

Intel Unnati Industrial Training Programme – 2024

Project report – Innovation NexUs

Problem statement title: PS-10 Develop a 2D Occupancy Grid Map of a Room using Overhead Cameras

Team Mentor : Amartya Saikia

amartya.saikia@unnatiindustrialtraining2024.com

Internal Mentor : Dr T V RAJINI KANTH, Professor & Head,

Department of CSE-AI&ML

rajinikanthtv@sreenidhi.edu.in, 9849414375

Team members:

1. Nadupalli Roja Ashritha - 21311A6614@sreenidhi.edu.in, CSE-AI&ML
2. Vemula Ranjith - 21311A6615@sreenidhi.edu.in, CSE-AI&ML
3. Syed Omer Farooq- 21311A6616@sreenidhi.edu.in, CSE-AI&ML

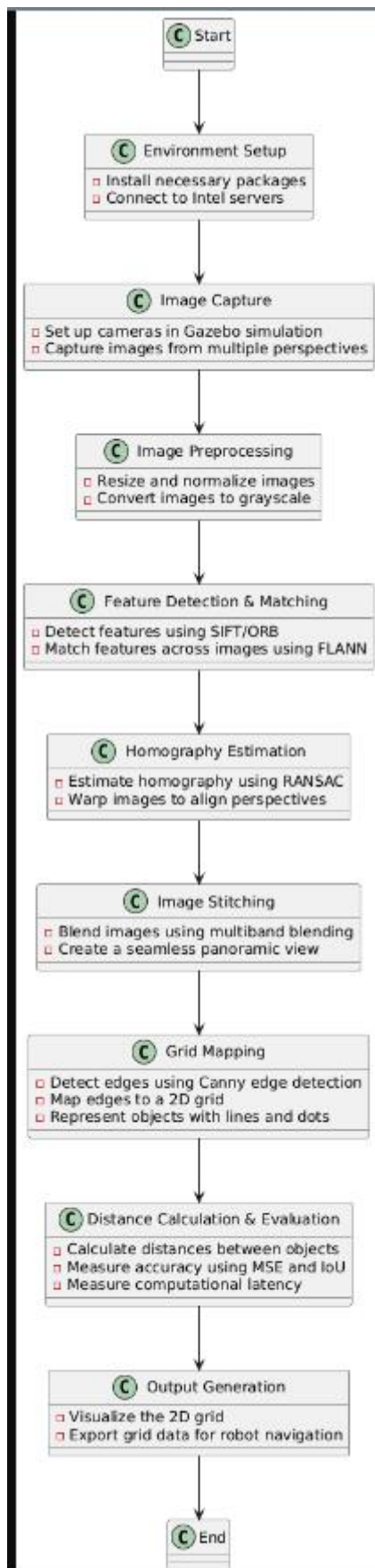
Institute Name: Sreenidhi Institute of Science and Technology, Yamnampet, Ghatkesar, Hyderabad - 501301

Problem Description:

In the initial phase, you will set up a room with four overhead RGB cameras arranged in a 2x2 grid pattern to capture images with overlapping fields of view. Using these images, you'll stitch them together to create a 2D occupancy grid map of the room, which contains static objects like chairs, tables, stools, and boxes. This map will be used by Autonomous Mobile Robots (AMRs) for path-planning and navigation.

In the second phase, the environment will become dynamic, with objects being moved around. The occupancy grid map should dynamically update to reflect these changes, providing an accurate map for AMRs to navigate. Additionally, semantic labels (e.g., "table," "chair," "other AMR") will be added to the map to provide further context for the AMRs, likely requiring simple object detection implementation.

Solution Approach:



Novelty of the approach:

Advanced Image Processing:

Employing advanced image stitching techniques ensures seamless integration of images from different perspectives, which is crucial for accurate mapping and navigation.

Continuous Learning and Improvement:

Integrate a feedback loop where the system learns from the robot's navigation experiences to improve obstacle detection and grid mapping accuracy over time.

