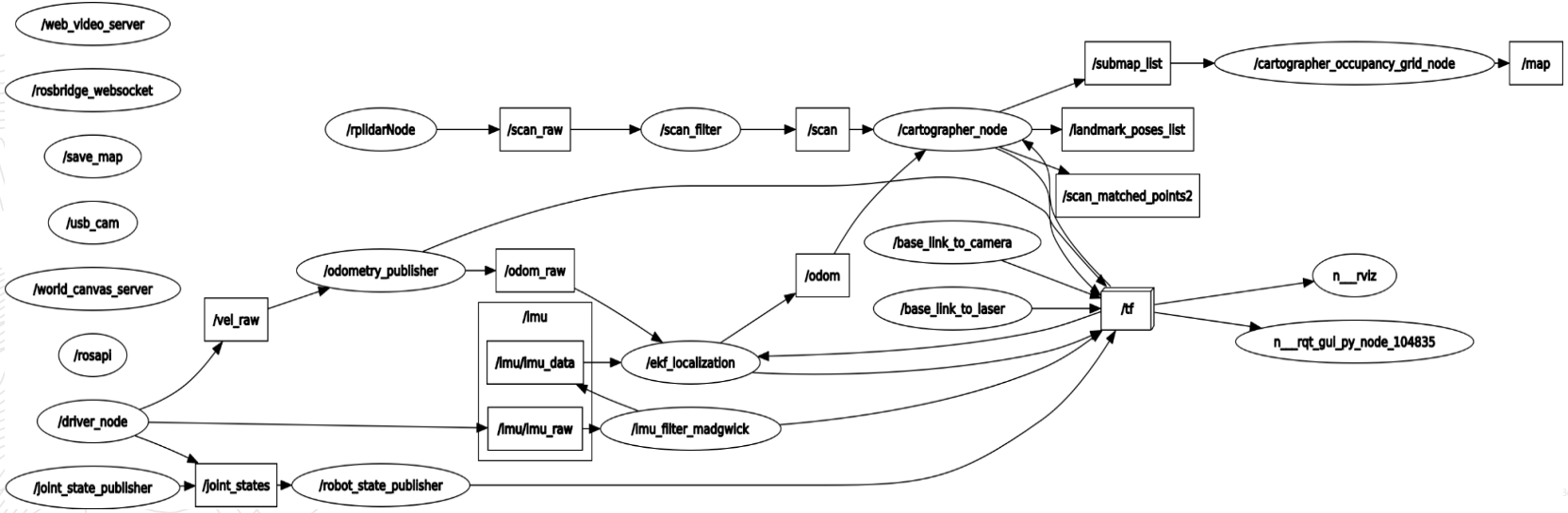
For detailed explanation, Refer here:
https://drive.google.com/file/d/1PLK_M5bFnc43ta5w6pTmVh16QdtvQ5xt/view?usp=drive_link

Additional Info

Command Used:

- roslaunch yahboomcar_nav laser_usb_bringup.launch #for camera bringup
- roslaunch yahboomcar_nav yahboomcar_map.launch use_rviz:=false map_type:=cartographer #for launching Cartographer algorithm on the robot
- roslaunch yahboomcar_nav view_cartographer.launch #for rviz view of map
- rosrun teleop_twist_keyboard teleop_twist_keyboard.py #for control of robot

RQT_Graph



Introduction to Gmapping SLAM

•Particle Filter Framework: GMapping utilizes a particle filter where each particle represents a potential trajectory of the robot along with an associated map.
•Initialization: Particles represent possible robot trajectories with initial empty or minimally informed maps.
•Motion Update: Predicts new particle positions based on odometry, considering sensor inaccuracies.
•Measurement Update: Updates particle map hypotheses by integrating new sensor data and computing likelihoods.
•Resampling: Particles are resampled based on agreement with measurements, reducing uncertainty.
•Map Update: Each particle's map is updated with new sensor data, improving accuracy over time.

Overview of Gmapping SLAM

- Particle Filter Framework: GMapping utilizes a particle filter where each particle represents a potential trajectory of the robot along with an associated map.
- Initialization: Particles represent possible robot trajectories with initial empty or minimally informed maps.
- Motion Update: Predicts new particle positions based on odometry, considering sensor inaccuracies.
- Measurement Update: Updates particle map hypotheses by integrating new sensor data and computing likelihoods.
- Resampling: Particles are resampled based on agreement with measurements, reducing uncertainty.
- Map Update: Each particle's map is updated with new sensor data, improving accuracy over time.

