Assignment 2 - Lab Report
AI and Robotics (236609)

Prunotto Alice
alicep@campus.technion.ac.il
XX Department
Technion - Israel Institute of Technology

Elashkin Andrew
swag@campus.technion.ac.il
XX Department
Technion - Israel Institute of Technology

February 4, 2023

1 Introduction

In this practical assignment we are provided a map of the walls of our environment, the approximate location and the number of objects with different shapes of unknown size placed in the room. Our task is to examine from all sides all the objects with a single robot with Lidar and add them to a map.
This problem can have real-life applications. For example, it finds an application in warehouse automation and inventory management, where a robot equipped with Lidar can map a warehouse and detect and examine all the items in the warehouse to update the inventory database. Another example can be in the field of industrial inspection, where a robot equipped with Lidar can be used to examine and map industrial facilities to detect possible maintenance needs.

2 Background

Below is the required background for our setup:

- SLAM (Simultaneous Localization and Mapping): it is an algorithm that enables a robot to map its environment and estimate its own pose (location and orientation) simultaneously. In this task, SLAM can be used examines objects and build a map of them. The robot can use the Lidar data to detect the walls and other objects in the room and update its map accordingly.
- Path planning: the process of determining the best path for the robot to follow to achieve a desired goal. In this task, path planning can be used to determine the most efficient path for the robot to follow to examine the objects.
- Object recognition: the process of identifying objects in an image or a point cloud.
- Map: maps are stored in an YAML file which is used to store configuration information in ROS, including information about maps.

- OpenCV (Open Source Computer Vision Library): a library of programming functions mainly aimed at real-time computer vision.

3 Setup

In this section we will give a brief explanation of our code.

Upon the turtlebot client creation, we subscribe to several channels, similar to the code provided in the tutorials: '/initialpose' to set the initial position, '/move_base/global_costmap/costmap_updates' to update the costmap and '/amcl_pose' to update our current position relative to the map. We also create the 'move_base' client to be able to set a goal command for the robot.

The algorithm can be split in two main parts:

1. Choosing the order in witch we visit our POI, planning the main route through all POI and implementing it.
2. Re-planning the walk-around for each of the objects when we reach it to properly map it from all angles.

For solving the part 1 we decided to go with the Greedy Planner approach. Because we know that our POI are not located in the maze we can simply choose to visit the closest one first, then plan for the second one that is the closest to the first one etc. There could be cases when this approach fails since we are using Euclidean distance to plan, but it can be easily fixed by more sophisticated heuristics. In our case the Euclidean distance seemed to be the best choice in terms of computational cost / final result.

For part 2 we've chosen to go with the square re-planning. First, we build the new costmap without walls (cww) map, that is build by subtracting the given walls from the costmap each time the costmap is updated. While navigating to the POI from the list we constantly check this map to see if there is a new wall between our current location and our goal. This is done by computing the shortest path on discrete map with bresenham algorithm and then comparing the points of the path to preset threshold on costmap. If we have a hit that means that there is a wall in front, so then we check that the robot is close to the goal and do the re-planning of the path. We know that the obstacle is either sphere or cube, so we re-plan to go around it in a square, where our current location is the square's corner and the goal is the center of a square. When this is done the robot switches to mode 1 that plans for the next POI as described in part 1.

4 Experiments

Unfortunately since the lab was closed for the renovation we could not test our code on the physical robot, so we the results below are for simulation only.
We used the "2 spheres map", which was the one provided to us. Our robot was set in an initial position and the two spheres were distributed in the room. We started giving an initial position to the robot, and then we gave the robot a position to reach. We learned how to use the move_base function, in order to give a goal command to the robot. Following, we worked on the costmap. We

gave the robot the positions of the objects and, through the Greedy Planner, the robot started to go to the given goal destination, and we could see that going into it, it was scanning the place and building the new map.