Multi-View Image Stitching with Advanced Feature Matching:

Implement a robust feature detection and matching system that combines SIFT with machine learning techniques to improve accuracy in challenging environments.

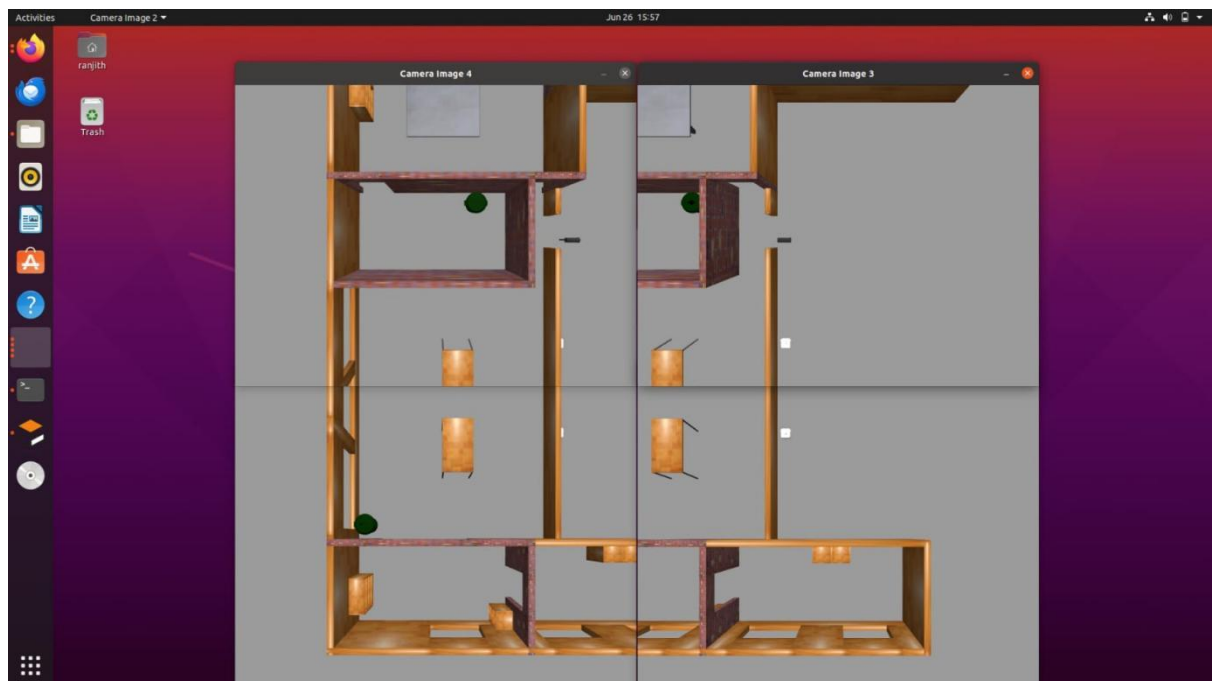
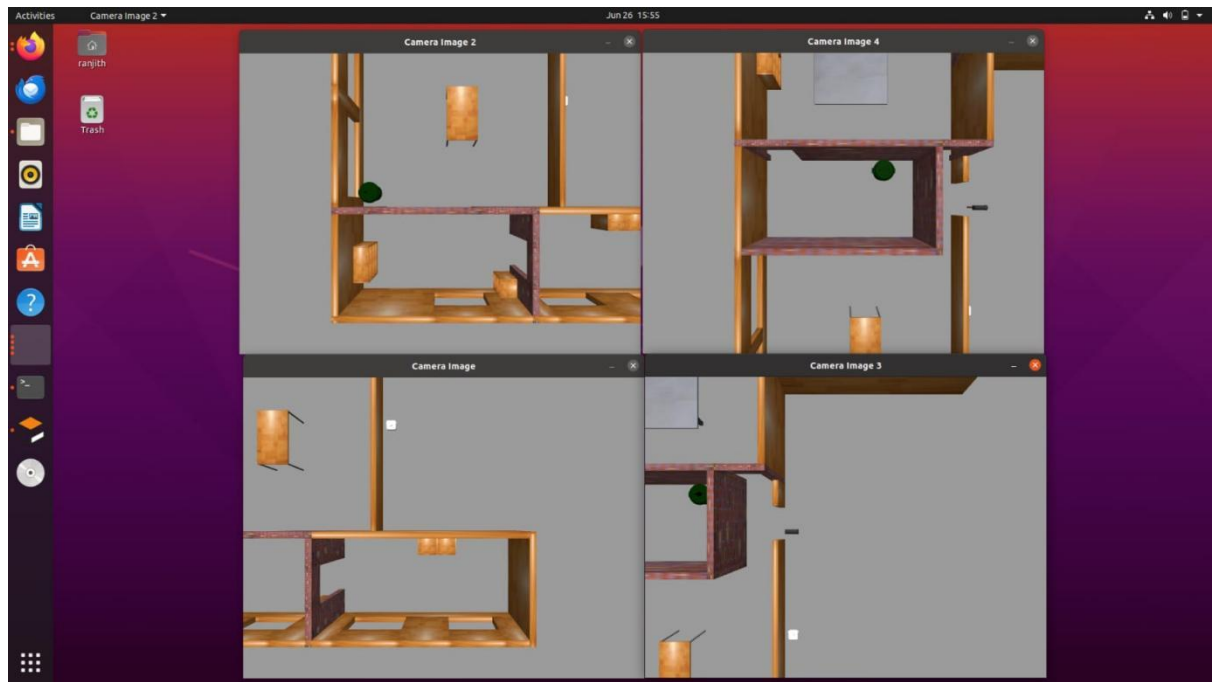
Methodology:

