LTL Constrained A* Planning for Mapping an Indoor Environment

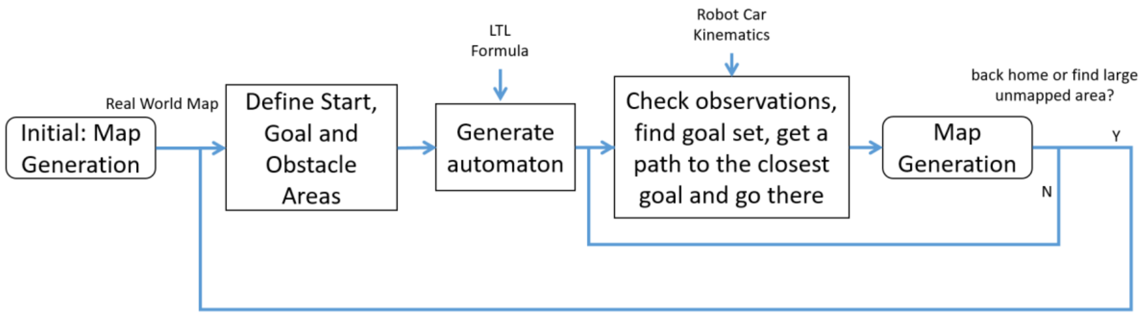
ME 734 Project report
Weipeng Wang

INTRODUCTION

ROBOT CAR

(Figure 1) A rear wheel drive car called ‘SunFounder Smart Video Car’ which I bought on Amazon, with size 9.9 x 5.2 x 2.9 inches with a single camera at the front. The movement of the wheels is controlled by two motors and a servo motor. The camera has two degrees of freedom controlled by two servo motors. Using this camera, it is possible to get a map of the indoor environment and the project aims to guide the robot car to fulfill the task of mapping the environment in an efficient way using LTL constrained A* search algorithm.

PROCESS OVERVIEW


Initial: Map Generation
Real World Map
Define Start, Goal and Obstacle Areas
LTL Formula
Generate automaton
Robot Car Kinematics
Check observations, find goal set, get a path to the closest goal and go there
Map Generation
back home or find large unmapped area?
Y
N

The process starts with the initial map preparation. The ground area in an image taken by the camera is extracted first and together with the depth information of the image, a real world map can be given for the path planning. With the start, goal and obstacle areas settled based on the map, a LTL formula generates an Automaton so that the robot car can check its state and decide the goal set that can lead it to the next state. A closest goal to the robot car’s current position is picked as the goal for the A* search algorithm and with a defined kinematics, a control strategy for the robot car to navigate and map the environment is generated. After the robot car eventually goes back home or finds large unmapped area, the process starts over again.

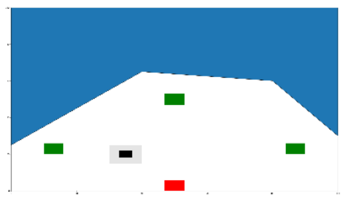
Detailed information about each section of the process are presented later in this report.

SECTION 1: MAP GENERATION

To generate a real world map for passable area in the environment with a monocular system, there are two main issues to solve: how to extract the ground area in an image and how to get the depth information of the scene. I applied machine learning method together with some data analysis techniques and the Max-Flow Min-Cut algorithm to extract the ground area from an image. The depth information is something to be further developed, I only applied dense optical flow to consecutive images to get a rough estimation of the depth of different regions in the original image.

Since this part is not directly related to the course, it will not be talked in details here. If you are interested, the code for this part as well as the document to describe it are all on my GitHub page. For the rest part of the report, it can be assumed that the map is given (although there is still work to be done about the depth estimation and in the python program I manually defined the map).

