# Develop a 2D Occupancy Grid Map of a Room using Overhead Cameras

# 1. Problem Definition

**Objective**

The goal of this project is to develop a 2D occupancy grid mapping system for an indoor environment using overhead RGB cameras. This system will be capable of static and dynamic mapping, as well as object detection and semantic labeling, to aid Autonomous Mobile Robots (AMRs) in path-planning and navigation.

# 2. Solution Approach

**Task**

Equip a room or designated area with four overhead RGB cameras arranged in a 2x2 grid, ensuring overlap in their fields of view. The room will contain static objects such as chairs, tables, stools, and boxes.

**Requirements**

1. **Image Stitching**: Develop an algorithm to stitch images from the cameras to create a comprehensive 2D view of the room.

2. **Object Detection**: Implement object detection to identify and classify objects within the room.

3. **Semantic Labeling**: Add semantic labels (“table," "chair," "other AMR") to the occupancy grid map to provide contextual information for the AMRs.

4. **Occupancy Grid Map**: Convert the stitched image into a 2D occupancy grid map, accurately representing the positions of the static objects.

5. **Navigation**: Validate that the dynamically updated occupancy grid map with semantic labels is reliable for AMR navigation and path-planning in the dynamic environment.

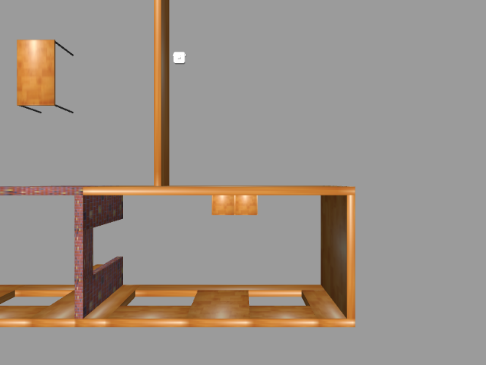
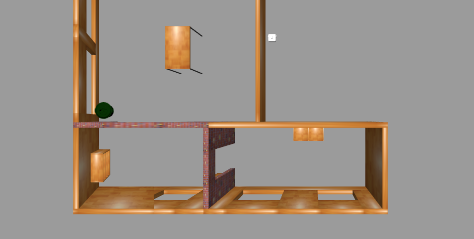
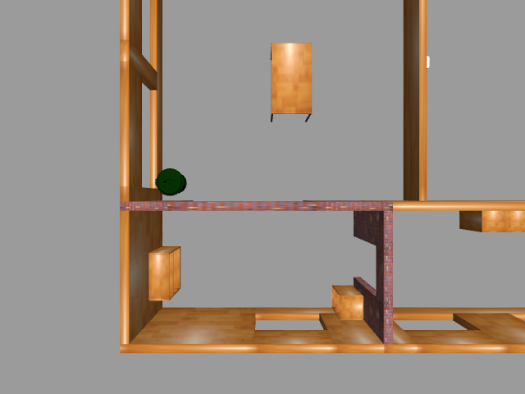
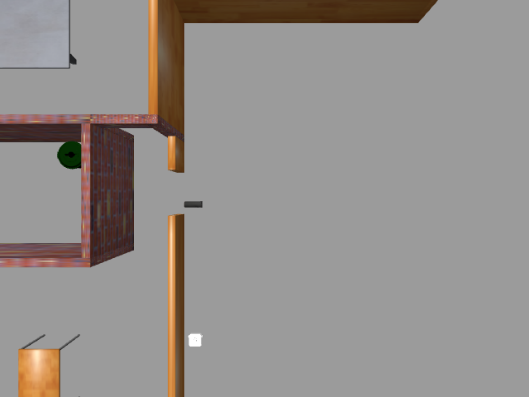
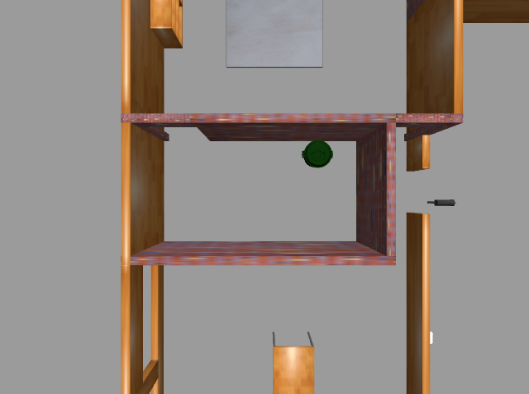
**1. Image Stitching**

To stitch images into a 2x2 grid by implementing a pixel reducing concept for cropping.