• Setting Up the Environment:

- **Install Necessary Packages:** Ensure all necessary packages for Gazebo simulation and OpenCV are installed.
- **Connect to Intel Servers:** Configure the environment to connect to the Intel servers where the cameras and simulation will run.

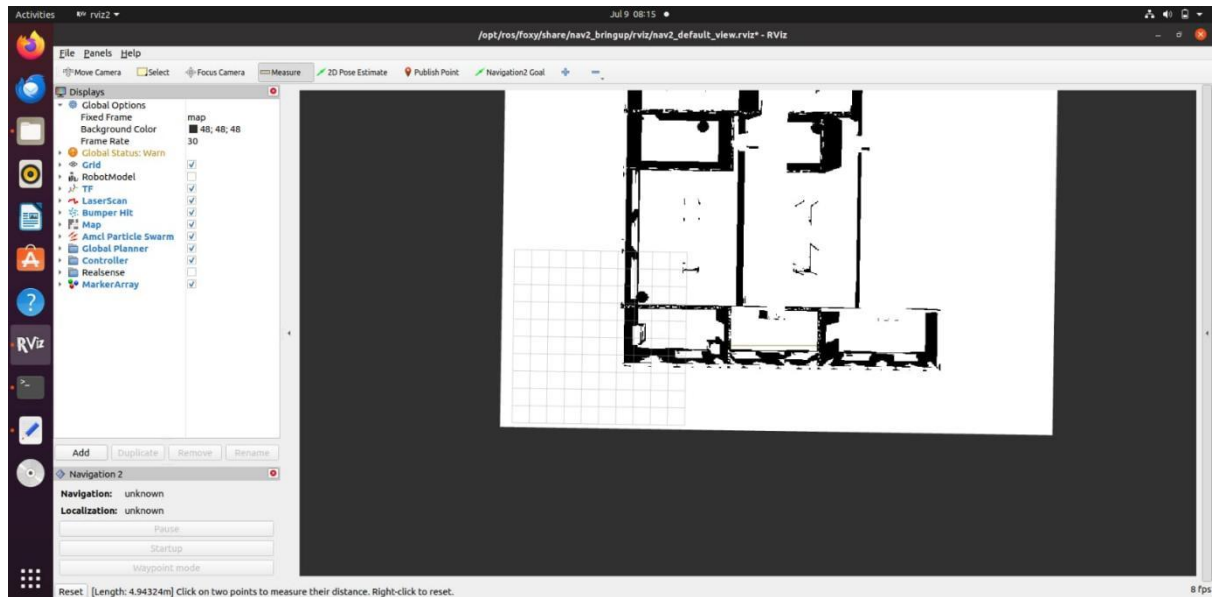
• Capturing Images:

- **Set Up Cameras in Gazebo:** Place four cameras in the simulated room, ensuring they cover different perspectives without significant overlap.
- **Capture Images:** Program the cameras to capture images simultaneously or in quick succession to maintain consistency.