DEFINE REGIONS ON THE GIVEN MAP



(Figure 2) The red square is the start region, named ‘s’. The blue region is the unpassable region, named ‘obs’. The black region is the obstacle, named ‘obs’ as well, which is just used to test the A* search algorithm later. The shadow around the black region is the safe region in case the path gets too close to the obstacle. The green regions are goal areas, named ‘g1’, ‘g2’, ‘g3’ from left to right

The goal areas are chosen to be close to the border of the ground area (white regions) but not too close in case the uncertainty of the map caused the robot car to crash into obstacles. Essentially, I want the robot car to go to the goal areas, have a further look at the environment, make sure that there are no large unmapped area left and eventually it should go back to where it starts and keep mapping the area on the other side of the map.

SECTION 2: LTL FORMULAE AND AUTOMATON

LTL FORMULA

It is wanted for the robot car to go to goal 1, 2 and 3 to have a further look at the environment make sure there is no ground areas left undetected. After been to the three goal areas, the robot car always wants to go back to the starting point and starting to navigate the area on the opposite side of the current map. While navigating, the obstacles always need to be avoided. So, the following formulae is defined:

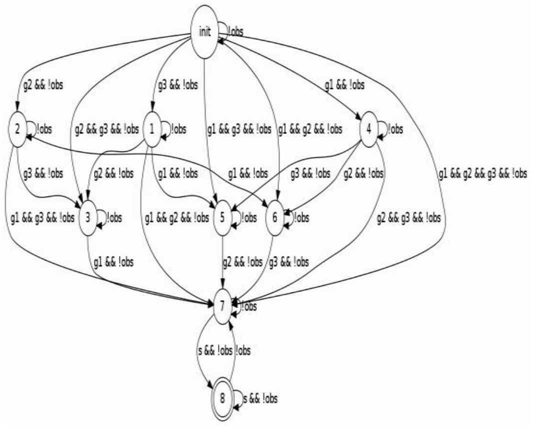
[]<>s && <>g1 && <>g2 && <>g3 && []!obs (1)

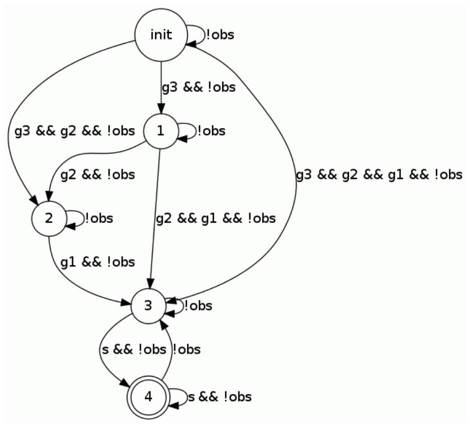
If it is wanted that the robot car navigate to the goals one by one, the formulae should be:

[]<>s && <>(g3 && <>(g2 && <>g1)) && []!obs (2)

AUTOMATON

Based on the two LTL formulae above, the corresponding automaton can be generated:





(Figure 3) Automaton for formulae (1) (Figure 4) Automaton for formulae (2)

From the automaton, it is clear that what kind of transitions can lead the robot car to go to the next state. Among these transitions, a goal set can be extracted. For example, the 'init' state in Figure 3, it is possible for it to go to state 4, 2, 1 by going to g1, g2, g3. Whenever there is a state change, there will always be a new goal set and by picking the closest goal, the A* search algorithm can be used to find the optimal path.

SECTION 3: PATH PLANNING

ROBOT CAR KINEMATICS

The robot car is set to move only forward with a constant speed and a turning angle θ from -45^o to 45^o. The robot car's kinematics is defined as following:

$$\begin{pmatrix} x_{t+1} \\ y_{t+1} \\ \varphi_{t+1} \end{pmatrix} = \begin{pmatrix} x_t \\ y_t \\ \varphi_t \end{pmatrix} + \begin{pmatrix} \tau\nu \cos\theta \\ \tau v \sin\theta \\ \theta \end{pmatrix} + uncertainty$$