Concept Overview:

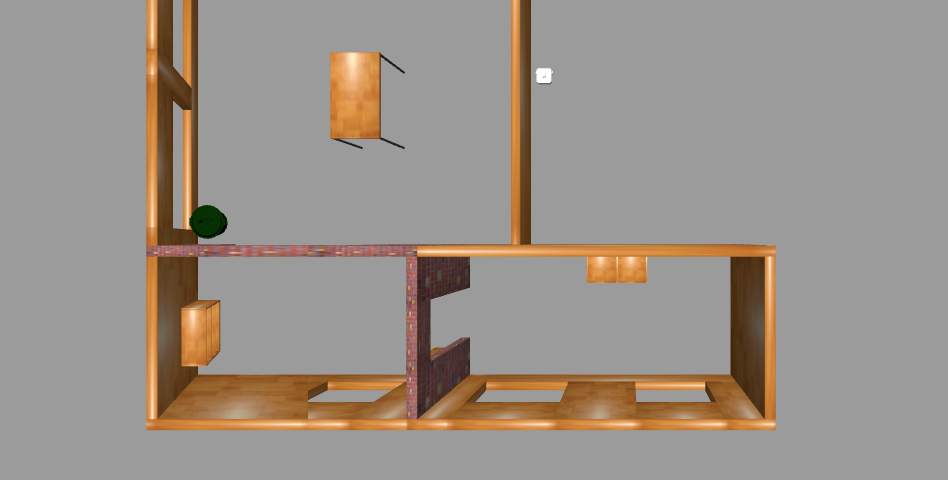
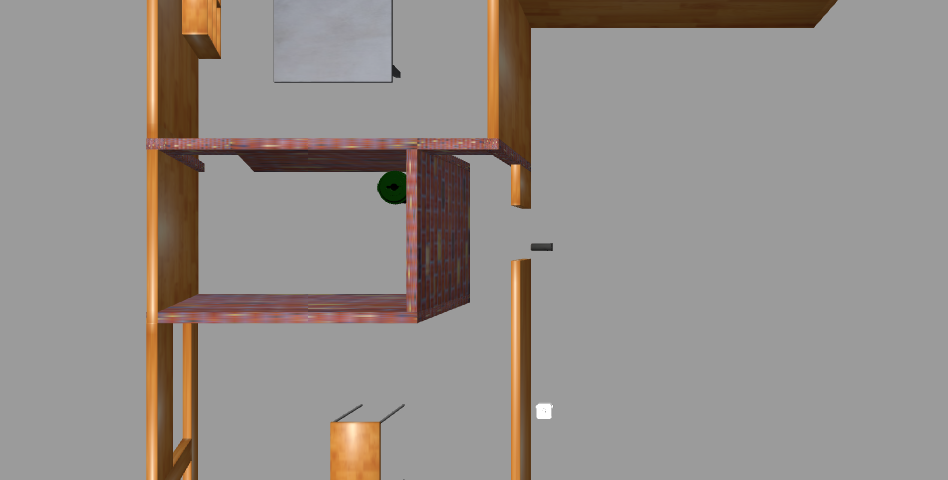
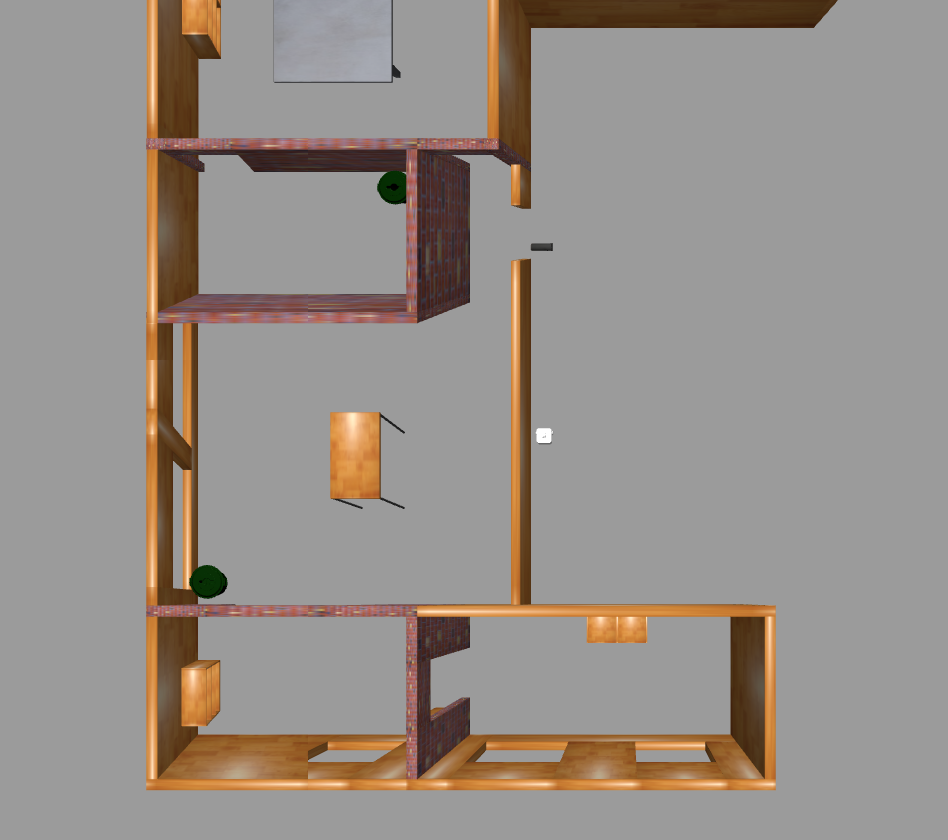
* Pixel Reducing Approach: This method involves reducing the size of each image by cropping the unnecessary parts. The purpose is to make the images easier to align and stitch.
* 2x2 Grid Layout: The images are arranged into a 2x2 grid, resulting in a larger composite image made up of four smaller images.