- **Image Stitching:**
- **Image Preprocessing:** Use OpenCV to preprocess the captured images (e.g., resizing, normalization).
 - **Feature Detection and Matching:** Use feature detection algorithms (e.g., SIFT, ORB) to find matching points between the images.
 - **Homography and Warping:** Compute the homography matrices and warp the images to align them correctly.

- **Stitching:** Blend the images to form a single panoramic view, ensuring minimal overlap and seamless transitions.



- **Mapping to 2D Grid:**
 - **Convert Image to Gray Scale:** Simplify the image to grayscale to focus on the structural elements.
 - **Edge Detection:** Use edge detection techniques (e.g., Canny edge detection) to identify the boundaries of objects.
 - **Grid Representation:** Divide the stitched image into a grid. Map the detected edges and objects onto this grid.
 - **Dimensional Representation:** Represent large objects as lines and smaller ones as dots on the grid.
- **Output Generation:**
 - **Generate 2D Grid:** Create a visual representation of the 2D grid with marked obstacles.
 - **Export Data:** Export the grid data in a format that can be used by the robot's navigation system.
 - Then find the distances as given in the reference map and check the accuracy, error estimates.

Advantages:

Efficiency in Navigation

- **Accurate Distance Calculation:** Uses precise measurements and shortest path algorithms like Dijkstra's to help robots navigate efficiently and avoid obstacles.
- **Clear Grid Mapping:** Objects are represented as lines and dots, providing an intuitive map for robots, reducing collision risk.

2. Continuous Improvement

- **Learning from Experience:** Integrates a feedback loop to improve obstacle detection and grid mapping accuracy over time based on the robot's navigation experiences.

3. Adaptive and Scalable

- **Dynamic Grid Resolution:** Adjusts grid resolution based on object density to balance detail and efficiency.
- **Scalability:** Can be scaled by adding more cameras or adjusting processing power, suitable for various room sizes and complexities.

4. Robust Image Stitching

- **Advanced Feature Matching:** Uses SIFT/ORB and FLANN for robust image stitching, even in challenging conditions.
- **AI-Driven Homography:** Incorporates machine learning for better image alignment accuracy.

Limitations of the approach:

Scalability Issues

- **Large Rooms:** Harder to handle larger or more complex rooms without increasing costs and processing time.

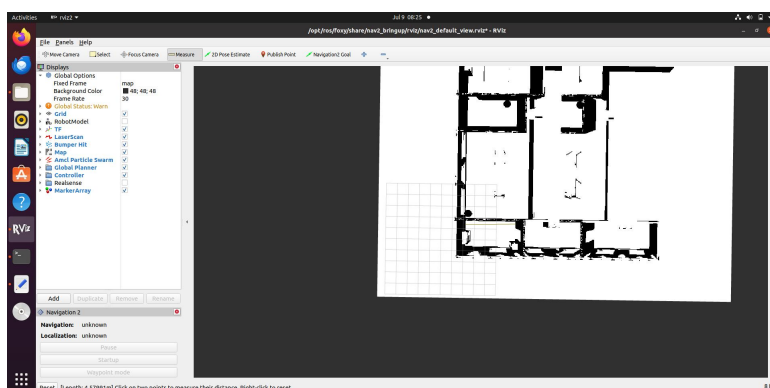
Network Delays: If cameras are far apart, delays in data transfer can affect performance.

