PATH PLANNING FOR TURTLEBOT USING MODIFIED A-STAR ALGORITHM WITH MONOCULAR SLAM

Project 5 Proposal

ENPM 661

Authors

Rohan Singh

Abhinav Modi

Kamakshi Jain

March 13, 2019

Contents

Chapter 1

Introduction

The term path planning has different definitions in different fields. In robotics, path planning concerns with problem as to how to move a robot from one point to another point. Generally, path planning is focused on designing algorithms that generate useful motions by processing simple or more complicated geometric models. A common task for a mobile robot(here Turtlebot) is to navigate in an indoor environment. Navigation of the Turtlebot in an environment is dependent on the knowledge of the environment. It might be asked to perform tasks such as building a map of the environment and determining its precise location within a map. These tasks are usually connected into one problem called SLAM (Simultaneous Localization And Mapping). SLAM techniques build a map of an unknown environment and localize the sensor in the map with a strong focus on real-time operation. In this project we implement ORB SLAM under the assumption that there exist no dynamic constraints of the robots movement and the path generated is the connection between the points of interest.

As the environment may be dynamically changing, the algorithm or the rules must be devised to ensure an optimistic collision-free path. A* algorithm is a heuristic function based algorithm that can be applied on grid/metric map for proper path planning. It calculates heuristic function's value at each node on the work area and involves the checking of too many adjacent nodes for finding the optimal solution with zero probability of collision. However, it also suffers from limitations involving processing time and work speed. Various variants of the A* algorithm have been devised in this project such Rectangular Symmetry Reduction (RSR), Jump Point Search (JPS). These modifications are focused primarily on computational time and the path optimality and will be implemented on the SLAM developed environment.

Motivation

A recent study has shown that the nurses spend about 30-35% of their time "fetching and gathering" general supplies for the hospital rooms. This makes them less available for the immediate help of the patients. A mobile robot can be used here to pre-fetch these supplies for the nurses and get them to the rooms where nurses can use them. For these problems the hospital environment can be mapped and a mobile robot can use it to navigate its way through the hospital. With this as our motivation we aim to use ORB-SLAM to map an environment using a single monocular camera so that the turtlebot can navigate its way to reach a desired location.

Chapter 2

Methodology

2.1 ORB SLAM

In this project, we plan on implementing monocular SLAM on the turtlebot, based on the work in [3],[4]. Their system is named ORB-SLAM (or ORB-SLAM2 for the recent version), and is available as an out-of-the-box SLAM library that can be used with ROS[6]. We will be using this library to generate maps to test the path planning algorithms mentioned in the following sections.
The ORB-SLAM system has three main parallel threads:
1) Tracking to localize the camera with every frame by finding feature matches to the local map and minimizing the re-projection error applying motion-only BA(Bundle Adjustment),
2) Local mapping to manage the local map and optimize it, performing local BA,
3) Loop closing to detect large loops and correct the accumulated drift by performing a pose-graph optimization. This thread launches a fourth thread to perform full BA after the pose-graph optimization, to compute the optimal structure and motion solution.

2.2 A* Algorithm

A* algorithm uses a combination of heuristic searching and searching based on the shortest path. In this algorithm, each cell in the configuration space is evaluated by the value,

$$f(v) = h(v) + g(v),$$

where $h(v)$ is heuristic distance (Manhattan, Euclidean or Chebyshev) of the cell to the goal state and $g(v)$ is the length of the path from the initial state to the goal state through the selected sequence of cells.
The paper [2] lists the some basic modifications to the A* algorithm like basic Theta*[8], Phi*[8] and JPS[5] (Jump Point Search) and implementation of RSR[1] (Rectangular Symmetry Reduction) to maps to reduce computation time for these modifications and improve their performance for real-time use.

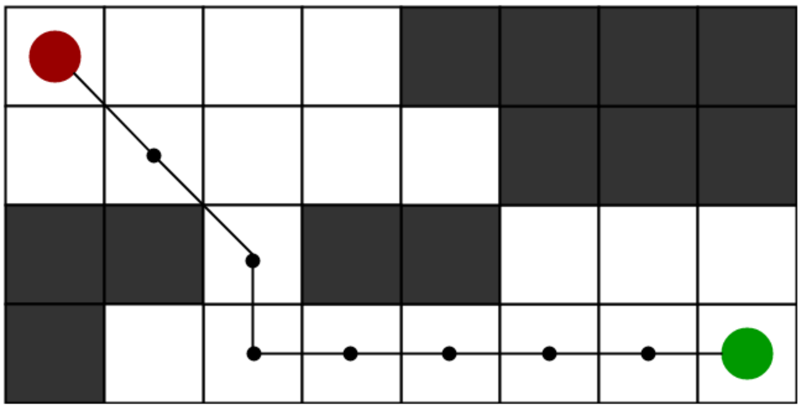

Figure 2.1: An example of A* algorithm. Green spot is start node, red is goal node.