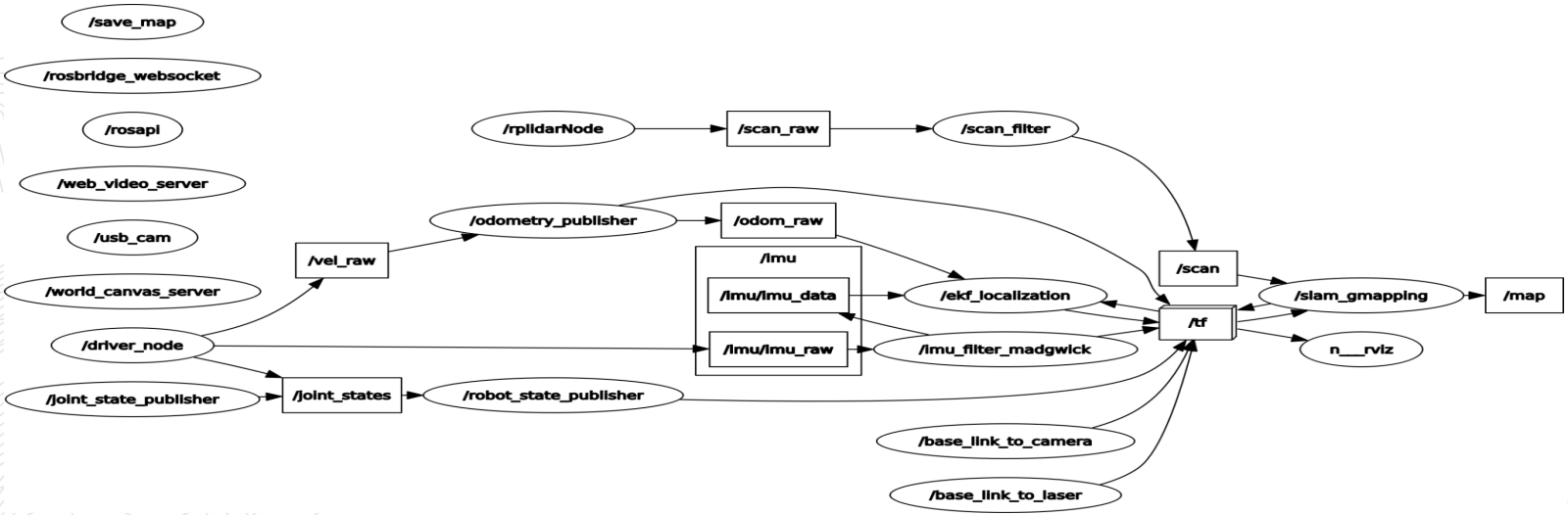
For detailed explanation, Refer here:
https://drive.google.com/file/d/1N_f75tOTYGCbBj7RuULWbpE20x_U8PT8/view?usp=drive_link

Additional Info

Command Used:

- roslaunch yahboomcar_nav laser_usb_bringup.launch #for camera bringup
- roslaunch yahboomcar_nav yahboomcar_map.launch use_rviz:=false map_type:=gmapping #for launching GMapping algorithm on the robot
- roslaunch yahboomcar_nav view_gmapping.launch #for rviz view of map
- rosrun teleop_twist_keyboard teleop_twist_keyboard.py #for control of robot