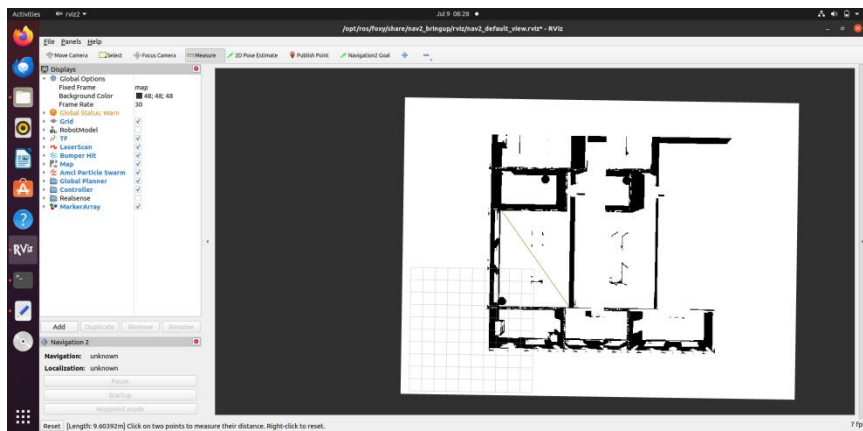
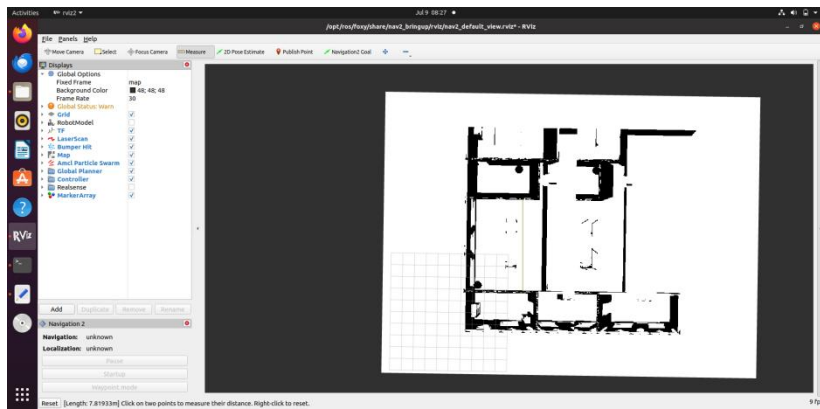
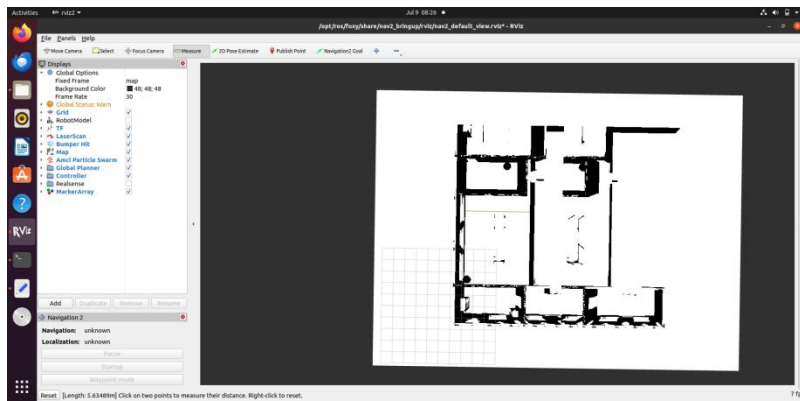
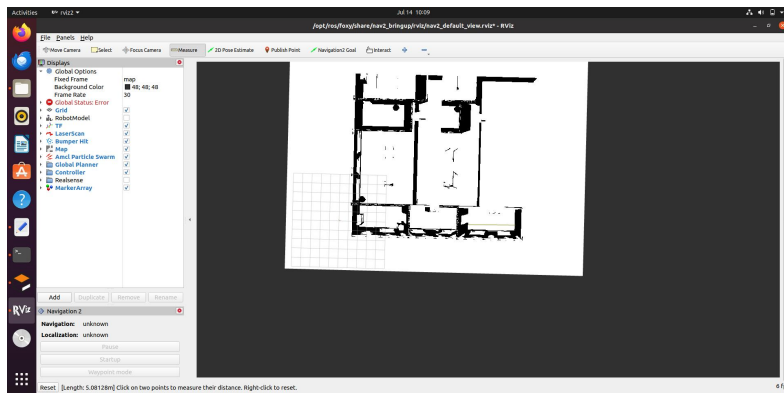
Dynamic Changes

