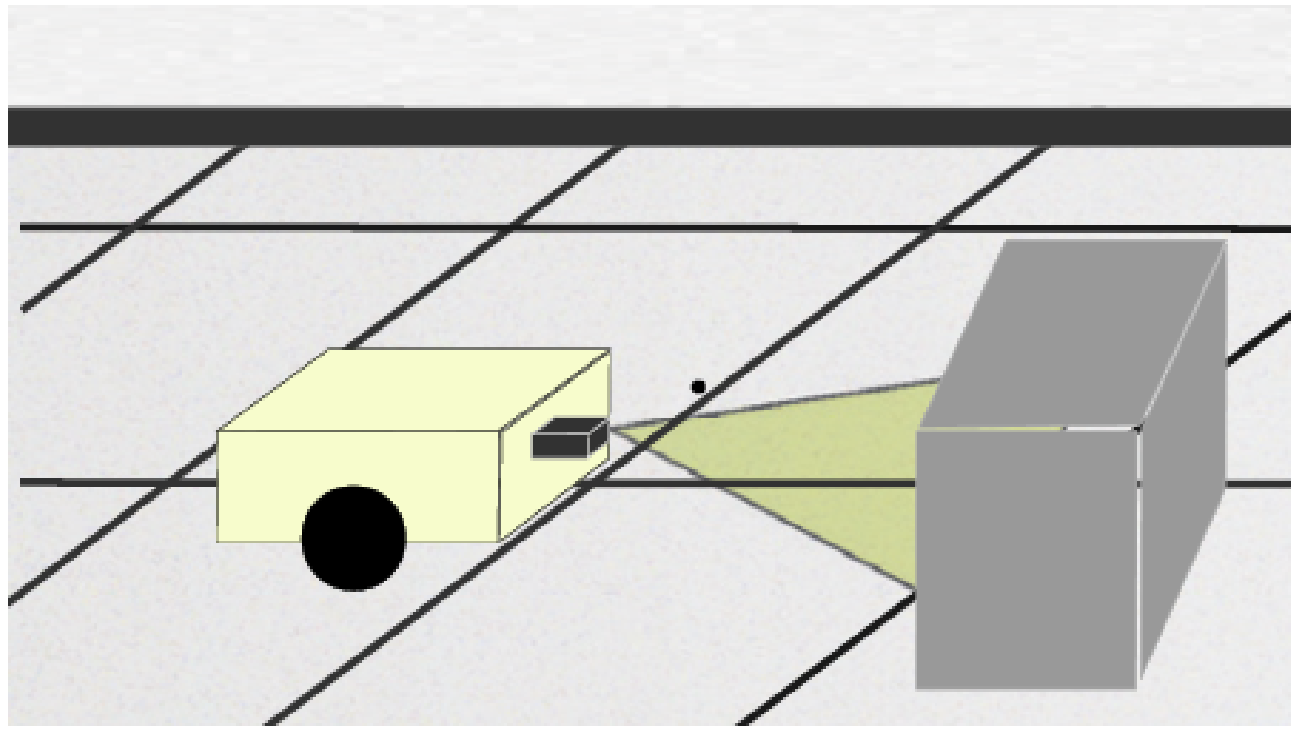
2D Grid Mapping Using Overhead Camera

Piyush Ranjan Satapathy
Kalinga Institute of Industrial Technology
2128080

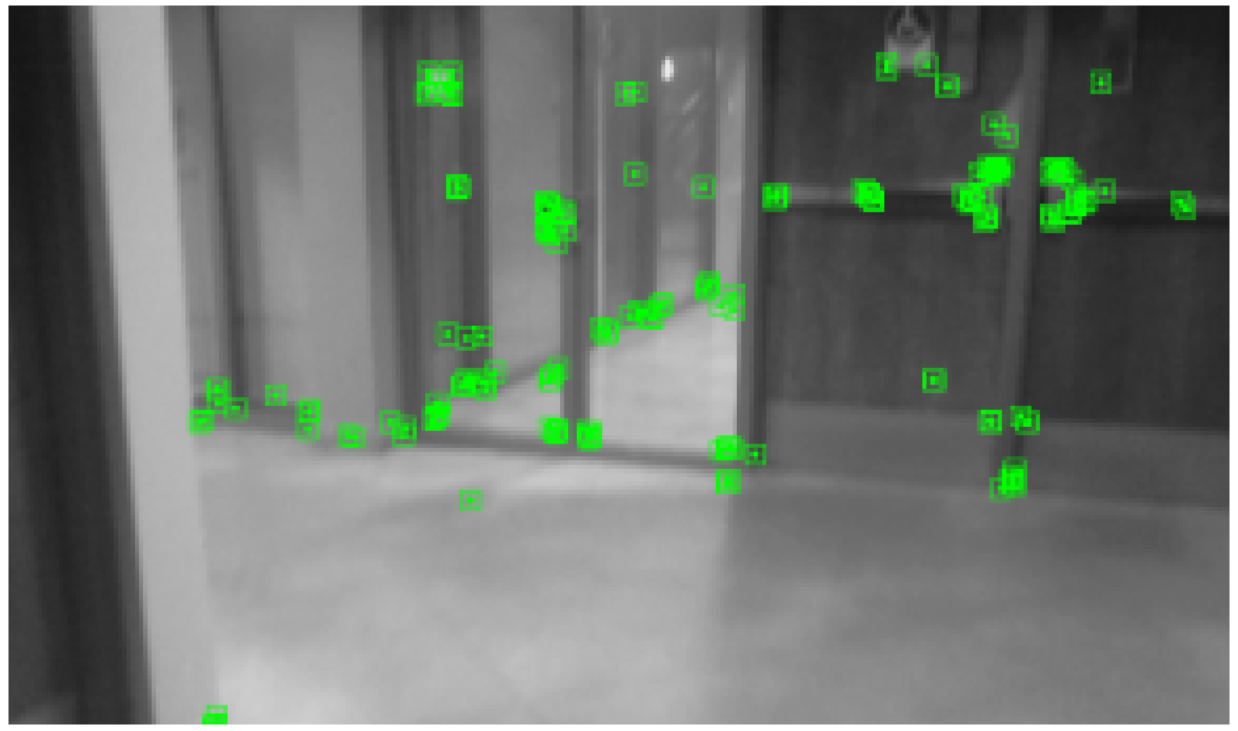
Problem Statement

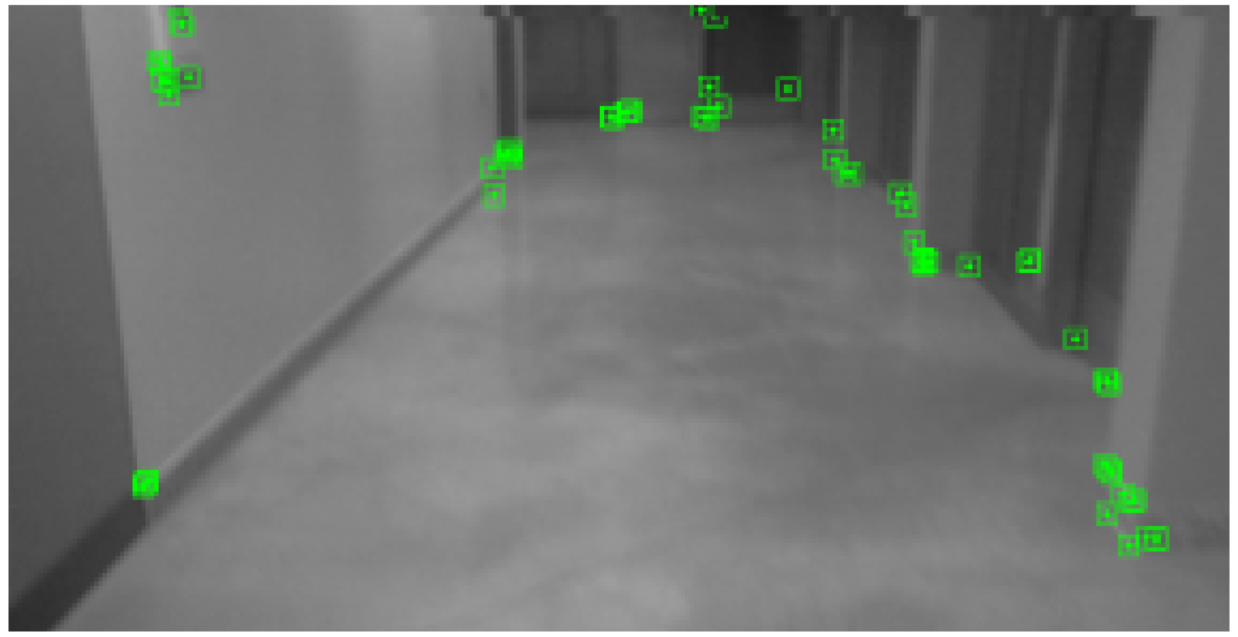
Simultaneous Localization and Mapping (SLAM) is essential for autonomous robots, enabling them to create maps of unknown environments while locating themselves within those maps. Advanced 3D sensors like LiDAR and Kinect offer robust SLAM capabilities but are impractical for low-cost systems and outdoor use due to their high cost and limitations under specific conditions. Visual SLAM, which utilizes 2D cameras, presents a promising yet challenging alternative. Current visual SLAM systems, such as ORB-SLAM, generate sparse point clouds that are insufficient for navigation algorithms that require dense 2D occupancy grid maps. Therefore, there is a pressing need to convert these 3D point clouds into usable 2D grid maps in real-time, facilitating effective navigation.