We will be using RSR and JPS in our project and a brief overview of these methods are given in the following sections.

2.3 RSR (Rectangular Symmetry Reduction)

RSR[1] is a pre-processing step applied to the map we generate from SLAM, which identifies and eliminates symmetries in the graph. Advantages of RSR are low computational demands and possibility of combination with the family of star algorithms such as A*, Theta* etc. The symmetries in the graph are defined as the rectangles inside which only free space occurs. These are treated as one big rectangle, and path inside these rectangles is defined only by the set of the rectangle's edges. If the initial, actual or goal state is inside the rectangle, cell representing this state is temporary added as a new cell. This temporary cell is connected with all peripheral cells of the correspondent rectangle (Fig. 2.2). By this data reduction, the computational time of path planning algorithms can be reduced.

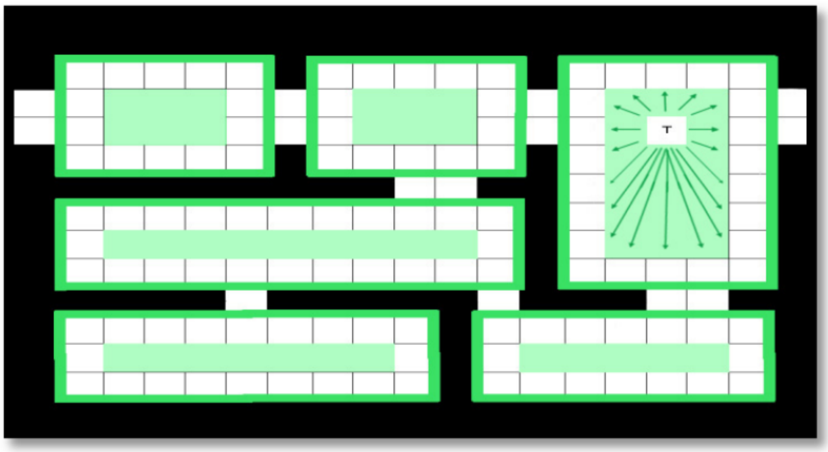

Figure 2.2: Temporal addition of the cell T with corresponding edges. The number of cells used in path planning algorithm has been reduced from 175 to 20.[2]

2.4 JPS (Jump Point Search)

Jump Point Search (JPS) [5] is an online symmetry(defined above) breaking algorithm which speeds up pathfinding on uniform-cost grid maps by "jumping over" many locations that would otherwise need to be explicitly considered. JPS requires no pre-processing and has no memory overheads. It can speed up A* search by over many orders of magnitude. This algorithm involves a few steps and there detailed explanation is skipped from this proposal. An example of path following with this method is given in Fig 2.3.

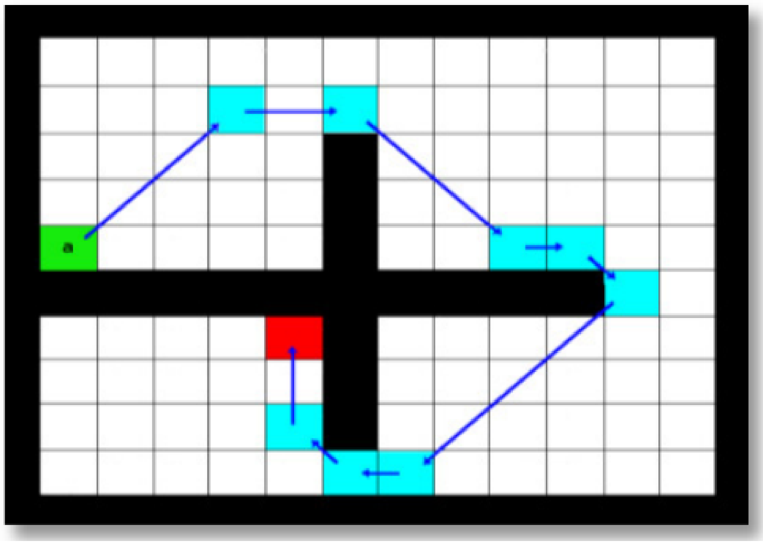

Figure 2.3: Path planning with JPS method. The initial state is green, the goal state is red, blue cells denote the jump points.[2]