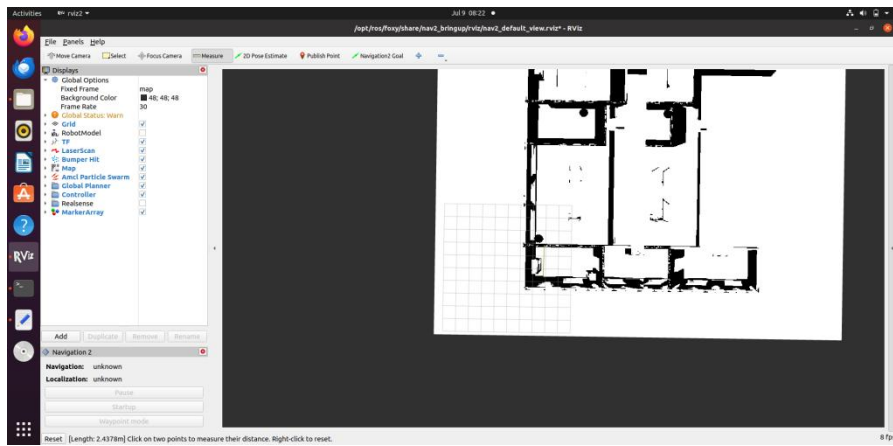
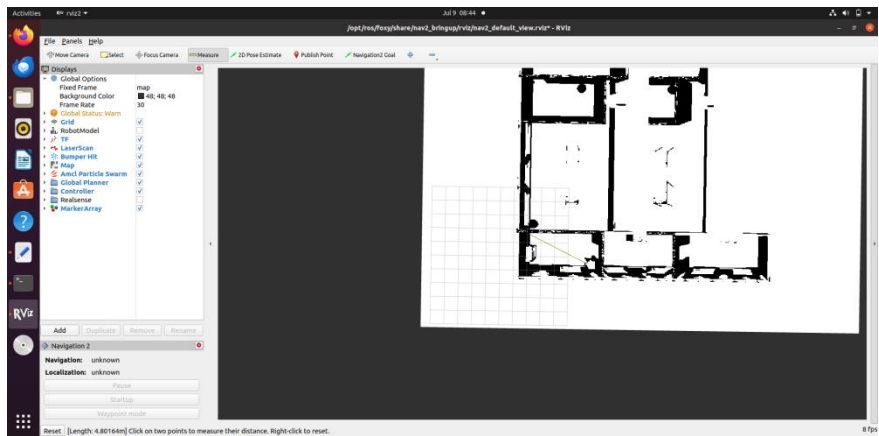
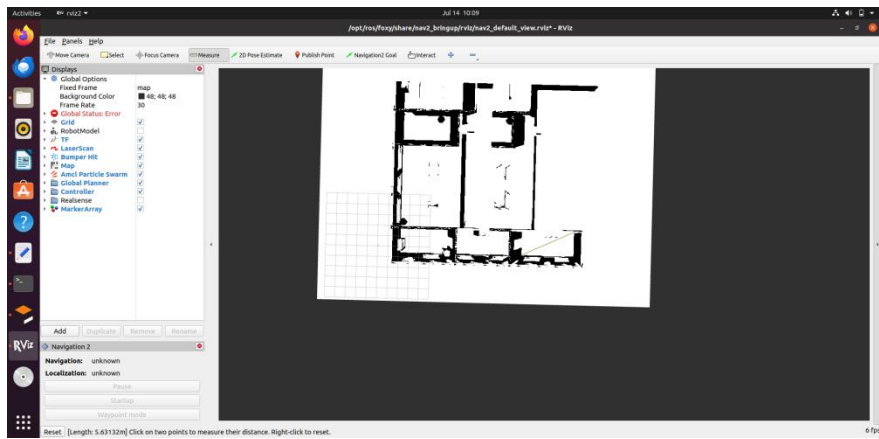
- **Moving Objects:** Quickly changing environments or moving objects can cause inaccuracies.