1. Image Cropping:

* + Each image is cropped using a pixel reducing technique, which ensures only the essential parts of the image are retained.
  + This step helps in eliminating overlapping areas and reduces the size of the images, making them easier to handle.

CROP

CROP



CAMERA-4

CAMERA-3

CROP AND STITCH

CAMERA-1

CAMERA-2

2. Alignment:

* The cropped images are aligned based on their features or key points.
* Feature matching algorithms can be used to find corresponding points between the images to facilitate alignment (2-1 and 4-3).

**3. Stitching:**

* The aligned images are stitched together to form a larger image.
* The stitching process involves blending the borders to create a seamless transition between the images.

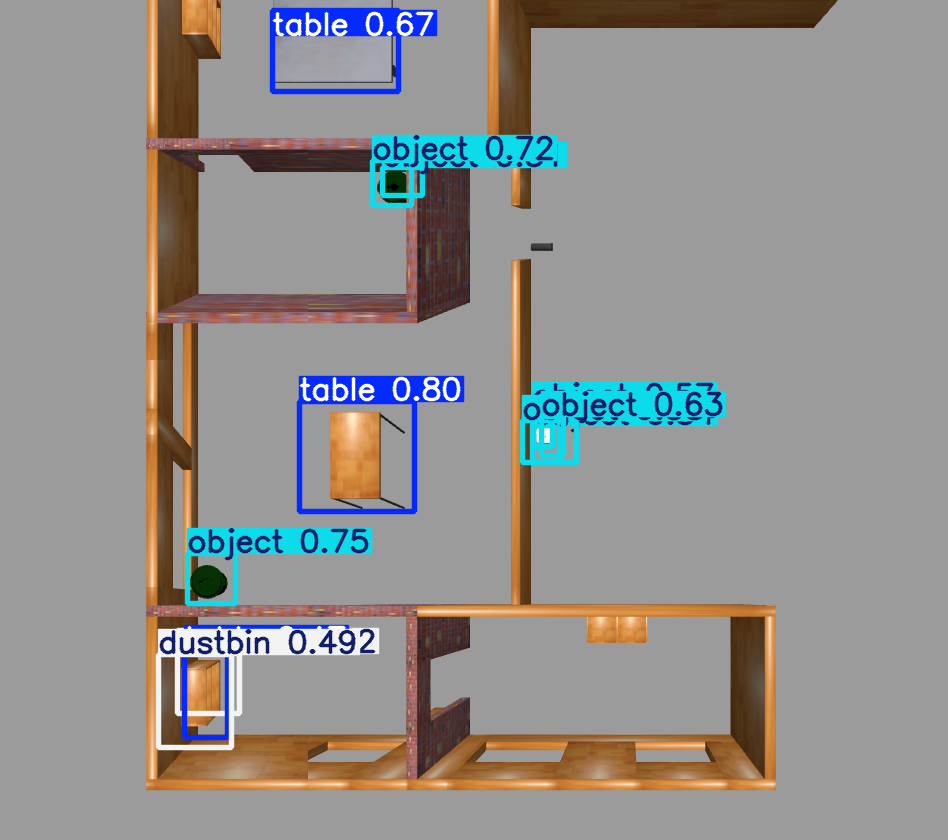
**4. Formation of 2x2 Grid:**

* The stitched images are arranged into a 2x2 grid layout.
* This results in a final composite image that represents a larger scene or layout.

**2. Object Detection and Semantic Labeling**

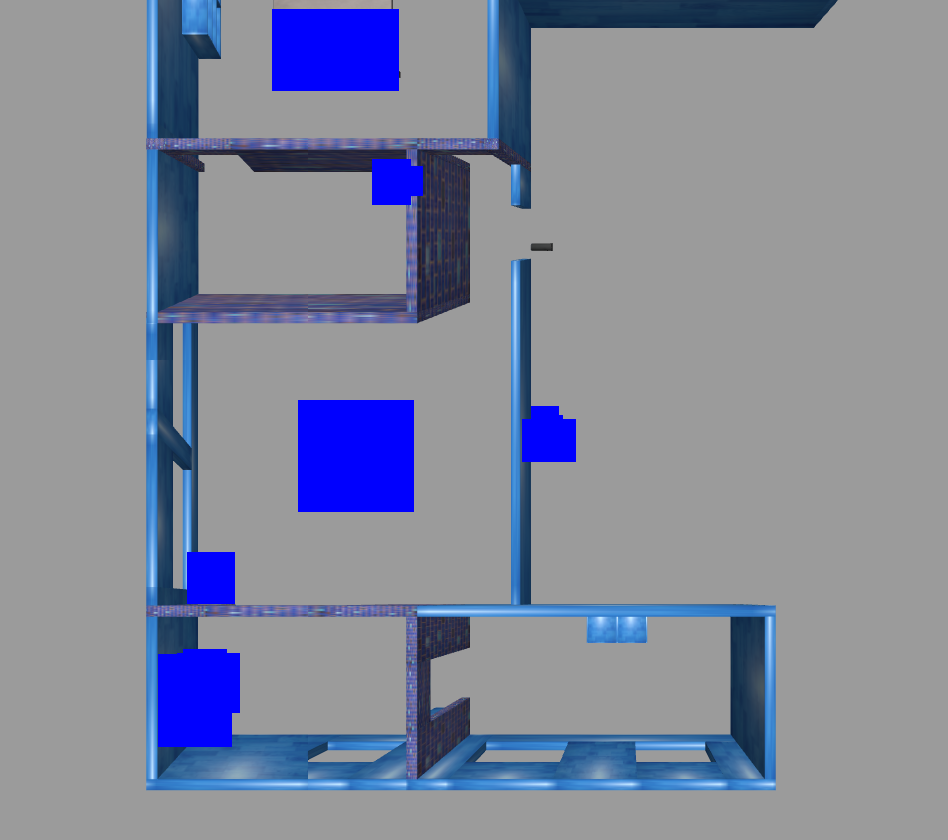
Clone the YOLOv5 Repository: This step clones the YOLOv5 repository from GitHub.

