INTEL-UNNATI AI&ML JOURNAL
Number 2 (2024), pp. 1–8

Code:IUCP-2024

MACHINE LEARNING

Creating a 2D Occupancy Grid using Overhead cameras

Alphonsa Abraham, Navneeth Krishnan, Sidharth V Menon, and Vyshnavi Dipu

Saintgits Group of Institutions, Kottayam, Kerala

Abstract: This project explores the development and implementation of a 2D occupancy grid mapping system using overhead cameras. The primary goal is to enhance environmental perception and navigation capabilities for autonomous systems in indoor settings. Overhead cameras provide a continuous and wide field of view, capturing dynamic changes in the environment. The system processes the video feed to generate real-time occupancy grids, representing the presence or absence of obstacles within a predefined grid space. Key components include transformation techniques to map camera coordinates to grid coordinates, and real-time updating mechanisms for the occupancy grid. The effectiveness of the approach is validated through a series of experiments in controlled environments, demonstrating its accuracy, responsiveness, and potential applications in robotics, security, and smart infrastructure management. This method offers a cost-effective and scalable solution for dynamic environment monitoring, contributing to advancements in autonomous navigation and intelligent system design.

Keywords: 2D occupancy grid, overhead cameras,environmental perception,autonomous systems

1 Introduction

The emergence of intelligent infrastructure and autonomous systems has made sophisticated techniques for navigation and environmental awareness necessary. The 2D occupancy grid mapping system is one such technique that makes use of overhead cameras to track and map changing indoor settings. The goal of this project is to put such a system into operation using Gazebo, a potent robotics simulation tool, and the ROS2 (Robot Operating System 2) framework.
A reliable and adaptable framework for creating and integrating intricate robotic systems

is offered by ROS2. Its support for modularity, scalability, and real-time connectivity makes it the perfect option for occupancy grid mapping. As the simulation environment, Gazebo provides lifelike models of sensors, robotics, and surroundings. When ROS2 and Gazebo are used together, the occupancy grid mapping system may be thoroughly tested and validated prior deployment in real-world scenarios
The first step in the implementation process is to set up the simulation environment in Gazebo. This involves modeling an indoor virtual space and adding both static and dynamic obstacles. The surroundings is continuously captured on video feeds by simulating overhead cameras. After that, these feeds are analyzed by ROS2 nodes, which carry out image processing functions creating a stitched image from 4 overhead cameras and converting into a Occupancy Grid Map. A two-dimensional occupancy grid is mapped each 1s to keep it continuously updated.

Motivation for this project:
This project is driven by the need to improve the navigation and environmental awareness of autonomous systems. The goal of real-time 2D occupancy grid mapping using overhead cameras is to advance robotics, security, and smart infrastructure applications by increasing accuracy, scalability, and cost-effectiveness in dynamic interior environments.

2 Libraries Used

In the project for various tasks, following packages are used.

```
Ubuntu 20.04 Focal Fossy
Python 3.8
ROS2 Foxy
Gazebo Classic
Turtlebot3
OpenCV
Rviz
```

3 Methodology

In this we have main requirements such as a 3D model of an environment with Overhead cameras attached, ROS2 workspace that has turtlebot3, Rviz and Gazebo installed to simulate and access the environment. To implement these the following step were used:

Launching the environment: With the required packages installed we are able to deploy and environment with 4 overhead cameras provided by the Industry Mentor and view the topics of these cameras from which the camera feed can be accessed. Launching the environment gives access the the camera feed topics:

```
rsid@Focal:~$ ros2 topic list
/camera/camera_info
/camera/image_raw
/clock
/cmd_vel
/imu
/joint_states
/odom
/overhead_camera/overhead_camera1/camera_info
/overhead_camera/overhead_camera1/image_raw
/overhead_camera/overhead_camera2/camera_info
/overhead_camera/overhead_camera2/image_raw
/overhead_camera/overhead_camera3/camera_info
/overhead_camera/overhead_camera3/image_raw
/overhead_camera/overhead_camera4/camera_info
/overhead_camera/overhead_camera4/image_raw
/parameter_events
/performance_metrics
/robot_description
/rosout
/scan
/tf
/tf_static
rsid@Focal:~$
```

Figure 1: Topics in ROS2

Collecting Camera Feed: This step requires the creation of a ROS2 package that has access to the ROS2 topics and can acquire images using the pub-sub model used by ROS2. The script creates a subscriber to the overhead camera topics and allows us access to each of the cameras.

