

Simultaneous Localization and Mapping (SLAM) using RTAB-Map

Sagarnil Das

Abstract— This paper implements Simultaneous Localization and Mapping (SLAM) technique to construct a map of a given environment. A Real Time Appearance Based Mapping (RTAB-Map) approach was taken for accomplishing this task. Initially, a 2d occupancy grid and 3d octomap was created from a provided simulated environment. Next, a personal simulated environment was created for mapping as well. In this appearance based method, a process called Loop Closure is used to determine whether a robot has seen a location before or not. In this paper, it is seen that RTAB-Map is optimized for large scale and long term SLAM by using multiple strategies to allow for loop closure to be done in real time and the results depict that it can be an excellent solution for SLAM to develop robots that can map an environment in both 2d and 3d.

Index Terms— Localization, Mapping, Occupancy Grid Mapping, RTAB-Map, SLAM, Graph SLAM

1 INTRODUCTION

IN SLAM (Simultaneous Localization and Mapping), a robot must construct a map of the environment, while simultaneously localizing itself relative to this map. This problem is more challenging than localization or mapping, since neither the map nor the robot poses are provided making this problem a 'chicken or a egg' problem. With noise in the robot's motion and measurements, the map and robot's pose will be uncertain, and the errors in the robot's pose estimates and map will be correlated. The accuracy of the map depends on the accuracy of the localization and vice versa. Given a series of sensor observations over discrete time steps, the SLAM problem is to compute an estimate of the agent's location and a map of the environment.

In this paper, two simulation environments were provided where SLAM was performed. The robot was successfully able to localize itself and map the 3d world. The benchmark environment is called kitchen-dining Figure 1 and the second environment is that of a cafeteria called sagar-cafe Figure 2.

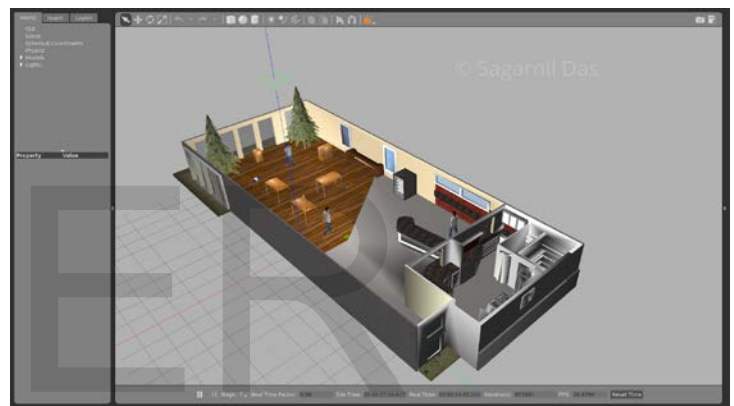


Figure 2: Kitchen - Dining world

2 BACKGROUND

SLAM algorithms generally fall into 5 categories:

1. Extended Kalman Filter SLAM (EKF)
2. Sparse Extended Information Filter (SEIF)
3. Extended Information Form (EIF)
4. FastSLAM
5. GraphSLAM

The two most useful approaches to SLAM are **Grid based FastSLAM** [1] and **GraphSLAM** [2] and these two algorithms will be discussed here.

2.1 Grid based FastSLAM

The FastSLAM algorithm uses a custom particle filter approach to solve the full SLAM problem with known correspondence. Using particles, FastSLAM estimates a posterior over the robot's path along with the map. Each of these particles holds the robot's trajectory which gives an advantage to SLAM to solve the problem of mapping with known poses. In addition to the trajectory, each particle holds a map and each feature of the map are represented by a local Gaussian.