2.5 TurtleBot

TurtleBot is a low-cost, personal robot kit with open-source software. TurtleBot3 is made up of modular plates that users can customize the shape. TurtleBot3 consists of a base, two Dynamixel motors, a 1,800mAh battery pack, a 360 degree LIDAR, a camera(+ RealSense camera for Waffle kit, + Raspberry Pi Camera for Waffle Pi kit), an SBC(single board computer: Raspberry PI 3 and Intel Joule 570x) and a hardware mounting kit attaching everything together and adding future sensors. We will be only employing the camera sensor for our project (if we move onto hardware implementation). For our simulation environment, we will use Gazebo with TurtleBot ROS implementation. We will be using RViz to get camera data in our simulation.

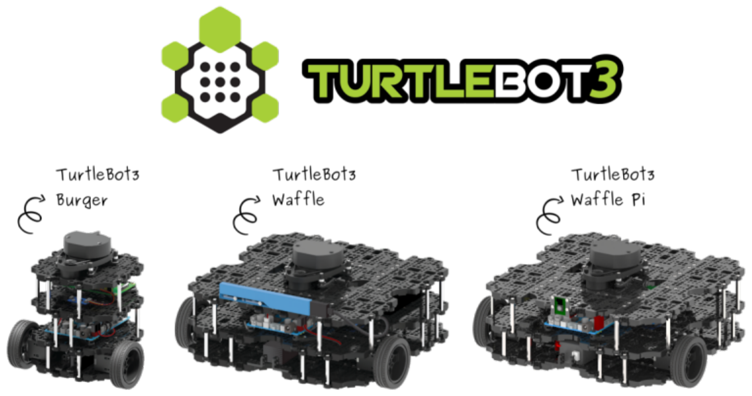

Figure 2.4: TurtleBot 3 variants

Chapter 3

Goal

1. Simulation of TurtleBot in Gazebo and implementing monocular SLAM using the ORB-SLAM2 libraries[6].
2. Implementing A* Algorithm.
3. Implementing Rectangular Symmetry Reduction (RSR).
4. Implementing Jump Point Search (JPS).
5. Doing a comparative study of modified A* with basic A* algorithm based on computation time.
6. If time permits, testing of above algorithms on hardware, i.e., the Turtlebot.

Chapter 4

Assumptions

1. The Primary assumption for ORB-SLAM to work is that the image frame contains a minimum number of features.
2. The speed of the turtle-bot is under a threshold so that there is no image-blur which can cause loss of features.
3. The environment in consideration is not dynamic.
4. The dynamics of the motion are considered ideal and no-slip condition is assumed for wheels of the turtle-bot.
5. For Algorithm implementation, the maps will be generated beforehand by manually controlling the TurtleBot.

Bibliography

[1] Harabor, D., 2012. Fast Pathfinding via Symmetry Breaking. 2013-11-20]. http://aigamedev, com/open/tutorial/symmetry-in-path finding.

[2] DuchoÅĹ, F., Babinec, A., Kajan, M., BeÅĹo, P., Florek, M., Fico, T. and JuriÅąica, L., 2014. Path planning with modified a star algorithm for a mobile robot. Procedia Engineering, 96, pp.59-69.

[3] Mur-Artal, R. and TardÃşs, J.D., 2017. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. IEEE Transactions on Robotics, 33(5), pp.1255-1262.

[4] Mur-Artal, R., Montiel, J.M.M. and Tardos, J.D., 2015. ORB-SLAM: a versatile and accurate monocular SLAM system. IEEE transactions on robotics, 31(5), pp.1147-1163.

[5] Harabor, D.D. and Grastien, A., 2011, August. Online graph pruning for pathfinding on grid maps. In Twenty-Fifth AAAI Conference on Artificial Intelligence.

[6] ORB-SLAM2, `https://github.com/raulmur/ORB_SLAM2`

[7] ORB-SLAM, `https://github.com/raulmur/ORB_SLAM`

[8] Yap, P.K.Y., Burch, N., Holte, R.C. and Schaeffer, J., 2011, October. Any-angle path planning for computer games. In Seventh Artificial Intelligence and Interactive Digital Entertainment Conference.