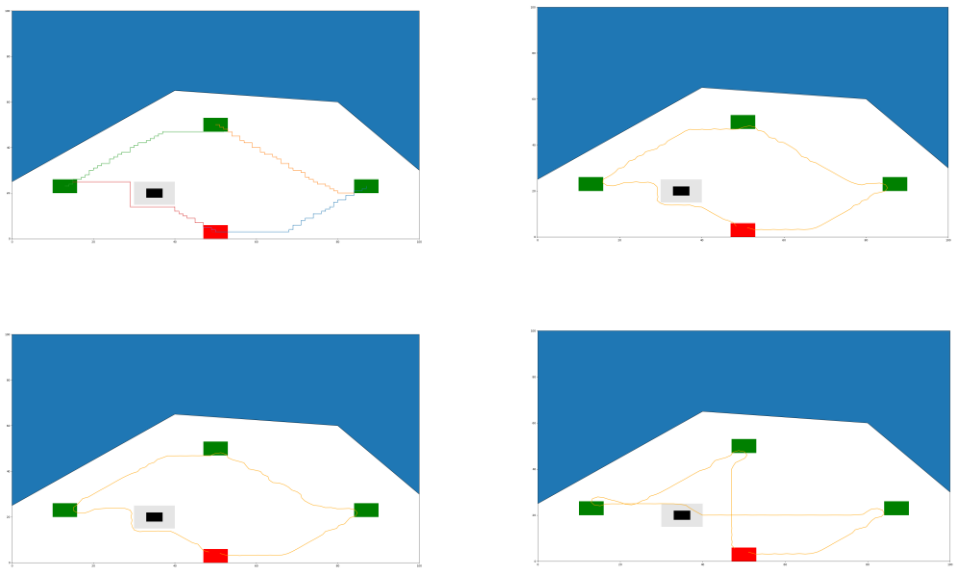
Where it has a constant speed ν, sampling period τ, head direction φ, turning angle θ and some uncertainty of the movement or environment.

Track the A* Generated Path

Given the map as is in Figure 2, make a 100*100 grid containing 10000 nodes to run the A* algorithm. Regions of obstacles are defined in this grid world so that no node in the obstacle regions can be added to the path. The algorithm can find optimal path between any two squares. Whenever there is a state change, there will be a new goal set for A* (or not if the robot car goes back to the start region) and the closest one is the goal for the A* algorithm. So the algorithm can find an optimal path to the goal region while avoiding the obstacles at the same time.

Based on the kinematics, for the robot car at a certain position in the map, 45 possible next movement (each varies by 2^o) are set and best one is picked as the one minimizes the sum of distance to the generated path and the goal. While the robot car is moving, the observations are checked. Once the robot car enters the goal region, there will be a state change immediately.

Results



(Figure 5) Upper Left: The path given by A*; Upper Right: Visit three goal regions in any order

Lower Left: Visit g1, then g2, then g3; Lower Right: Visit g2, then g1, then g3

The result shows that the robot, under its kinematics and the LTL constrains, takes the optimal path to reach all three goal regions and gets back in the end. For the last one, visit g2, then g1, then g3, the path gets close to the obstacle, but thanks to the safe region defined, it does not hit the obstacle. This indicates a drawback of the A* algorithm that the path may get too close to the obstacle.

CONCLUSION AND FUTURE WORK

This project combines the A* searching algorithm to the hybrid automaton, serving to find an optimal path to fulfill the task of mapping the indoor environment. Although due to the unfinished work about the depth estimation and the speed of the ground recognition algorithm, the method described in this report cannot be demonstrated on the robot car, it is very clear that with a fairly precise map given, the robot car, guided by the given control strategy, can navigate and map the environment in an efficient way.

In the future, the following few points may be good directions to further develop the system:

- A good algorithm to define the goal regions automatically
- Define some potentially dangerous regions (like the regions close to the border of the ground area), and when the robot car enter this kind of region, lower its speed and check the environment
- The localization of the robot car, especially when its movement has some uncertainties and the map given is not that precise
- Dealing with the problem of how to put all the map given by the map generation process together and get a full map of the entire environment
- Simulate the method on more complex environment and eventually implement it on the real robot car

All the code including the map generation part are on my GitHub Page, take a look if interested:

https://github.com/Robertwwp/Ground-Recognition