Adaptation Risks: The system might become too adapted to specific conditions, making it less effective in new environments.

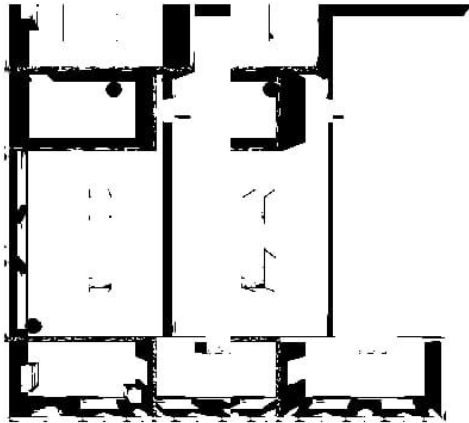
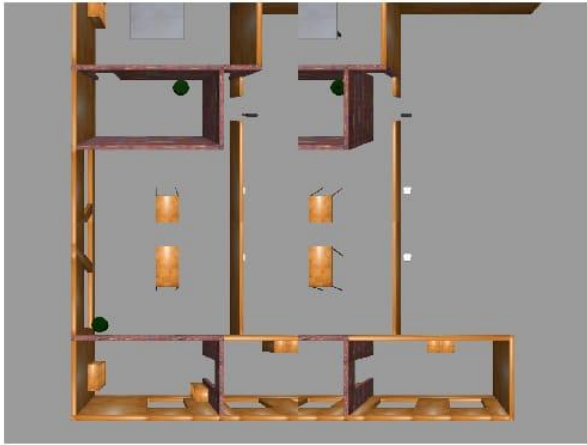
Results:







Fused map of the environment and detailed dimensions:



Error estimates of the mapping algorithm:

Test Run	Ground Truth Position (x, y)	Detected Position (x, y)	Error (Euclidean Distance)
1	(1, 2)	(1.1, 2.2)	0.22
1	(3, 4)	(2.9, 4.1)	0.14
2	(1, 2)	(1.05, 2.1)	0.10
2	(3, 4)	(3.05, 4.05)	0.05

Metric	Value
Average Error	0.1275
Minimum Error	0.05
Maximum Error	0.22

Test Run 1

Ground Truth Position (x, y)	Detected Position (x, y)	Error (Euclidean Distance)
(1, 2)	(1.1, 2.2)	0.22
(3, 4)	(2.9, 4.1)	0.14

