LONDON'S GLOBAL UNIVERSITY

COMP0037

Report

Exploration in Unknown Environments

Group AS

Student Name	**Student number**
Arundathi Shaji Shanthini	16018351
Dmitry Leyko	16021440
Tharmetharan Balendran	17011729

Department: Department of Electronic and Electrical Engineering

Submission Date: 17th of March 2020

Contents

1 Reactive Planner

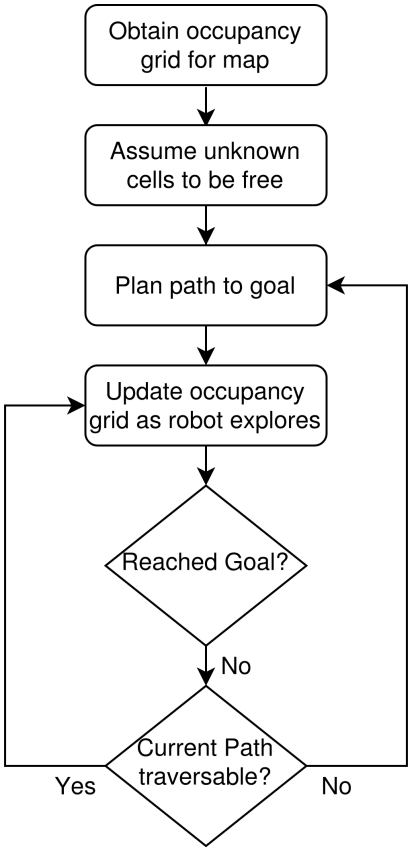


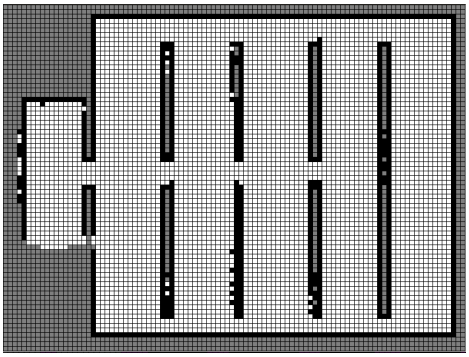
Figure 1: Flowchart for the reactive planner

1.1 Our Implementation of a Reactive Planner

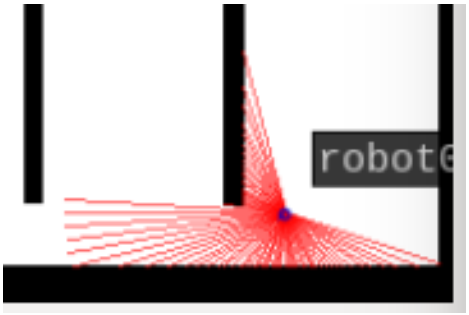
A reactive planner is able to adapt its path based on the information it obtains about the environment as it explores it. The high-level functionality of a general reactive planner is described in the flowchart in Fig.1. The reactive planner initially utilizes the available occupancy grid and assumes any unknown cells to be free. The planner then plans a path using this assumption and starts to traverse the path. The robot will explore the environment as it traverses the path and if new information suggests that the currently planned path is no longer traversable, a new path is planned using the new occupancy grid. This process is repeated until the robot arrives at the goal.

As a means of testing the code the robot was set to visit a list of goals on the factory map. The final mapper node occupancy grid is shown in Fig.2a. It is clear to see that there are inaccuracies in the mapping of the world as well as some areas which have still not been explored completely. The inaccuracies occur as a result of rounding errors and other noise due to timing mismatch (caused by networking delays), these errors are mostly corrected as the robot re-explores an area.

The way the map is updated is using the laser range finder that the robot is equipped with. Depending on the reflections of this laser, the environment surrounding the robot can be mapped. Fig.2b depicts an instance in time of the robot. The red lines represent the individual laser beams used to map the surrounding environment. It is clear to see that the laser beam is stopped (and reflected) by the opaque walls. By measuring the time taken for the reflected beam to return, the distance to the opaque object can be calculated. In addition to this, it can also be seen that some of the beams stop despite not having hit any opaque objects. This is due to the maximum measurable range of the laser range detector.