RQT_Graph



Comparison of SLAM Algorithms

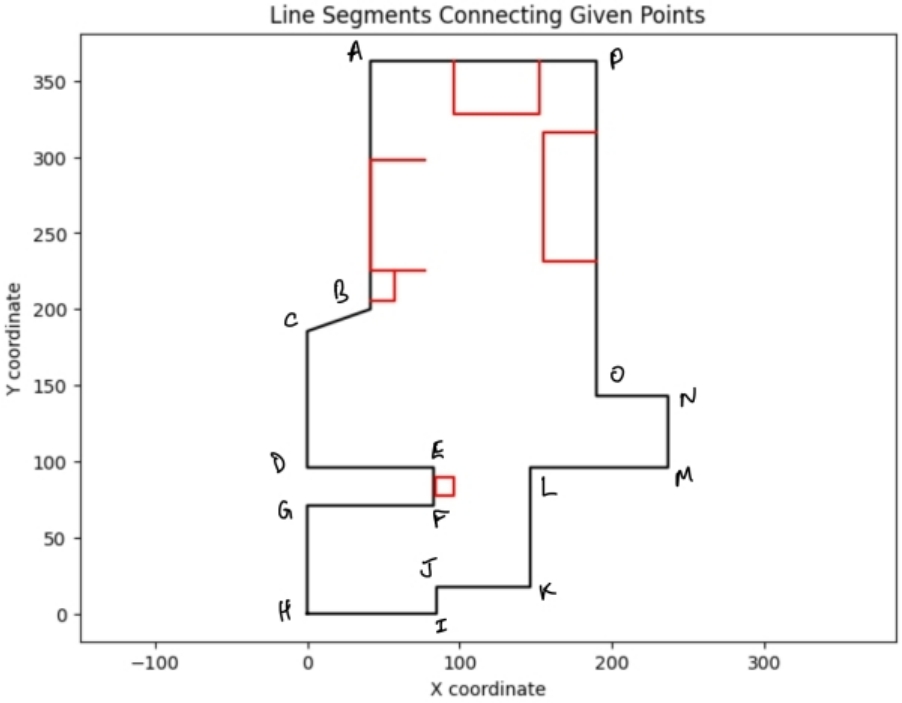
Feature	Cartographer	GMapping
Mapping Dimensions	3D and 2D mapping	Primarily 2D mapping
Algorithm	Real-time loop closure and pose graph optimization	Rao-Blackwellized Particle Filter (RBPF)
Sensor Integration	Integrates multiple sensor types simultaneously	Primarily uses LIDAR or other range finders
Dynamic Environments	Handles dynamic objects and is robust to changes	Less robust to dynamic objects and changes
Computational Load	Higher due to 3D mapping and real-time processing	Lower, suitable for less powerful systems
Use Cases	Suitable for complex environments and applications	Used extensively for indoor and simple environments
Complexity	Higher complexity, steeper learning curve	Simpler and well-documented, easier to start with
Real-Time Performance	Designed for real-time SLAM	Real-time but may struggle in large environments
Kidnapping Problem	Better at handling sudden relocations	Can struggle with the kidnapping problem
Map Representation	Occupancy grid maps and other representations	Occupancy grid maps

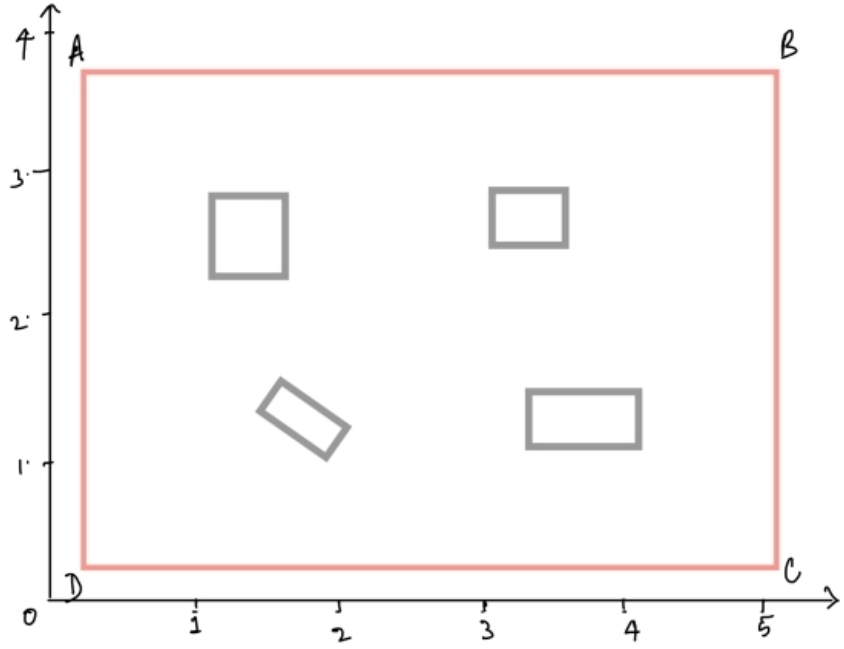
Methodology

- Both, Cartographer SLAM and GMapping SLAM are types of LIDAR based SLAM. We have used RPLIDAR, which is a 2D lidar which uses Laser Triangulation ranging principles.
- We controlled the robot using the Teleopkey commands and moved it around the house.
- The speed of the robot was slow in order to incorporate most features in the map. The slower the speed the better the map.
- The path travelled by the robot was kept the same, in order keep it a fair comparison.