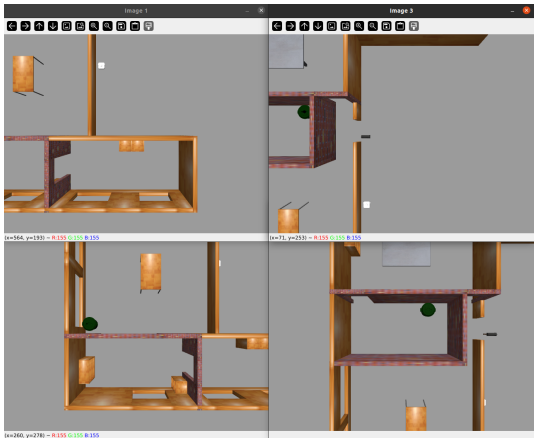

Figure 2: Camera feeds from 4 cameras

Image Stitching: We have obtained images from 4 different cameras at 4 different angles, for the preparation of an Occupancy Grid we need to stitch these distinct images together to form a complete top down image that can be mapped.

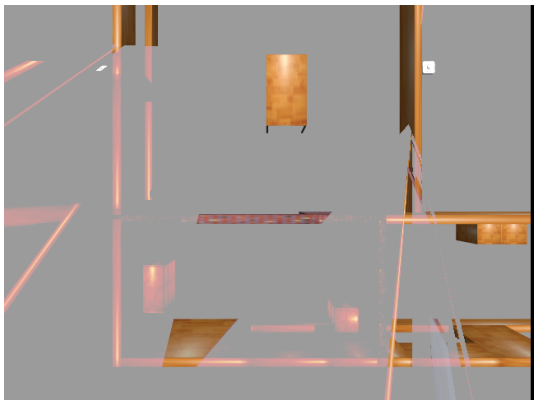

Figure 3: Stitched Image

Color image to Binary image: To convert an image to an occupancy grid it needs to be a Binary image filled with binary values. This is done using OpenCV's image manipulation function.

Generating a 2D Occupancy Grid: A 2D occupancy grid is generated using ROS2 package OccupancyGrid. This package uses the information in the Binary image to generate a Occupancy grid of the environment in the image. This published occupancy grid topic is then used by the map_saver package to save the Occupancy grid as a .pgm file.

Occupancy Grid Evaluation: The Occupancy grid that is generated is then opened in Rviz and we measure the key distances that have been prescribed for evaluation.

4 Implementation

The 1st step in implementation was launching the model provided by the Industry Mentor and checking for the topics of the Overheard cameras that are part of the model. These topics are necessary as they allow us to access the camera feeds in scripts we use to create the occupancy map. These 4 different map feeds are then stitched together using a SIFT

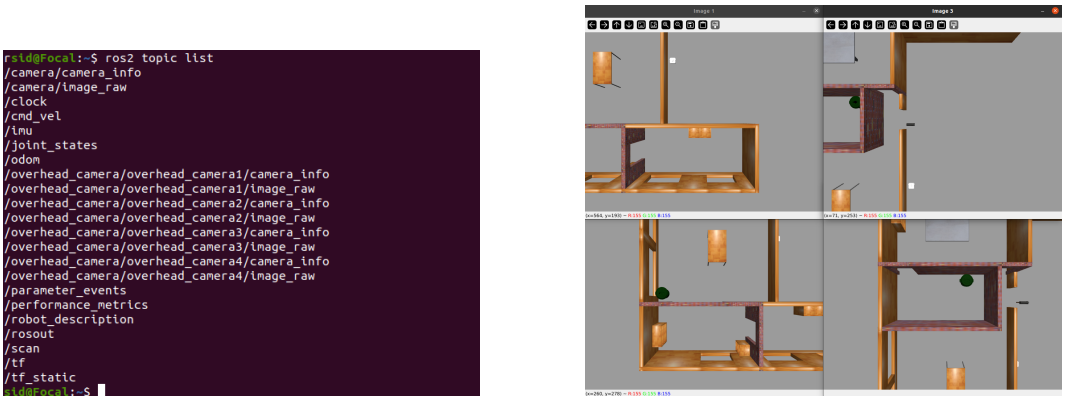

Figure 4: Image feeds and their topics

algorithm so find the similarities between the 4 images and stitch them accordingly. This helps to create a image that is ordered and looks similar to the environment.
The steps are described below:

- *Feature Detection:* Identify potential keypoints by searching for local maxima and minima in the Difference of Gaussian (DoG) images.Refine the location of keypoints by fitting a quadratic function to nearby data points and rejecting low-contrast keypoints.Assign an orientation to each keypoint based on the local image gradient direction.Generate a descriptor for each keypoint by computing the gradient magnitude and orientation in a local region around the keypoint.
- *Feature Matching:* Descriptor Matching: Compare descriptors from different images and find pairs of keypoints with the smallest distance between their descriptors.
  Use techniques such as the ratio test (e.g., Lowe's ratio test) to filter out false matches.
- *Homography Estimation:*RANSAC Algorithm: Use the Random Sample Consensus (RANSAC) algorithm to robustly estimate the homography matrix, which helps in dealing with outliers.
- *Image Warping and Blending:* Perspective Transformation: Apply the homography matrix to transform the image. Blending Techniques: Blend the overlapping regions of the images using techniques like linear blending, multi-band blending, or feathering to create a seamless transition.
- *Panorama Creation:* Combine the warped images to create a single panoramic image. Ensure that the entire region of interest is covered and that the transition between images is smooth.