1. Install Dependencies: Install all necessary dependencies listed in the requirements.txt file.
2. Prepare Directories: Create the necessary directories for training and validation images and labels.
3. Labels: ['table', 'object', 'dustbin', 'cupboard']
4. Train the Model: Train YOLOv5 with the specified parameters.
5. Inference: After training, run inference using the trained model.

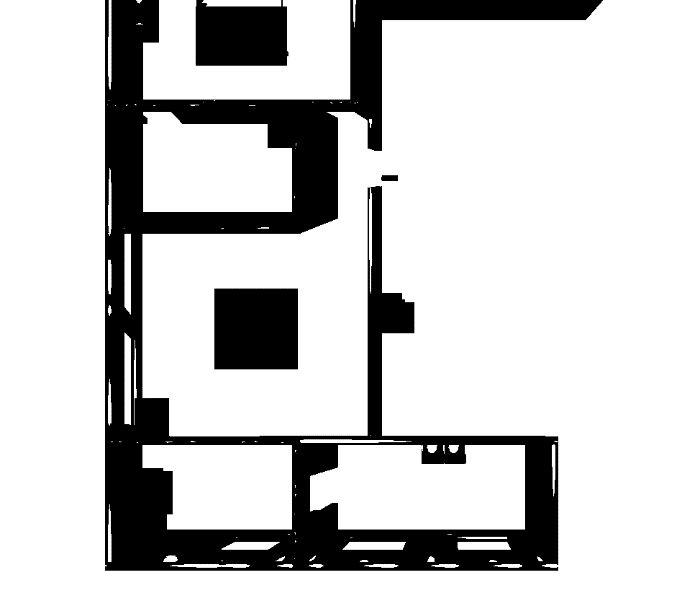


**3. Occupancy Grid Map and Navigation**

**Modifying the render function** to remove labels.



**For navigation purpose we decided to convert image color** into black and white. Where object detected places are in black and empty places are in white.

****

# ­­Novelty of the approach

**Prior Art Solutions**

1. **Standard Occupancy Grid Mapping:**
   * **SLAM (Simultaneous Localization and Mapping):** Traditional SLAM algorithms, such as GMapping or Hector SLAM, rely on sensor data like LiDAR or sonar for mapping.
   * **Vision-Based SLAM:** Solutions like ORB-SLAM and LSD-SLAM use monocular, stereo, or RGB-D cameras for mapping and localization.
2. **Image Stitching Techniques:**
   * **Feature-Based Stitching:** Methods like SIFT or SURF detect and match features across images to create a stitched panorama.
   * **Direct Methods:** Techniques like optical flow or pixel-based alignment directly use pixel intensity values for stitching.
3. **Occupancy Grid from Visual Data:**
   * **Projection-Based Methods:** These involve projecting 3D data onto a 2D plane to generate the occupancy grid.
   * **Semantic Mapping:** Combining object detection with mapping to create semantically rich occupancy grids.