5 Results

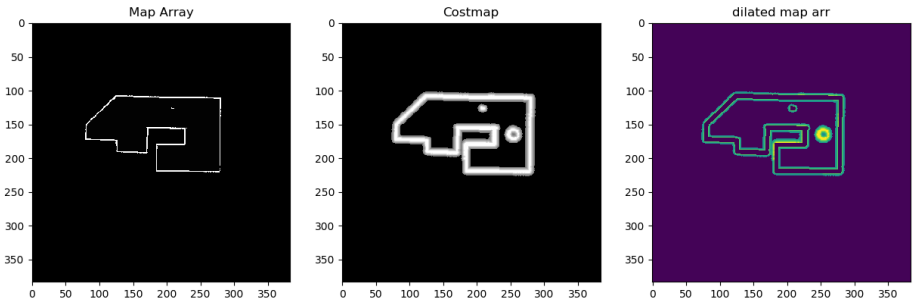


Figure 1: Map array, Costmap and Dilated Costmap after mapping the first sphere.

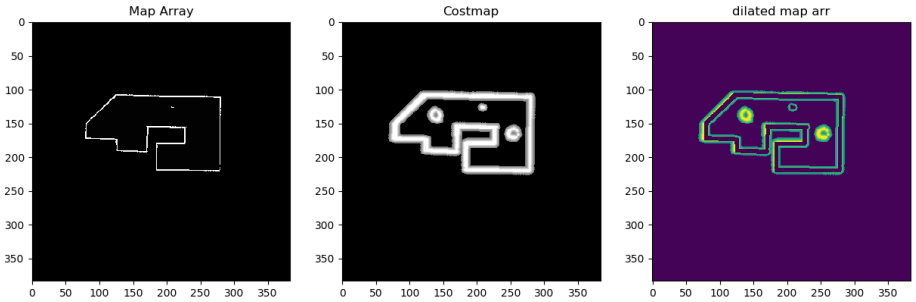
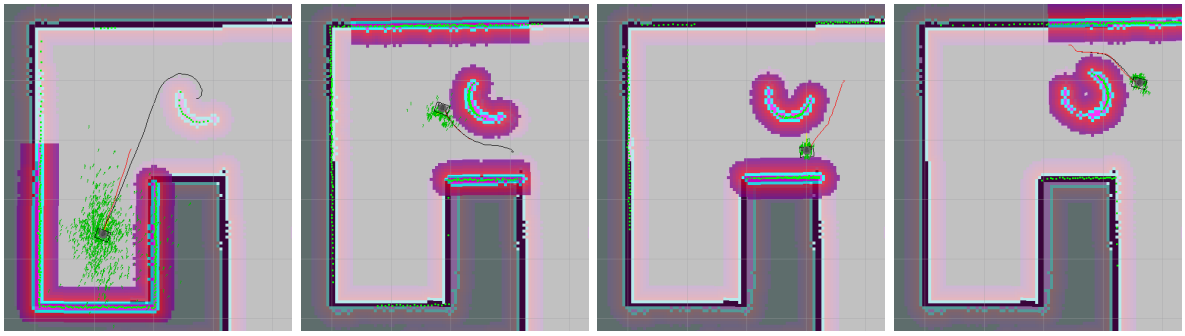


Figure 2: Map array, Costmap and Dilated Costmap after mapping the second sphere.

Only one map was provided for the experiments, so the results are for the map with 2 spheres only. Our algorithm completed the mapping task in 2 minutes and 22 seconds. This is almost twice lower than the given threshold of 4 minutes. Figure 1 and 2 shows the mapping process. The map array is static and provided by the GetMap at the beginning of the run. The costmap is updated every time the costmap_updates message is published and is not 100% accurate because of the drift, but the spheres can be easily recognized. The third map is a subtraction of dilated map array from the costmap and is the map we used to detect obstacles during our run.

Replannig process can be seen on Figure 3. The code can be easily modified to go around the POI in n-edged polyhedron by providing parameter n to the replanner, the square was chosen for the simplicity.



(a) Initial plan (b) Replan to corner 1 (c) Replan to corner 2 (d) Replan to corner 3

Figure 3: Replanning process

6 Discussion

The experiment teaches us several important tasks from different fields. We learned the importance of an accurate mapping in an environment with uncertain obstacles. This is important because the map is used to decide the robot's movements and actions. We then learned the importance of detect and recognize objects, and distinguish them from the map's walls. This task thought us how to scan an object after finding it. Finally, we learned the importance of choosing the efficient planning algorithm that is relatively simple and can plan an efficient path to scan all the objects.