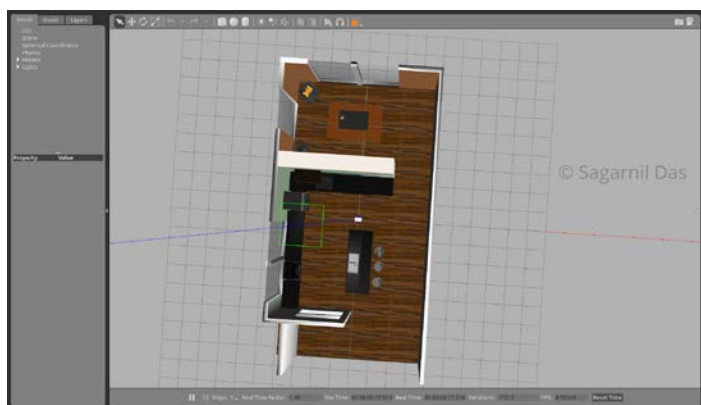


Figure 1: Kitchen - Dining world

With the FastSLAM algorithm, the problem is now divided into two separate independent problems, each of which aims to solve the problem of estimating features of the map. To solve these independent mini-problems, FastSLAM will use the low dimensional Extended Kalman Filter. While map features are treated independently, dependency only exists between robot pose uncertainties. This custom approach of representing the posterior with particle filter and Gaussian is known by **Rao-Blackwellized particle filter approach** [3]. The Grid based FastSLAM is really an extension of FastSLAM and it adapts FastSLAM to grid maps.

With grid mapping algorithm, the environment can be modeled using grid maps without predefining any landmark position. So by extending the FastSLAM algorithm to occupancy grid maps, the SLAM problem can now be solved in an arbitrary environment. While mapping the real world environment, mobile robots equipped with range sensors can be used and the FastSLAM algorithm can be extended to solve the SLAM problem in terms of grid maps.

2.1.1 Grid based FastSLAM techniques

Adapting the FastSLAM algorithm to grid maps is altered in the grid based FastSLAM algorithm. Since the grid based FastSLAM algorithm uses a particle filter approach and represents the world in terms of grid maps, both MCL (Monte Carlo Localization) and Occupancy Grid Mapping algorithm are combined. Now three different techniques are needed which are represented by 3 probability functions to adapt FastSLAM to grid mapping. These techniques are known as:

1. **Sampling motion** $P(x_t | x_{t-1}^{[k]}, u_t)$: Estimates the current pose given the k^{th} particle's previous pose and controls u (MCL).
2. **Map Estimation** $P(m_t | z_t, x_t^{[k]}, m_{t-1}^{[k]})$: Estimates the current map given the current measurements, the current k^{th} particle's pose and the previous k^{th} particle map (use Occupancy Grid Mapping).
3. **Importance weight** $P(z_t | x_t^{[k]}, m_t^{[k]})$: Estimates the current likelihood of the measurement given the current k^{th} particle pose and the current k^{th} particle map (MCL).

The sampling motion, map estimation and importance weight techniques are the essence of the grid based FastSLAM algorithm. Grid based FastSLAM implements them to estimate both the map and the robot's trajectory, given the measurements and the control. The grid based FastSLAM algorithm looks very similar to Monte Carlo localization algorithm with some additional statements concerning the map estimation. Figure 3 denotes the Grid based FastSLAM algorithm.

2.2 GraphSLAM

Graph SLAM is a SLAM algorithm that solves the full SLAM problem. This means that the algorithm recovers the entire path and map, instead of just the recent pose and map. This

Figure 3: Grid based FastSLAM algorithm

Algorithm 1 Grid based FastSLAM

```

1: procedure GRID BASED FASTSLAM( $X_{t-1}, u_t, z_t$ )
2:    $\bar{X}_t = X_t = \phi$ 
3:   for  $k = 1$  to  $M$  do
4:      $x_t^{[k]} \leftarrow \text{sample-motion-model}(u_t, x_{t-1}^{[k]})$ 
5:      $w_t^{[k]} \leftarrow \text{measurement-model-map}(z_t, x_t^{[k]}, m_{t-1}^{[k]})$ 
6:      $m_t^{[k]} \leftarrow \text{updated-occupancy-grid}(z_t, x_t^{[k]}, m_{t-1}^{[k]})$ 
7:      $\bar{X}_t = \bar{X}_t + \langle x_t^{[k]}, m_t^{[k]}, w_t^{[k]} \rangle$ 
8:   for  $k = 1$  to  $M$  do
9:     draw  $i$  with probability  $\propto w_t^{[i]}$ 
10:    add  $\langle x_t^{[i]}, m_t^{[i]} \rangle$  to  $X_t$ 
11:  Return  $X_t$ 

```

difference allows it to consider dependencies between current and previous poses. One of the benefits of graph SLAM is the reduced need for significant on-board processing capability. Another is graph SLAM's increased accuracy over fast SLAM. Fast SLAM uses particles to estimate the robot's most likely pose. However, at any point in time, it is possible that there is not a particle in the most likely location. In fact, chances are slim to none especially in large environments. Since graph SLAM solves the full SLAM problem, this means that it can work with all of the data at once to find the optimal solution.