Results

Ground truth maps for both the environments. We measured the dimensions of both the layouts using a measuring tape.





Results

- There are no readily available methods to compare different SLAM algorithms.
- We tried to compare the two SLAM algorithms based on their error % in getting the dimensions of the map correct.
- We measured the ground truth using a measuring tape. For simplicity we calculated the dimensions of the boundary and omitted the obstacles like the suitcases (black outline) in layout 1 and sofas and tables (red outline) in layout 2.
- For getting the dimensions from the generated maps, we used the PGM files and found out the number of pixels involved in each line. Then we multiplied the number of pixels by their resolution (0.05m) and found the lengths.
- We computed the Absolute Error between the Actual Measurement and the Calculated Measurements from the generated maps.
- We then calculated the error by using MSE (Mean Square Error) and MAE (Mean Absolute Error) for both the Algorithms across both Layouts.

Results

All dimensions are in meters

LAYOUT 1

Side	Ground Truth	Cartographer	Error	Sq Error	GMapping	Error	Sq Error
AB	4.9	4.81	0.09	0.0081	4.85	0.05	0.0025
BC	3.5	3.75	0.25	0.0625	3.66	0.16	0.0256
CD	4.9	4.75	0.15	0.0225	4.72	0.18	0.0324
DA	3.5	3.3	0.2	0.04	3.6	0.1	0.01

LAYOUT 2

Side	Ground Truth	Cartographer	Error	Sq Error	GMapping	Error	Sq Errror
AB	4.15	4.25	0.1	0.01	4	0.15	0.0225
BC	1.2	1.5	0.3	0.09	1.47	0.27	0.0729
CD	2.27	2.32	0.05	0.0025	2.15	0.12	0.0144
DE	1.95	1.8	0.15	0.0225	2	0.05	0.0025
EF	0.63	0.8	0.17	0.0289	0.76	0.13	0.0169
FG	2.1	2	0.1	0.01	2.15	0.05	0.0025
GH	1.8	1.95	0.15	0.0225	1.85	0.05	0.0025
HI	2.16	2	0.16	0.0256	2.05	0.11	0.0121
IJ	0.5	0.52	0.02	0.0004	0.65	0.15	0.0225
JK	1.56	1.7	0.14	0.0196	1.87	0.31	0.0961
KL	1.2	1.35	0.15	0.0225	1.35	0.15	0.0225
LM	2.3	2.2	0.1	0.01	2.16	0.14	0.0196
MN	1.2	1	0.2	0.04	1.33	0.13	0.0169
NO	1.2	1.37	0.17	0.0289	1.12	0.08	0.0064
OP	6.58	6.25	0.33	0.1089	6.05	0.53	0.2809
PA	3.8	3.65	0.15	0.0225	3.96	0.16	0.0256

Results

ERROR RESULTS

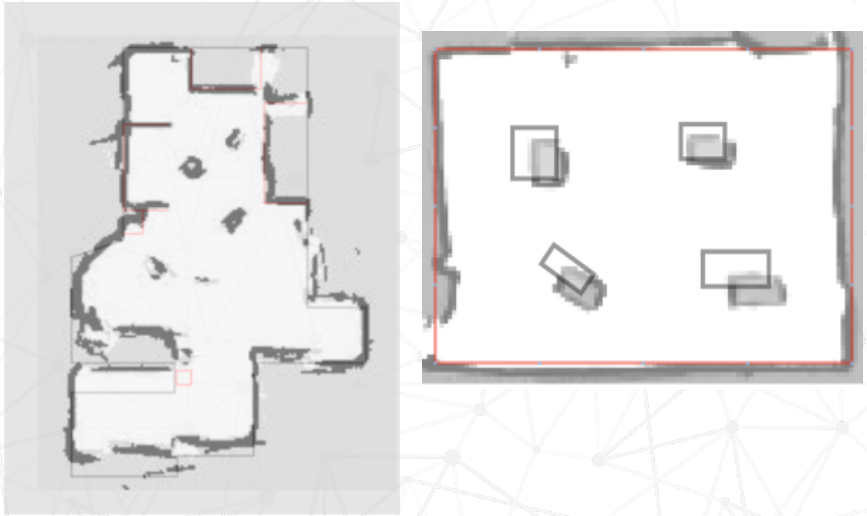
MAP	Algorithm	MAE	MSE
Layout 1	Cartographer	0.1725	0.0333
	GMapping	0.1225	0.0177
Layout 2	Cartographer	0.1513	0.029
	GMapping	0.1612	0.0398