(a) Mapper Node occupancy grid



(b) Robot Laser Range Finder

Figure 2: a) Mapper occupancy grid at the end of traversing the set of goals. b) The robot traversing the map with the laser range finder visualized by the red lines. The maximum range as well as the opaque walls preventing the laser from passing through can also be seen.

1.2 Improving the performance of the Reactive Planner

Our implementation of a reactive planner is quite inefficient and may be computationally intense. The current global planner that is used to plan the paths uses an implentation of Dijkstra's algorithm to plan the path. One way of improving the performance is by utilizing the A* algorithm. This way fewer cells will be considered when planning the path resulting in a lower computational cost.
Another method of improving efficiency is to prevent the algorithm from having to replan the whole path. In a dynamic environment, obstacles may move in and out of the robot's path. Therefore, sometimes it may be sufficient to take a remedial action to overcome/avoid the obstacle. This may involve simply waiting for the obstacle to pass or in more complicated cases it may involve the robot moving the obstacle out of the way. How-ever not all obstacles may be overcome in such a way. For example a newly discoverd wall will neither move on its own nor is it possible for the robot to move the wall. In such a case, a local planner may be utilized to plan a path around the obstacle. This involves planning a path to another point on the proposed path and thereby avoiding the obstacle. These local planners are very responsive and generally utilize gradient-based techniques which use physics-based approximations. Despite being fast and responsive, they generally get stuck in local minima.

2 Frontier-Based Exploration System

2.1 Frontier Exploration

A frontier is a boundary where the knwon world ends and the unknown world beguins. In the exploaration the algoritm plans the route to account for the obstacles in the known world and assumes that all of the unknown world is empty. Thus it plans the rought based on this. The robot then starts to travel along this path, and constantly exploring the seroundings, expanding the frontier. Once it detects that the current panth can't be compleated due to an obstacle in the way, it will rerun the planning algorithm with the newly updated frontier. This continues until the robot reaches it's destination. This path may not be the optimal as it does hevely depend on the movements of the robot to descover the new frontiers.

2.2 Frontier Detection

There are two main ways of detecting frontiers that we will be using in this cource work. They are Wave Front Detection and Fast Fronier Detection.

In Wave Front Detection the algorithm uses already explored map to then use deapth first search from

the reobots initial position to find cells that identify as frontier cells. These cells are free, and they have other cells arround them that it doesn't know the state of. If such a sell is found it is marked as a frontier cell. It then suspends the deapth first search, and looks at the neighbours of the found frontier cell. If they also qualify to be frontier cells they are marked so, and the process of expanding the frontier continues. Once the algorithm has found the final frontier cell in this chain it returns to the original deapth first search, and compleates it to look for further frontiers. This algorithm looks at a whole map a once.

The Fast Fronier Detection works on the basis that the robot constantly updates the data collected from the sensors. This data is then used to create a contour, which are then classified into frontier and non-frontier segments. Once these are identified they are stored in the database. These will also get updated once data gets changed or updated.

2.3 Waypoint Selection

Once the frontiers are identified the robot needs to decide on where to move next to, as it needs to explore the map. For this there are two main methods, heuristic based aproach and information based aproach. Useing these the robot will be able to pick the next waypoint to move to.

2.3.1 Heuristic Approaches

There are two main heuristics that are used for waypoint selection the closeset frontier heuristic and largest frontier heuristic. For the closest frontier waypoint the robot looks at the frontiers that it has found, and identifies the closest one to the robot. The robot then assigns it as a waypoint and reroutes the path to it. This has a big advantage of having the shortest path to each waypoint from the robots position, thus saving on travel distance and travel time. For the seccond heuristic, the largest frontier heuristic, the algorithm will look for the largest frontier avalible. This could potentialy signify a much larger are to explore for the robot thus also saving on exploration time.

2.3.2 Information Based Approach

This method relies on calculation the unsertainty in the map and deciding what the best action would be to reduce thw said unsertainty. To calculate the unsertainty we will use entropy of the grid. Entropy, in this case, is a measure that tells the unsertainty that the particular grid has empty or ocupied spaces. The entropy is calculated based on the already explored map, thus extrapolating information about the remaider of the map.

3 Mapping the Environment

4 Information-Theoretic Path Planning

References

Appendices