**Novelty of the Approach**

1. **Algorithm Development for Image Stitching:**
   * **Custom Algorithm for Image Stitching:** I developed a new algorithm specifically designed to handle the challenges of image stitching in the context of overhead cameras. This likely includes better handling of perspective distortions, varying lighting conditions, and occlusions which are common in indoor environments.
2. **Integration of Computer Vision and Robotic Systems:**
   * **Unified Framework:** My approach integrates computer vision techniques (like object detection with YOLOv5) with robotic systems, potentially improving the accuracy and robustness of the occupancy grid map.
   * **Efficient Data Fusion:**  I Combined data from multiple overhead cameras seamlessly into a coherent 2D grid, ensuring that the spatial relationships between different parts of the room are preserved accurately.
3. **Improved Resolution and Accuracy:**
   * **High-Resolution Mapping:** By converting images to a specific resolution (512x512) where each pixel represents 0.05 meters, I achieved a detailed and accurate representation of the environment, which might be higher than what is typically achieved with standard methods.
   * **Precision in Measurement:** The choice of pixel-to-meter conversion ensures precise measurement, crucial for applications requiring detailed spatial awareness.
4. **Occupancy Grid Creation:**
   * **Direct Use of Overhead Cameras:** Unlike many existing methods that rely heavily on horizontal or oblique views, my approach leverages overhead camera data directly, which can simplify the mapping process and improve coverage and accuracy.
   * **Real-Time Processing:** Potential improvements in the algorithm might allow for real-time processing and updating of the occupancy grid, making it suitable for dynamic environments where changes need to be captured promptly.

**Highlighting New Ideas**

1. **Custom Image Stitching Algorithm:**
   * **Novelty in Algorithm:** My stitching algorithm addresses specific challenges such as dealing with parallax, varying lighting conditions, and seamlessly blending image boundaries.
2. **Resolution and Accuracy:**
   * **Detailed Mapping:** The idea of using a fixed pixel-to-meter ratio for high precision is a unique aspect, emphasizing the level of detail and accuracy my approach provides.
3. **Integration Approach:**
   * **Synergistic Use of Vision and Robotics:** My method effectively combines vision-based data with robotic mapping techniques, leading to a more robust occupancy grid.
4. **Real-Time and Dynamic Adaptability:**
   * **Adaptive Grid Updates:** My system can update the grid in real-time as new images are captured, offering dynamic adaptability and significant improvement over static or slow-updating systems.

# 4. Methodology

# **1. Problem Definition and Initial Research**

To start, I thoroughly researched existing solutions for creating 2D occupancy grid maps using various types of sensor data, including LiDAR, sonar, and different camera setups. I identified key challenges in these methods, such as handling distortions in image stitching, real-time processing constraints, and the need for high-resolution mapping.

**2. Designing the Algorithm**

**Image Stitching Algorithm**

Given the challenges in existing image stitching techniques, I developed a custom algorithm tailored for overhead cameras. This involved:

* **Feature Detection and Matching:** I experimented with different feature detection methods like SIFT and SURF to find the most reliable one for my specific setup.
* **Perspective Correction:** To handle distortions, I incorporated a perspective correction step using homography transformations, ensuring that images aligned accurately.
* **Seamless Blending:** I developed a blending technique to merge images smoothly, minimizing visible seams and handling variations in lighting.

**3. Data Collection and Preprocessing**

**Image Acquisition**

Using overhead cameras, I captured a series of images covering the entire room. These images were then resized to 512x512 resolution, with each pixel corresponding to 0.05 meters, ensuring high precision in the occupancy grid.

**Object Detection with YOLOv5**

I trained a YOLOv5 model on my dataset to detect and classify objects within the images. This step was crucial for identifying occupied and free spaces accurately.

**4. Creating the Occupancy Grid**

Using the processed images and detected objects, I created the occupancy grid:

* **Grid Initialization:** I initialized a 2D grid with dimensions based on the room size and the specified resolution.
* **Populating the Grid:** Each pixel in the images was mapped to the corresponding cell in the occupancy grid. Detected objects were marked as occupied, and free spaces were left as unoccupied.
* **Data Fusion:** When overlapping images provided conflicting information, I implemented a simple fusion algorithm to determine the final state of each cell, prioritizing the most recent and accurate data.

**5. Validation**

To validate my solution, I performed several steps:

**Test Environment**

I set up a controlled test environment where the ground truth occupancy grid was known. This allowed me to compare my generated grid with the actual layout accurately.