Unique Idea Brief (Solution)

This project aims to bridge the gap between the sparse point clouds generated by ORB-SLAM and the dense 2D occupancy grid maps required for autonomous navigation. By leveraging novel algorithms and enhancements, this solution enables real-time generation of accurate 2D occupancy grids from ORB-SLAM's 3D data, making it suitable for integration with the ROS navigation stack. This approach not only enhances the navigation capabilities of robots but also maintains the cost-effectiveness and adaptability of visual SLAM systems.





Features Offered

- **Real-time 2D Occupancy Grid Generation:** Converts 3D points from ORB-SLAM into 2D grid maps in real-time. This feature ensures that the robot can dynamically update its map while navigating, facilitating responsive and adaptive navigation.

- **Local and Global Counter Enhancement:** Utilizes local and global counters to maintain the accuracy of the occupancy grid. This prevents misrepresentation of co-linear points, which could otherwise result in erroneous free or occupied cells.

- **Thresholding Techniques:** Visit and height thresholding techniques are applied to filter out outliers and irrelevant floor points. This step improves the reliability of the generated map by ensuring only significant points are considered.

- **Gaussian Smoothing**: Smoothers transitions between free and occupied cells using Gaussian functions. This helps in reducing noise and creating a more accurate representation of the environment, which is crucial for precise navigation.

- **Canny Edge Detection:** Employs Canny edge detection to accurately mark boundaries in the grid map. This technique enhances the delineation of obstacles and free spaces, aiding in the robot’s path planning and obstacle avoidance.

- **Slope Thresholding:** Differentiates between horizontal planes (e.g., floors) and vertical obstacles by evaluating the slope. This feature helps in correctly identifying navigable areas and obstacles, enhancing the map's accuracy.

- **ROS Integration:** Integrates with the Robot Operating System (ROS) to visualize the generated map and support navigation. ROS nodes handle the data communication and processing, while Rviz is used for real-time visualization of the map and robot’s movements.

- **Probabilistic Modeling:** Uses the robot’s pose, pose uncertainties, and a homographic matrix for grid mapping. This probabilistic approach helps in accurately representing the environment despite uncertainties in the robot’s position and sensor measurements.

- **Image Segmentation:** Segments the image into "floor" and "non-floor" regions. This classification is critical for distinguishing navigable areas from potential obstacles in the environment.

- **Reclassification and Mapping:** Reclassifies non-floor cells as obstacles or occlusive regions and maps them using the homography matrix. This step refines the initial segmentation, ensuring that the map accurately represents obstacles and free spaces.

