# Problem Definition

Create a 2-D occupancy map of the given simulation (indoor environment), using the mounted over-head RGB cameras at the specified positions. The map is to be used for tasks such as AMR navigation.

# Solution Approach

Describe your solution with the help of a block diagram / figure

# Novelty of the approach

Custom trail of edge and blob detections along with image transformations to produce a stitched image of the given environment

# Methodology

The first step was to set up the environment and load the overhead cameras into the simulation. Then the variables have been defined. The proper stitching order is calculated such that all the images on the same vertical axis are first stitched and then the horizontal stitching is done. The proper overlap percentages are obtained and used to ensure the proper structure of the final image. Then an adaptive thresholding is ran on the final image in order to get a binary inversed image. Then the pgm and the yaml files are generated.

# Describe advantages and limitations of the approach

This approach ensures fast compute time, which translates to faster responses to changes in the target environment. The use of simple algorithms for stitching ensure the scalability and consistency during the reproduction of the program.

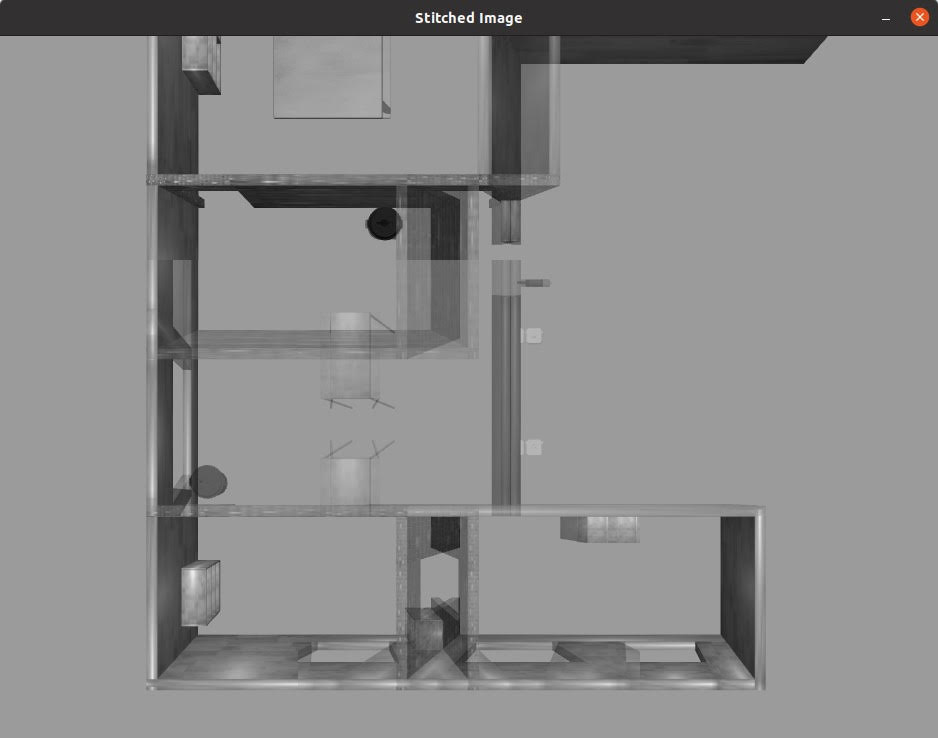
Since the environment is a simulation, were unable to employ advanced object detection and mapping strategies that involve images and data that is prevalent in an actual environment.