In graph SLAM, the idea is to organize information in a graph. A node in the graph represents either a robot pose x_t at a specific time step t or the location of a feature in the environment denoted as $m^{(i)}$ with $i = 1, \dots, \alpha$. An edge in the graph represents either a measurement constraint between a pose and a feature or a motion constraint between two successive poses. Since the spatial constraints are soft, they can be considered as springs connecting two masses. In this analogy, the full SLAM problem can be solved as a global graph optimization problem. The optimal graph configuration is the one where the springs are relaxed, and the forces on each of the nodes are minimized.

The Maximum Likelihood Principle (MLE) is used to optimize the graph. When applied to SLAM, likelihood tries to estimate the most likely configuration of state and feature locations given the motion and measurement observations. The measurement update at time step t is given by

$$\bar{z}^t := x_t + m_t^{(i)} \quad (1)$$

which represents for instance a laser range finder measuring the distance to the landmark $m^{(i)}$. Equivalently, a motion update can be defined as

$$\bar{x}^t := x_{t-1} + u_t \quad (2)$$

which could be realized as a control command instructing the robot to move a certain distance u_t . The updates are assumed to have Gaussian noise. The corresponding probability distributions are given by

$$p_u(x_t) = \frac{1}{\sigma_u \sqrt{2\pi}} e^{-(x_t - \bar{x}_t)^2 / 2\sigma_u^2} \quad (3)$$

$$p_m(z_t) = \frac{1}{\sigma_m \sqrt{2\pi}} e^{-(z_t - \bar{z}_t)^2 / 2\sigma_m^2} \quad (4)$$

In some simple cases it is possible to find an analytical solution to MLE by converting the target function to the negative log-likelihood form

$$J_{GraphSLAM} = \sum_t \left(\frac{z_t - \bar{z}_t}{\sigma_m} \right)^2 + \sum_t \left(\frac{x_t - \bar{x}_t}{\sigma_u} \right)^2 \quad (5)$$

trying to minimize the sum of all constraints. In more complex realistic scenarios, approximate numerical solutions are needed, for instance by applying gradient descent techniques.

In real world, most systems are multi-dimensional and to tackle such scenarios, matrices and covariances must be used. The state and measurement are given by x_t and z_t . The constraints are given by

$$J_{GraphSLAM} = x_0^T \Omega x_0 + \sum_t (w_t^T R_t^{-1} w_t + v_t^T Q_t^{-1} v_t) \quad (6)$$

The goal of graph SLAM is to create a graph of all robot poses and features encountered in the environment and the most likely robot's path and map of the environment. This task can be broken down into two sections. The *front-end* and the *back-end*.

2.2.1 Grid based FastSLAM techniques

The front end of graph SLAM looks at how to construct the graph, using the odometry and sensory measurements collected by the robot. This includes interpreting sensory data, creating the graph and continuing to add nodes and edges to it as the robot traverses the environment. Naturally the front end can differ greatly from application to application depending on the desired goal, including accuracy, the sensor used and other factors e.g. the front end of a mobile robot applying SLAM in the office using a Laser Range finder would differ greatly from the front end of a vehicle operating on a large outdoor environment and using a stereo camera. The front end of graph SLAM also has the challenge of solving the data association problem. In simpler terms, this means accurately identifying whether features in the environment have been previously seen.

The back end of graph SLAM is where the magic happens. The input to the back end is the completed graph with all of the constraints and the output is the most probable configuration of robot poses and map features. The back end is an optimization process that takes all of the constraints and finds the system configuration that produces the smallest error. The back end is a lot more consistent across applications. The front end and the back end can be completed in succession or can be performed iteratively, with a back end feeding an updated graph to the front end for further processing.

2.2.2 Using RTAB-Map for 3D Graph SLAM

RTAB-Map (Real Time Appearance Based Mapping) [4] is a

graph based SLAM approach. Appearance based SLAM means that the algorithm uses data collected from vision sensors to localize the robot and map the environment. In appearance based methods, a process called *Loop Closure* is used to determine whether the robot has seen a location before. As the robot travels to new areas in its environment, the map is expanded and the number of images that each new image must be compared to increases. This causes the loop closure to take longer with the complexity increasing linearly. RTAB-Map is optimized for large scale and long term SLAM by using multiple strategies to allow for loop closure to be done in real time. Figure 4 shows the block diagram of the front end and the back end.