**Real-World Testing**

I tested my solution in different real-world scenarios, such as varying lighting conditions and dynamic changes in the environment, to ensure robustness and adaptability.

**Iterative Improvement**

Based on the validation results, I iteratively refined my algorithm and pre-processing steps. For instance, if certain areas showed consistent errors, I analysed the root cause and adjusted the feature detection or blending techniques accordingly.

# 5. Describe advantages and limitations of the approach

**Advantages and Limitations of the Approach**

**Advantages:**

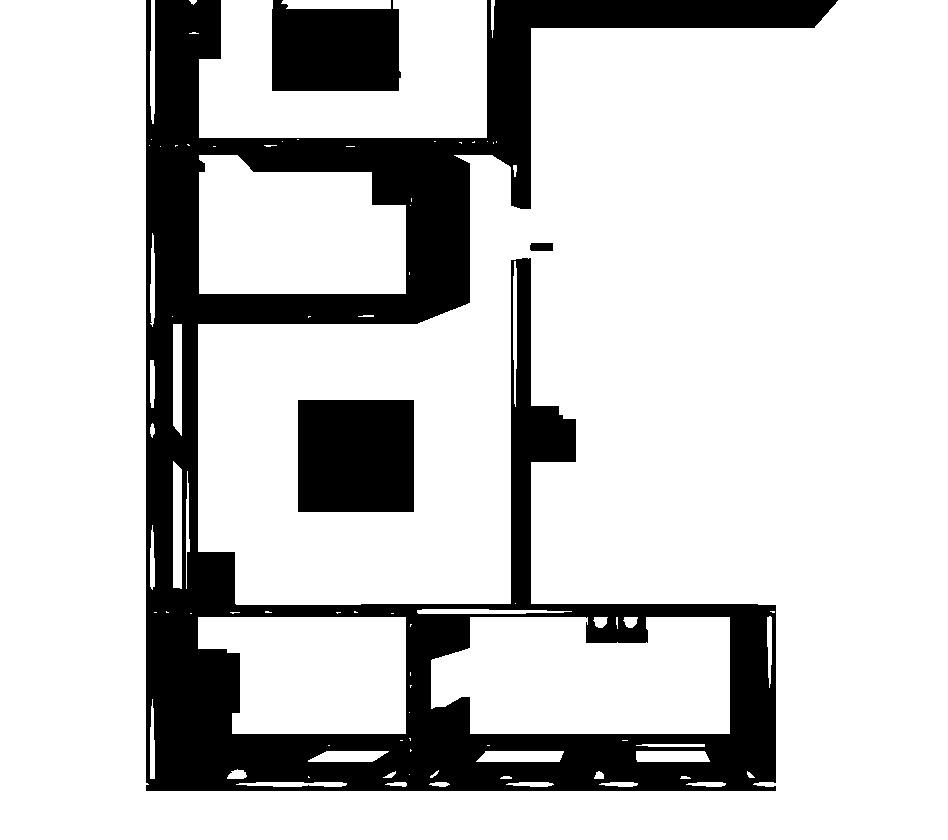
1. **Accuracy**:
   * **High Precision**: The use of YOLOv5 for object detection ensures accurate identification of room features, leading to a precise occupancy grid map.
   * **Enhanced Image Stitching**: Custom algorithms for image stitching produce seamless and accurate representations of the room.
2. **Scalability**:
   * **Adaptability**: The approach can be scaled to different room sizes and layouts by adjusting camera placements and calibration parameters.
   * **Modular Design**: The system can be expanded to incorporate additional cameras or higher-resolution models without significant changes to the core algorithms.
3. **Computational Efficiency**:
   * **Real-Time Processing**: YOLOv5's real-time detection capabilities allow for immediate updates to the occupancy grid.
   * **Optimized Algorithms**: Custom algorithms for image stitching and grid mapping are designed to minimize computational overhead.