- **Pose Uncertainty Handling:** Expands obstacle cells based on the robot’s pose uncertainties. By incorporating the variances of the robot’s position, this feature ensures that the map remains reliable even when the robot's exact location is not perfectly known.

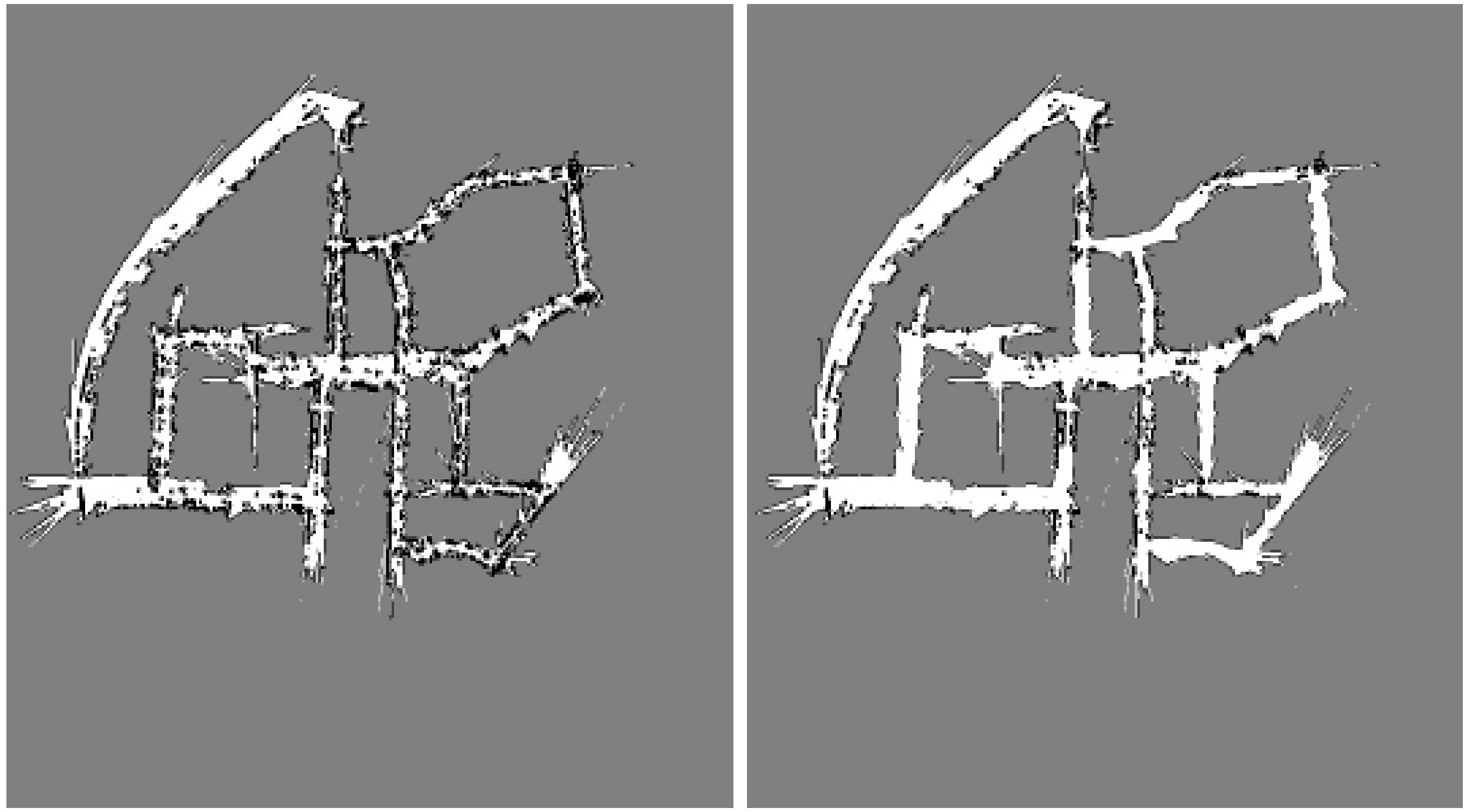


Fig: Effect of height thresholding on KITTI 00 sequence. In the left image, height thresholding has been disabled while the right one shows the result in the presence of the height thresholding