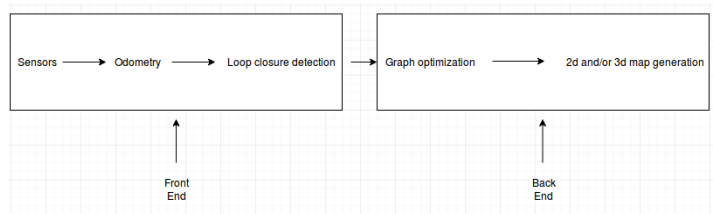


Figure 4: RTAB-Map Front end and Back End block diagram



3 SCENE AND ROBOT CONFIGURATION

The ROS package **slam_project** deploys the RTAB-Map to perform SLAM on two environments. The first environment is an environment provided by Udacity as a part of this research project and is named *Kitchen-Dining*. The second environment is a custom made environment of a cafeteria named *sagar_cafe*. Just like other robotics project, this project has been organized into different folders containing the meshes, Gazebo SDF files, scripts, robot model URDFs, configuration files and launch files.

The URDF folder contains the files *sagar_bot.xacro* defining the links and joints of the robot model used for physics simulation and visualization as well as the file

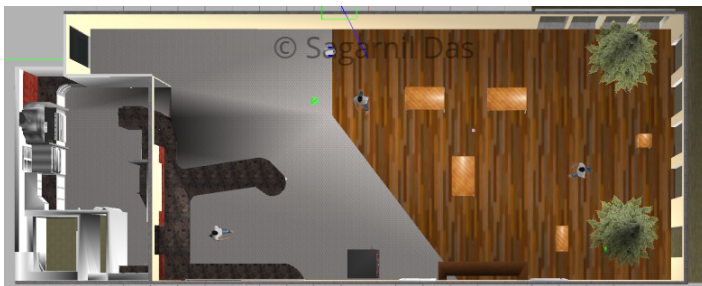


Figure 8: Bird's eye view of Sagar cafe

3.3 Launch file configuration

Four launch files are required for a successful mapping of the environments in simulation. The gazebo simulation environment is (kitchen-dining or cafe) specified in the **world.launch** file. The **teleop.launch** file launches the teleop keyboard which is required for moving the robot in the simulation world. The **mapping.launch** file is used to start the RTAB-Map node. This node is used for loop closure detection using the ORB-SLAM algorithm. ORB-SLAM is a versatile and accurate Monocular SLAM solution able to compute in real-time the camera trajectory and a sparse 3D reconstruction of the scene in a wide variety of environments, ranging from small hand-held sequences to a car driven around several city blocks. It is able to close large loops and perform global re-localization in real-time and from wide baselines. Finally, the **rviz.launch** file starts visualization of the rover, sensor data, as well as map and camera topics in RViz. Figure 9 depicts RViz view of the world at the starting point.

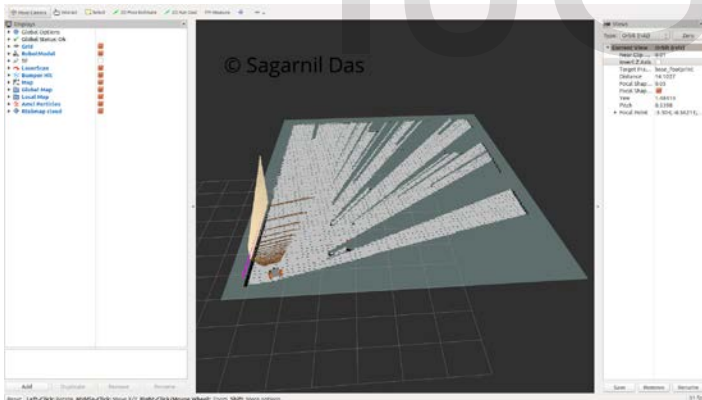


Figure 9: Starting point of the robot as seen in RViz

During the mapping of the environment, the mapping data is saved in the rtabmap.db database. The localization.launch file can be started in order to localize the robot during the run.

4 RESULTS

The mapping was done by the robot controlled by the teleop keyboard. In order to be able to have more than 3 loop closure detection, which was the project's benchmark, the robot was navigated through the full environment of both the worlds so that it could collect more images.

4.1 Kitchen-Dining world

The mapping run in the provided world ended with 66 global loop closures. This file has a size of 315 MB and is named as rtabmap_kitchen_dining.db. Figure 10 shows the robot's trajectory as well as the 2d occupancy grid map of the kitchen_dining world.