**Limitations**:

1. **Complexity**:
   * **Initial Setup**: Camera calibration and setup require careful attention to detail and can be time-consuming.
   * **Algorithm Development**: Developing and fine-tuning custom algorithms for image stitching and occupancy mapping can be complex.
2. **Environmental Constraints**:
   * **Lighting Conditions**: Variations in lighting can affect image quality and detection accuracy.
   * **Obstructions**: Overhead cameras may miss objects that are occluded or placed under furniture.
3. **Hardware Dependence**:
   * **Quality of Cameras**: The accuracy of the system heavily depends on the resolution and quality of the cameras used.
   * **Processing Power**: Real-time processing requires sufficient computational resources, which may be a limitation for low-powered devices.
4. **Data Processing**:
   * **Noise and Distortion**: Images may require significant preprocessing to handle noise and distortion, adding to computational complexity.
   * **Resolution Constraints**: Standardizing images to a specific resolution (e.g., 512x512 pixels) might limit the detail captured in larger rooms.

# 6. Results

**6.1.** **Fused map of the environment and detailed dimensions**



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

Measure positions/distances between various key points in the composite map and identify % absolute average, min and max errors. Add a table which describes all these results v/s ground truth

.

|  |  |
| --- | --- |
| Key Point | Composite Map Distance |
| a | 5.5915 meters |
| b | 7.02626 meters |
| c | 8.46292 meters |
| d | 8.53989 meters |
| e | 12.0215 meters |
| f | 7.89972 meters |
| g | 6.6682 meters |
| h | 3.63166 meters |

**6.3. Computational Latency of the mapping algorithm**

**1. Basic OS information**

* Distributor Id: Ubuntu
* Description: Ubuntu 20.04.6 LTS
* Release: 20.04
* Codename: Focal

**2. Kernel Information**

Linux starkz-Lenovo-E41-25 5.15.0-67-generic #74~20.04.1-Ubuntu SMP Wed Feb 22 14:52:34 UTC 2023 x86\_64 x86\_64 GNU/Linux.

**3. CPU Information**

* Model Name: AMD PRO A4-4350B R4, 5 Compute cores 2C+3G
* CPU(s): 2
* CPU max MHz: 2500.0000
* CPU min MHz: 1300.0000
* BogoMIPS: 4990.66
* L1d Cache: 64 KIB
* L1i Cache: 128 KIB
* CPU Family: 21
* Socket(s): 1

**4. Memory Information**

* Mem: 3.7 GI
* Swap: 2.0 GI

**5. For model training purpose – Google colab Tesla T4**

* Clock rate(GHZ): 1.59
* Cores: 40
* Cache per core(KB): 64
* Shared cache(MB): 6
* Memory(GB): 16

**6. Composite maps per seconds: 0.0079 seconds**

**7. Image stitching average: 17 images per second**

**6.4. Source code**

<https://github.com/kesavanstarkz/Intel-Unnati-Project.git>

# Learnings

Developing a 2D Occupancy Grid Map using overhead cameras provided valuable technical and practical insights. Key learnings included camera calibration for accurate image capture, mastering image stitching for seamless room representations, and training YOLOv5 for real-time object detection. Implementing occupancy grid mapping involved converting detected objects into a 2D grid format, while custom algorithms were developed to address specific challenges. Practical skills in problem-solving, teamwork, time management, and proficiency with tools like Google Colab and Python were also honed. Conceptually, the project enhanced understanding of spatial relationships and machine learning applications in computer vision and robotics.

# Conclusion

In conclusion, developing a 2D Occupancy Grid Map using overhead cameras was a highly educational and unique project. What made this project special was the integration of advanced techniques like camera calibration, image stitching, and real-time object detection with YOLOv5 to create accurate and functional room representations. The project not only enhanced my technical expertise but also honed practical skills in problem-solving, collaboration, and time management. Additionally, it deepened my understanding of spatial awareness and the application of machine learning in computer vision. This comprehensive learning experience provided a solid foundation for future endeavors in robotics and automation, setting this project apart as a significant milestone in my development.