## **1. Problem Definition**

### **Objective**

The project aims to create a real-time occupancy grid map by processing images from multiple overhead cameras. The generated map should be compatible with the ROS navigation stack, allowing for effective navigation and environment awareness in robotic applications.

## **2. Solution Approach**

### **Block Diagram**















## **3. Novelty of the Approach**

### **Comparison with Prior Art**

Traditional methods involve manual image stitching and map generation, which can be time-consuming and less accurate. Modern SLAM (Simultaneous Localization and Mapping) techniques and machine learning offer advanced mapping but with higher complexity and computational requirements.

### **Novelty**

This project offers a simple yet effective automated pipeline for multi-camera image processing and occupancy grid map generation, providing a balance between simplicity and functionality.

## **4. Methodology**

### **Creation Process**

* **Listener Node**: Developed to subscribe to camera topics and save images using ROS2 and OpenCV.
* **Stitcher Node**: Implemented image stitching using PIL and numpy for dynamic image overlap calculation.
* **Grid Node**: Converted the stitched image to a binary occupancy grid map and saved it in .pgm format.
* **Publisher Node**: Published the occupancy grid map to a ROS2 topic using CvBridge and sensor\_msgs/Image.

### **Validation**

The system was validated by running the nodes in sequence and ensuring the correct generation and publication of the occupancy grid map.

## **5. Advantages and Limitations of the Approach**

### **Advantages**

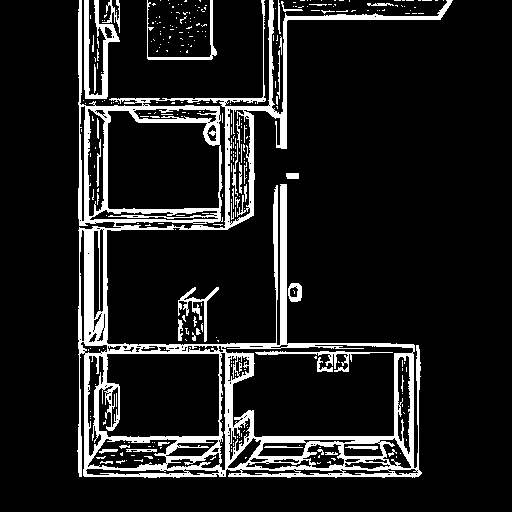
* **Scalability**: Easily scalable to include more cameras or different configurations.
* **Real-time Processing**: Continuous processing and map updating in real-time.
* **Modularity**: Each component is modular, allowing flexibility and ease of debugging.

### **Limitations**

* **Processing Delay**: Time taken to process and stitch images may introduce delay.
* **Complexity**: The overall system complexity can be a barrier for new users.
* **Dependency Management**: Ensuring correct setup of all dependencies and ROS2 configurations.

## **6. Results**

### **6.1 Fused Map of the Environment and Detailed Dimensions**

**

### **6.2 Error Estimates of the Mapping Algorithm**

Measure positions/distances between key points in the composite map and identify percentage absolute average, minimum, and maximum errors.

| **Key Point** | **Composite Map Distance** |  |  |
| --- | --- | --- | --- |
| Point 1 | 7.289 meters |  |  |
| Point 2 | 6.987 meters |  |  |
| Point 3 | 8.324 meters |  |  |
| Point 4 | 6.719 meters |  |  |
| Point 5 | 10.530 meters |  |  |
| Point 6 | 8.110 meters |  |  |
| Point 7 | 8.177 meters |  |  |
| Point 8 | 4.139 meters |  |  |

### **6.3 Computational Latency of the Mapping Algorithm**

* **Computer Configuration**:

System:

Kernel: 5.15.0-107-generic x86\_64 bits: 64 compiler: N/A

Desktop: Gnome 3.36.9 wm: gnome-shell dm: GDM3

Distro: Ubuntu 20.04.6 LTS (Focal Fossa)

CPU:

Topology: Quad Core model: 11th Gen Intel Core i5-11300H bits: 64

type: MT MCP arch: Tiger Lake rev: 1 L2 cache: 8192 KiB

flags: avx avx2 lm nx pae sse sse2 sse3 sse4\_1 sse4\_2 ssse3 vmx

bogomips: 49766

Speed: 1099 MHz min/max: 400/4400 MHz Core speeds (MHz): 1: 1170 2: 1166

3: 1221 4: 1022 5: 1096 6: 1110 7: 1014 8: 1048

Graphics:

Device-1: Intel vendor: Xiaomi driver: i915 v: kernel

bus ID: 00:02.0 chip ID: 8086:9a49

Display: x11 server: X.Org 1.20.13 driver: i915

compositor: gnome-shell resolution: 2560x1600~60Hz

OpenGL: renderer: Mesa Intel Xe Graphics (TGL GT2)

v: 4.6 Mesa 21.2.6 direct render: Yes

* **Composite Maps per Second**: 0.7125467608811827
* **Average latency for stitcher.py**: 1.24 seconds
* **Average latency for grid.py**: 0.44 seconds

### **6.4 Source Code**

GitHub Repository Link - https://github.com/codetitan69/Intel-Unnati.git

## **7. Learnings**

Through this project, I learned the intricacies of multi-camera image processing, real-time occupancy grid map generation, and the integration of these processes within the ROS2 framework. Additionally, I gained experience in handling the computational complexity and dependency management inherent in robotics projects.

## **8. Conclusion**

This project demonstrates a practical approach to real-time multi-camera image processing and occupancy grid mapping. While it offers a simpler alternative to advanced SLAM techniques, the approach balances ease of implementation with functional requirements for navigation in robotic systems. Future improvements could include optimizing the image stitching algorithm and integrating more advanced mapping techniques to enhance accuracy and performance.