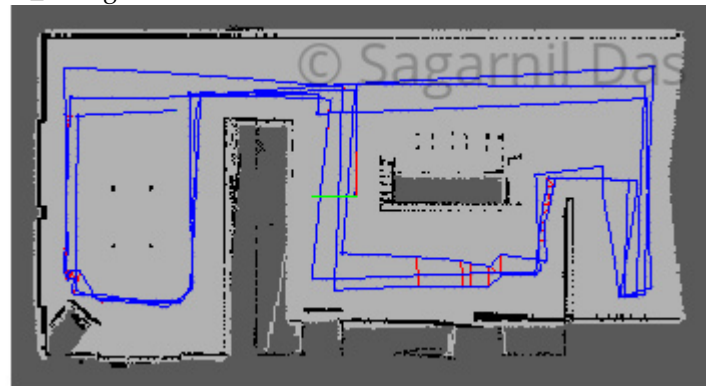
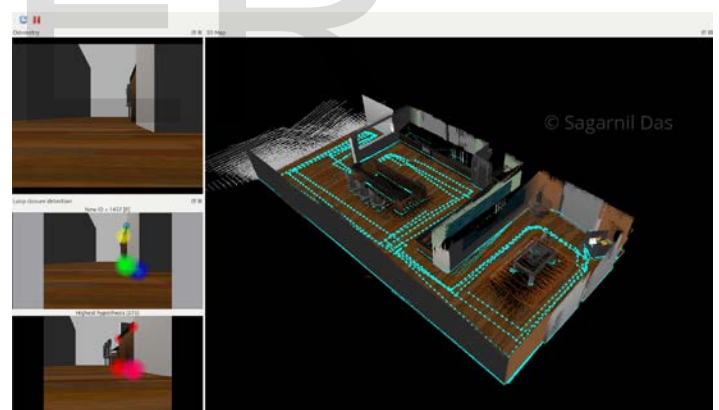


Figure 10: Robot's trajectory and 2d occupancy grid map of the Kitchen-Dining world

At the end of the multiple passes, a well structured 3d point cloud map was created by using the *Export 3d Clouds* functionality. Figure 11 depicts the reconstructed 3D point cloud data. It can be seen that most features in the world like the chairs and tables are reconstructed properly and are distinctive. Figure 12 shows the RViz result of the same world after the end



of the mapping task.

Figure 11: Reconstructed Point cloud data in RTAB-Map viewer of the Kitchen-Dining world

At the end of the map, the loop closures can be seen in the rtabmap_kitchen_dining.db. Figure 13 shows one of them.

4.1 Sagar-cafe world

In the custom made world - sagar_cafe.world, the robot performed well. As the robot is very short, some of the taller objects like the people, trees are not fully mapped. The kitchen also could not be traveled by the robot. Figure 14 and 15 depicts the RTAB-map view and the RViz view of this world respectively at the end of the mapping.

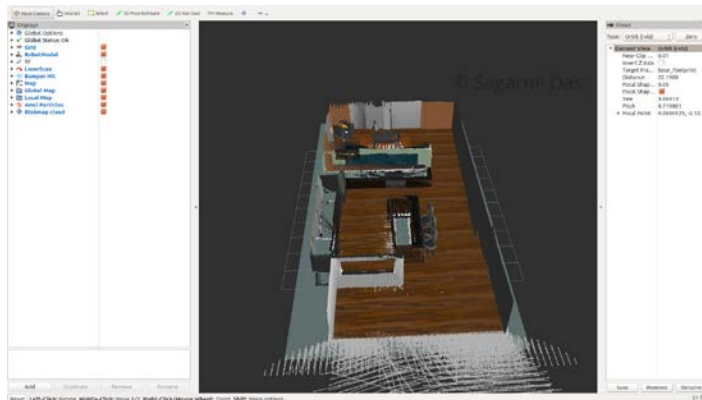


Figure 12: Rviz view of the Kitchen-dining world

5 DISCUSSION

In both the environments, successful mapping was performed in order to identify the ground truth and the distinctive features of the environment like, walls, tables, chairs, trees, people etc. In the *sagar_cafe* world, the robot could not enter the kitchen area, which can be seen in right lower corner of Figure 14 and Figure 15. One of the possible explanations might be that there is a height difference of the floor between the living room and the kitchen. Another possible explanation might be that there is a transparent door separating those two rooms and hence the robot couldn't pass through.



Figure 13: Loop Closure detection

The generated 2d and 3d maps can be improved by doing more mapping runs which cover the environment in a more complete manner and by optimizing the loop closure detection further.