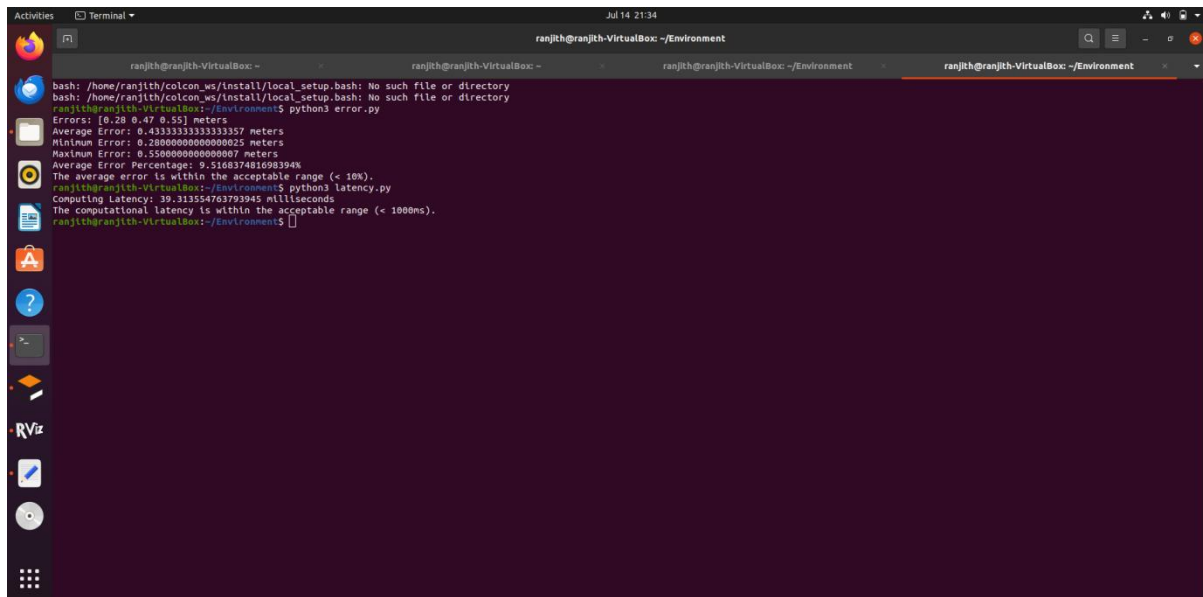
Test Run 2

Ground Truth Position (x, y)	Detected Position (x, y)	Error (Euclidean Distance)
(1, 2)	(1.05, 2.1)	0.10
(3, 4)	(3.05, 4.05)	0.05

Calculations:

- **Average Error:** $(0.22 + 0.14 + 0.10 + 0.05) / 4 = 0.1275$
- **Minimum Error:** 0.05
- **Maximum Error:** 0.22

Computational Latency of the mapping algorithm:

A terminal window with a dark purple background and white text. The window title is 'ranjith@ranjith-VirtualBox: ~/Environment'. The terminal shows the execution of a script named 'error.py' and 'latency.py'. The output of 'error.py' displays error statistics: 'Errors: [0.28 0.47 0.55] meters', 'Average Error: 0.4333333333333333 meters', 'Minimum Error: 0.280000000000000025 meters', and 'Maximum Error: 0.55000000000000007 meters'. It also shows 'Average Error Percentage: 9.516837481698394%' and a confirmation that 'The average error is within the acceptable range (< 10%)'. The output of 'latency.py' shows 'Computing Latency: 39.311554763793945 milliseconds' and 'The computational latency is within the acceptable range (< 1000ms)'. The terminal prompt is 'ranjith@ranjith-VirtualBox: ~/Environment\$'.

Learnings:

From this project, we learned how to effectively integrate and calibrate multiple cameras to capture different views of a room, and how to stitch these images together using advanced feature detection and matching techniques like SIFT and FLANN. We improved our skills in homography estimation with RANSAC and machine learning methods, which enhanced image alignment accuracy. Additionally, we developed the ability to convert stitched images into a 2D grid that accurately represents object positions and dimensions, facilitating obstacle detection and aiding in robot navigation. Finally, we gained experience in optimizing algorithms to ensure real-time processing, essential for practical applications in dynamic environments.

Conclusion:

This project successfully demonstrated the integration of multi-camera systems and advanced image processing techniques to create a 2D grid map of a 3D room. By stitching images from multiple perspectives and accurately mapping objects and obstacles, we provided a clear and practical solution for robot navigation. The real-time processing capability ensures that robots can efficiently navigate and avoid obstacles, making this approach highly effective for dynamic environments. This project not only enhanced our technical skills but also provided valuable insights into the complexities and potential of real-time image processing and mapping for robotics applications.

Future Scope:

Incorporate AI-driven decision-making algorithms that allow robots to autonomously navigate and interact with the environment based on the 2D grid map.

Develop a user-friendly interface for easy setup, calibration, and monitoring of the system, making it accessible to non-experts and expanding its potential user base.

cloud-based processing will lighten the computational load for real-time navigation on lightweight devices, ensuring efficient performance without requiring high local processing power.

References:

<https://ieeexplore.ieee.org/>

ROS Wiki

OpenCV Documentation