- From the table we can see that GMapping has a lower MSE and MAE on simpler and less complicated layout (Layout 1), while Cartographer has a lower MSE and MAE on more complicated layout (Layout 2).
- From the Images in the next slide, we can see that though the errors may be less, the precision (tracing the map), is not much, indicating that just the error is not a very good estimate to compare the maps.

Results

Ground truth maps vs SLAM Algorithms for both the environments. We overlaid the Ground truth map and the generated map one on the other to see how similar the maps are to the ground truth. From this we can see that the algorithms do a decent job at mapping the environments correctly, but at regions of the map, it is not able to get the dimension right, or the location of the obstacles accurately.

Gmapping vs Ground Truth



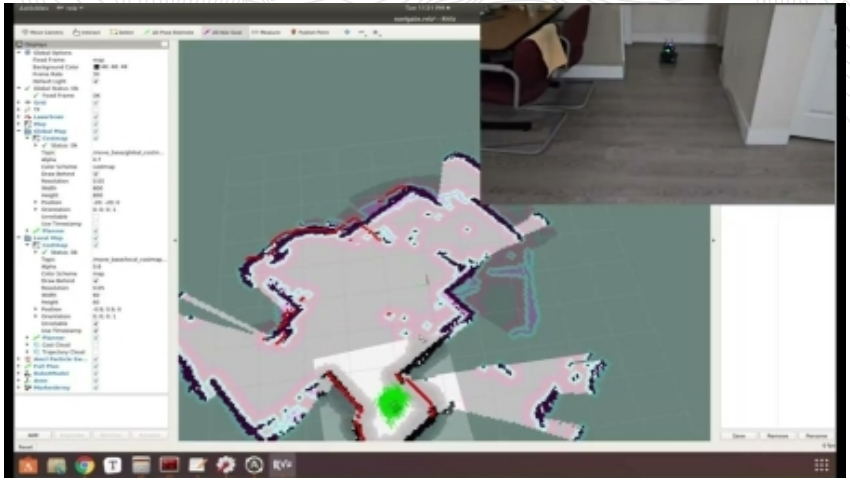
Cartographer vs Ground Truth

Conclusion

- From the results we can see that both Cartographer and GMapping SLAM Algorithms have performed decently well in mapping the two environments which we designed.
- GMapping did better on smaller and less complicated layout compared to more complicated environments.
- The speed of the robot played a significant role in the accuracy of the measurements.
- With a 2D Single Line LIDAR mounted on the robot, the map constructed is a 2D map. In the environment with obstacles, the walls are occluded from LIDAR’s view.

Challenges Faced

1. The main challenge faced was that it is hard to quantitatively compare these SLAM Algorithms as the accuracy was more or less similar for our chosen environments.
2. Blind Spot Challenges:
    - The height of the LIDAR is at 7 inches. Obstacles located below the 7-inch threshold remain undetected, creating blind spots.
3. Navigation Issues:
    - The robot navigates paths that may lead to collisions with these undetected obstacles in the blind spots.



Proposed SLAM to tackle limitations

1. Obstacle Detection Enhancement:
    - Integrate additional sensors to detect obstacles within the LIDAR’s blind spots and augment the map data accordingly.
2. Utilization of Depth Cameras:
    - Employ depth cameras to identify and map lower-lying obstacles not detected by the LIDAR.
3. Sensor Calibration:
    - Ensure precise alignment and calibration between the depth camera and the LIDAR for effective obstacle detection.

Contributions

Name	ASU ID	Worked on
Jayaram	1228699721	Cartographer Analysis, comparison study and metrics, ground truth measurements
Pavan	1228186754	Gmapping Analysis, Algorithmic Complexity analysis
Pranav	1230287970	ORB SLAM, Mapping Cartographer and GMapping for 1 layout, Methodology, PPT
Swapnil	1225866657	Ground truth calculations, actual length calculations from obtained maps
Yoganandam	1225622606	ORB SLAM, ground truth and actual calculations, Mapping SLAM Algos for 1 Layout, proposed improved SLAM

Demonstration Video