6 CONCLUSION/FUTURE WORK

An interesting future work would be to explore the RTAB-Map package's visualization section in more details. The obstacle detection feature can be deployed in order to extract obstacles and the ground from your point cloud. With this information in hand, these obstacles can be avoided when executing a desired path. Another potential area would be Wifi signal strength mapping. This feature allows the user to visualize the strength of your robot's Wifi connection. This can be important in order to determine where the robot may lose its signal, therefore dictating it to avoid certain areas. The situation can also be remedied with large

er antennas.

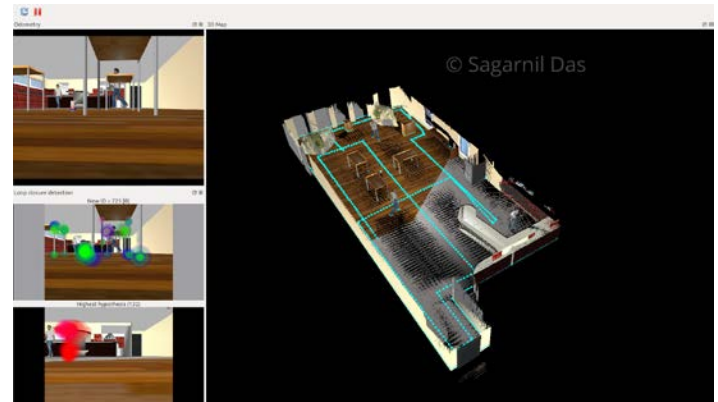


Figure 14: Reconstructed Point cloud data in RTAB-Map viewer in the *sagar_cafe* world

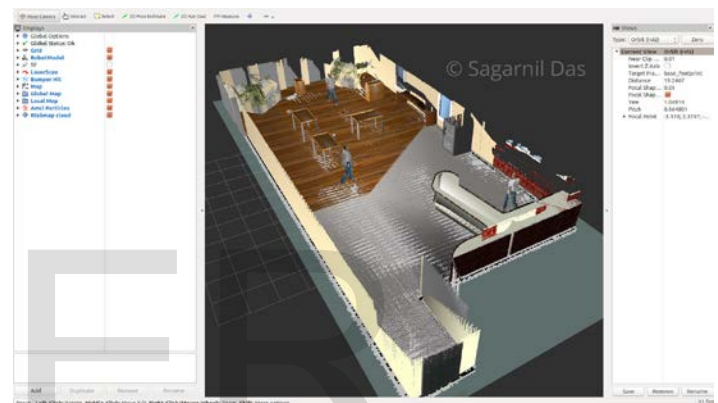


Figure 15: RViz view of the *sagar_cafe* world

7 END SECTIONS

7.1 Acknowledgements

The author sincerely thanks Udacity for their support and critiques throughout the project. The *kitchen-dining* world was provided by them.

REFERENCES

- [1] M. Montemerlo, S. Thrun, D. Koller and B. Wegbreit, "FASTSLAM: A FACTORED SOLUTION TO THE SIMULTANEOUS LOCALIZATION AND MAPPING PROBLEM," *In Proceedings of the AAAI National Conference on Artificial Intelligence*, pp. 593-598, 2002, AAAI.
- [2] S. Thrun and M. Montemerlo, "THE GRAPH SLAM ALGORITHM WITH APPLICATIONS TO LARGE SCALE MAPPING OF URBAN STRUCTURES," *The International Journal of Robotics Research* volume 25, issue 5-6, pp. 403-429, 2006, IJRR.
- [3] G. Grisetti, C. Stachniss and Wolfram Burgard, "IMPROVED TECHNIQUES FOR GRID MAPPING WITH RAO-BLACKWELLIZED PARTICLE FILTERS," *IEEE Transactions on robotics*, volume 23, issue 1, pp. 34-46, 2007, (IEEE transactions).
- [4] M. Labbe and F. Michaud, "ONLINE GLOBAL LOOP CLOSURE DETECTION FOR LARGE SCALE MULTI-SESSION GRAPH BASED SLAM," *IEEE/RSJ International Conference on Intelligent Robots and systems (2014)*, pp. 2661-2666, doi: 10.1109/IROS.2014.6942926.
- [5] S. Das, "ROBOT LOCALIZATION IN A MAPPED ENVIRONMENT USING

ING ADAPTIVE MONTE CARLO ALGORITHM", unpublished,
[https://drive.google.com/open?id=13pkunfwA80RmjE8vxIkLJ8PTaOl6DR](https://drive.google.com/open?id=13pkunfwA80RmjE8vxIkLJ8PTaOl6DRYG)
YG

IJSER