Process Flow

- **ORB-SLAM Setup:** Install and configure ORB-SLAM on the robot. Calibrate the camera to ensure accurate data capture.
- **Data Reproduction:** Reproduce ORB-SLAM results using standard datasets like KITTI and TUM to ensure the system works correctly.
- **2D Grid Map Generation:** Create a Python script to convert ORB-SLAM’s keyframes and map points into a 2D occupancy grid. This involves projecting the 3D points onto a 2D plane.
- **Visualization:** Use ROS nodes to visualize the generated 2D map in Rviz. This helps in real-time monitoring and debugging.
- **Enhancements Implementation:** Apply the local and global counter enhancements, thresholding techniques, Gaussian smoothing, edge detection, and slope thresholding to improve map accuracy and reliability.

- **Evaluation:** Compare the generated map against ground truth data to measure accuracy and completeness. Adjust parameters as needed to optimize performance.

- **Homography Matrix Utilization:** Calculate the homography matrix at the start of the exploration. Use this matrix to map image lines directly to the world coordinates.

- **Segmentation and Classification:** Segment the image into "floor" (free space) and "non-floor" (obstacles). Use this classification to update the occupancy grid.

- **Reclassification:** Reclassify non-floor cells as obstacles or occlusive regions. Map these cells using the homography matrix.

- **Obstacle Expansion:** Expand the obstacle cells based on the robot’s pose uncertainties to account for potential errors in the robot’s position**.**

- **Real-time Processing:** Convert the Python prototype into a C++ implementation for real-time performance. Create ROS nodes for efficient data handling.

Architecture

ORB-SLAM System

- **Input:** Monocular camera captures the environment.
- **Process:** ORB-SLAM performs feature extraction, matching, pose estimation, and map point generation.

2D Grid Map Generation

- **Projection:** Keyframes and map points from ORB-SLAM are projected onto the XZ plane to create a 2D grid.
- **Enhancements:** Apply local/global counters, thresholding, Gaussian smoothing, edge detection, and slope thresholding to the projected points.

Occupancy Grid Modeling

- **Matrix Calculation:** Calculate the homography matrix for accurate mapping of image lines to world coordinates.
- **Segmentation:** Segment the image into "floor" and "non-floor" regions.
- **Reclassification:** Reclassify non-floor cells as obstacles or occlusive regions.
- **Mapping:** Use the homography matrix to map cells to the world coordinates. Expand obstacle cells based on pose uncertainties.

Occupancy Grid Modeling

- **Matrix Calculation:** Calculate the homography matrix for accurate mapping of image lines to world coordinates.
- **Segmentation:** Segment the image into "floor" and "non-floor" regions.
- **Reclassification:** Reclassify non-floor cells as obstacles or occlusive regions.
- **Mapping:** Use the homography matrix to map cells to the world coordinates. Expand obstacle cells based on pose uncertainties**.**

ROS Integration

- **Nodes:** Create ROS nodes (monopub and monosub) for publishing and subscribing to map data.
- **Visualization:** Use Rviz and Gazebo for real-time visualization of the map and robot navigation.