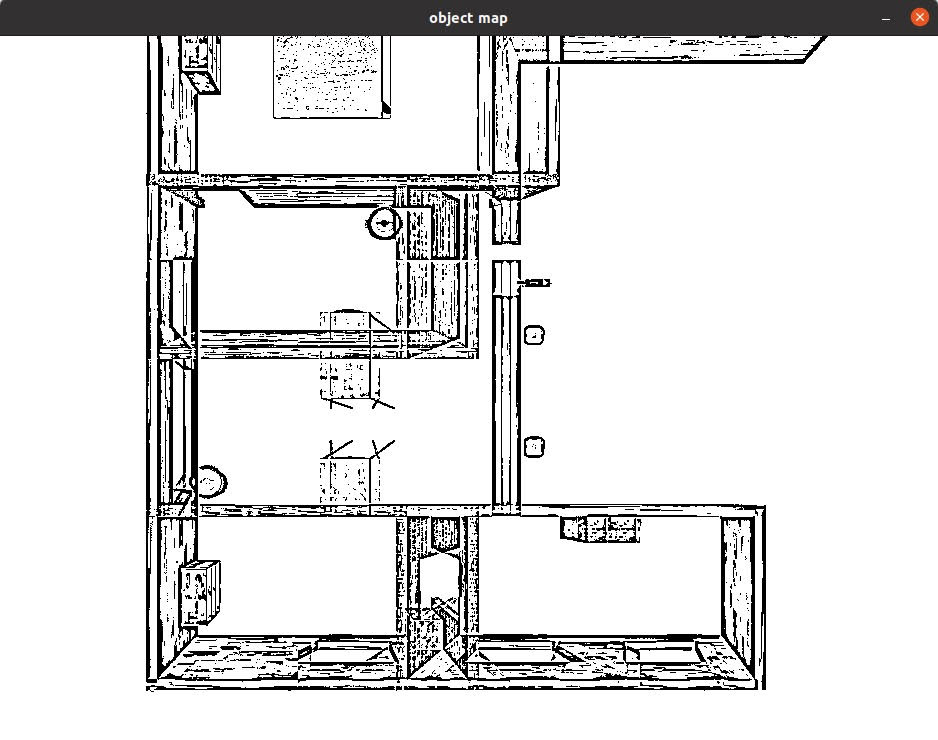
The perspective of the cameras were a hinderance to make a proper stich, as the overlapping calculations were affected by it.

# Results

The object map and pgm files have been generated and has been proven to be the best of out multiple approaches like: Edge detection (Canny, log, Sobel filter), Blob detection, Midas to detect close objects, YOLOv5 for single detection, Feature Extraction with common overlap, Custom algo using OpenCV to calculate common overlap.

## Fused map of the environment and detailed dimensions



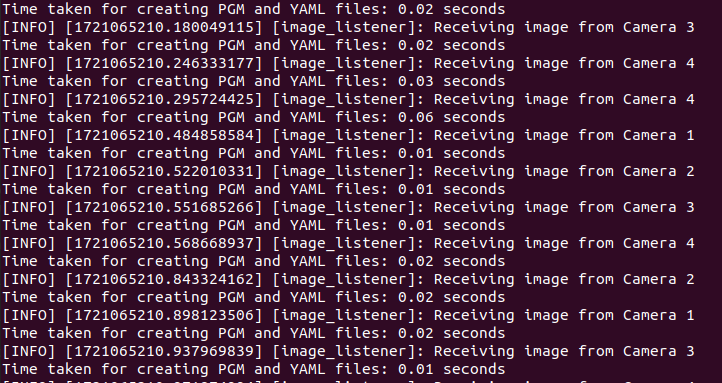


## Error estimates of the mapping algorithm

Rviz error prevented loading of ground values and evaluation of the generated pgm file.

## Computational Latency of the mapping algorithm

The computational latency is between 0.01ms -0.06ms



## Source Code

Share links to the source code repository and provide appropriate access.

# Learnings

We were introduced to Ross and Gazebo. We were also able to explore further into different algorithms and methods of using computer vision on multiple images with different levels of overlap and dissimilar edges.

We made extensive use of OpenCV for testing, manipulation and generation of images.

# Conclusion

From setting up VirtualBox running Ubuntu 20.04 and installing Ross-Foxy to generating the pgm and yaml files of the stitched image lead to us learning ROSS and Gazebo in great depth and further peaked our interests in the field of robotics and computer vision.

We thank the Unnati Team at Intel for this learning opportunity and would love to be included in further projects.

Note:

1. **Accuracy** (absolute avg, min, max error) of generated fused map v/s distances of key-point in simulation. **The expectation is that the avg error should be < 10% (lesser the better)**.
   * The algo and code will be tested on a modified map to verify accuracy of the algorithm
2. **Computational Complexity** (Computational latency in milliseconds to process each set of images from camera and convert them to a composite map) as measured on an Intel Core i5 (10th gen CPU, iGPU can be used). **The expectation it that the overall computational latency should be < 1000ms (lesser the better).**
3. **Novelty, practicality and efficiency of the solution** – this is a subjective metric

**Weightage will be given to the above criteria**