This completes the stitching process of the image. Now this stitched image is published as a topic and a Occupancy grid is created using OccupancyGrid module from this image. The grid is then saved as a map or .pgm file by using map_saver module.

In the evaluation stage `Rviz` is used to find the distances between key points in the Grid map . The key points are highlighted in the image below and the distances are given in the table below:

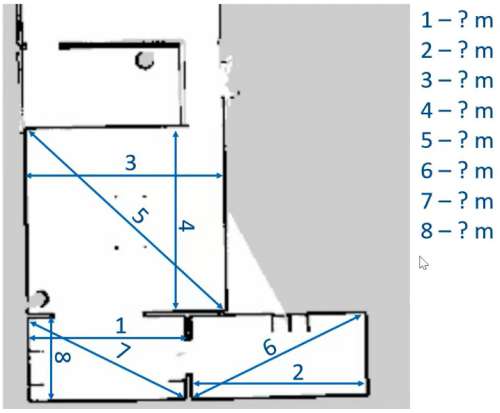

Figure 5: Key points

Connection	Grid map Length
1	4.84m
2	5.63m
3	6.84m
4	7.24m
5	9.28m
6	6.86m
7	5.76m
8	3.73m

Table 1

5 Results & Discussion

Measuring the distances between key points on a grid map using RViz is necessary to guarantee accuracy and precision. The reliability of the map can be evaluated by putting markers at known places and contrasting the measured distances on the grid map with actual distances. Disparities point to areas that need to be adjusted or calibrated. This procedure makes sure that the grid map accurately depicts the surroundings, which is important for applications like robotic path planning and autonomous navigation. The measured points are highlighted in the table below:

Connection	Grid map Length
1	4.84m
2	5.63m
3	6.84m
4	7.24m
5	9.28m
6	6.86m
7	5.76m
8	3.73m

Table 2

The distances are close to the actual distances the variations are almost completely scaled to the same amount across the key points.

6 Conclusions

The use of above cameras to construct a 2D occupancy grid mapping system that is connected with Gazebo and ROS2 shows a notable improvement in autonomous systems' ability to perceive their surroundings. This study effectively used simulation tools and strong image processing techniques to create an accurate, real-time occupancy grid. The system's accuracy and dependability were validated by the RViz study, which also highlighted the system's potential for a wide range of robotics, security, and smart infrastructure applications. Able to map and monitor indoor environments dynamically provides significant advances in intelligent system design and autonomous navigation. To further confirm the system's performance and scalability, future study could include adding more sensors, improving image processing methods, and implementing the system in real-world situations. This initiative establishes a strong basis for future innovation in intelligent environments and autonomous systems.

Acknowledgments

We would like to express our heartfelt gratitude and appreciation to Intel© Corporation for providing an opportunity to this project.First and foremost, we would like to extend our sincere thanks to our team mentor Dr Starlet Ben Alex for his invaluable guidance and constant support throughout the project.We are deeply indebted to our college Saintgits College of Engineering and Technology for providing us with the necessary resources,and sessions on machine learning. We extend our gratitude to all the researchers, scholars, and experts in the field of Robotic Operation System 2, Python programming and Gazebo whose seminal work has paved the way for our project. We acknowledge the mentors, institutional heads, and industrial mentors for their invaluable guidance and support in completing this industrial training under Intel© -Unnati Programme whose expertise and encouragement have been instrumental in shaping our work. []

7 References

1. Project Reference, Amartya Saikia, https://github.com/florist-notes/aicore_n/blob/main/notes/int2.MD

2. ros2, Quigley, M., Gerkey, B., & Smart, W. D. (2015). *Programming Robots with ROS: A Practical Introduction to the Robot Operating System*. O'Reilly Media, Inc.

3. gazebo, Koenig, N., & Howard, A. (2004, September). Design and use paradigms for Gazebo, an open-source multi-robot simulator. *In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2149-2154.

4. sift, Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2), 91-110.

5. occupancy grid, Elfes, A. (1989). Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6), 46-57.

6. rviz, Gossow, D., Kurniawati, H., & Collett, T. (2011). *RViz: A 3D visualizer for ROS*. [Software]. Available: https://wiki.ros.org/rviz.

7. image processing Gonzalez, R. C., & Woods, R. E. (2008). *Digital Image Processing* (3rd ed.). Pearson.

8. ransac, Fischler, M. A., & Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6), 381-395.

9. feature matching Bay, H., Ess, A., Tuytelaars, T., & Van Gool, L. (2008). Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, 110(3), 346-359.

10. autonomous navigation Thrun, S., Burgard, W., & Fox, D. (2005). *Probabilistic Robotics*. MIT Press.