Probabilistic Modeling

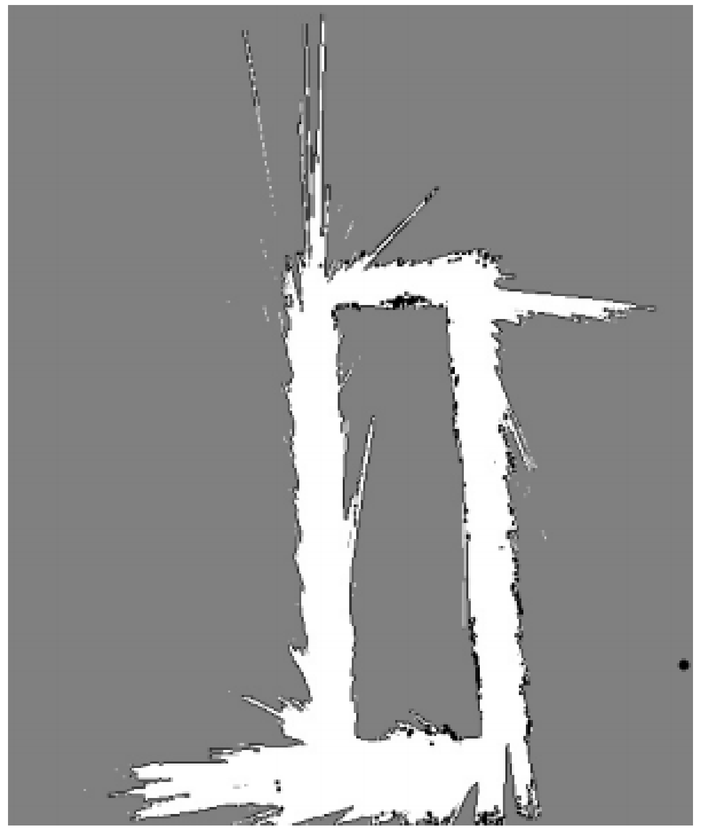
- **Sensor Modeling:** Model the camera as a distance sensor using a Gaussian function to calculate cell occupation probabilities.
- **Variance Calculation:** Calculate the sensor's variance behavior to account for measurement inaccuracies. Use experimentally determined constants to refine this model.

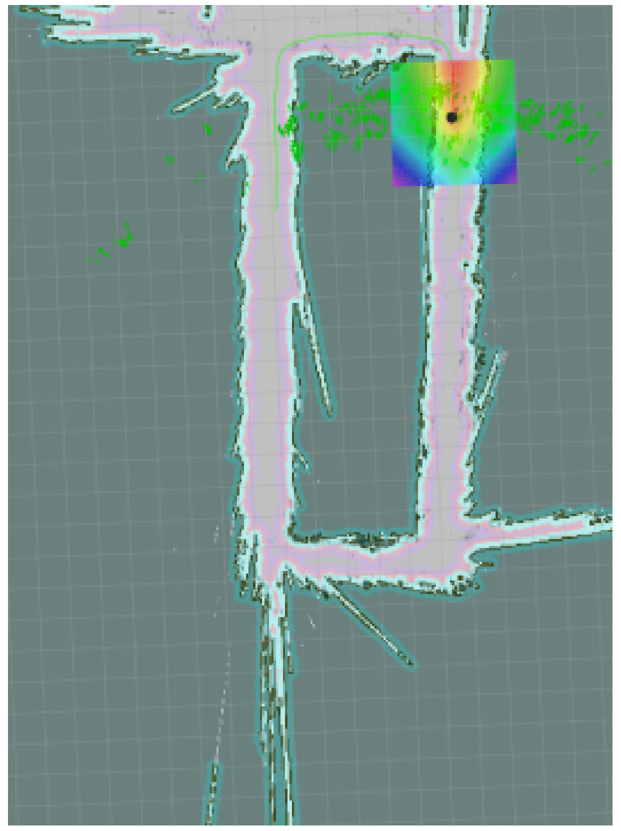
Technology Used

- **SLAM System:** ORB-SLAM2 for visual SLAM.
- **Programming Languages:** Python for prototyping, C++ for real-time processing
- **Robotics Middleware:** Robot Operating System (ROS) for integration and visualization.
- **Camera:** Logitech C615 1080p webcam for image capture.
- **Datasets:** KITTI and TUM for SLAM performance evaluation.
- **Algorithms:** Bresenham's line drawing, Canny edge detection, Gaussian smoothing, and slope thresholding for map generation.
- **Visualization Tools:** Rviz and Gazebo for map and navigation visualization.

Conclusion

This project demonstrates the effective generation of a 2D occupancy grid map in real-time from ORB-SLAM’s 3D point clouds. Through a series of enhancements, including local and global counters, thresholding techniques, Gaussian smoothing, and edge detection, the system achieves a high level of map accuracy and usability for navigation. By incorporating probabilistic modeling using pose uncertainties and homography, the generated maps are reliable for real-world applications, significantly enhancing autonomous navigation capabilities using visual SLAM data.





Thank You