1 Introduction

In this assignment you will be working on writing Matlab code to implement planning systems that work on 2D grid like environments. For both sections the input map will be specified as a 2D logical array where the false or zero entries correspond to free cells and the true or non-zero entries correspond to obstacle cells. The goal of these planning routines is to construct a route between the specified start and destination cells.

2 Assignment

2.1 Files in this Assignment

`DijkstraGrid.m` [*] - This file contains the skeleton code for the Dijkstra planner. It is expected to return the route(path) found, along with the number of nodes expanded by the planner. Your job is to complete the function by finishing the marked section in the file.

`AstarGrid.m` [*] Similarly this file contains the skeletal implementation of the A* planner. Your job is to complete the implementation of this planner. This function is also expected to return the route(path found) and the number of nodes expanded.

`TestScript1.m` [*] This file invokes the DijkstraGrid.m and AstarGrid.m with some start and goal coordinates. Use this script to test your planners with different configurations of the environment. This file also contains the '*map*' variable that defines the environment and the obstacles in it. Vary the environments and pose challenging path-planning problems for your planners.

`evaluate.p` - Code that evaluates your implemented functions

`submit.m` - Script which calls the `evaluate` function for generating

* indicates the files you will need to complete submission result

2.2 Tasks

Part 1 : [15 points] Dijkstra's algorithm

The skeleton code provided in DijkstraGrid.m contains the following 2D arrays:

map An array of integer values which indicates the current status of each cell in the grid. Different integer values are used to indicate the different possible states listed below:

- Whether the node is part of an obstacle
- If the node is in freespace,
- Whether or not it has been visited
- Is currently on the list of nodes being considered.

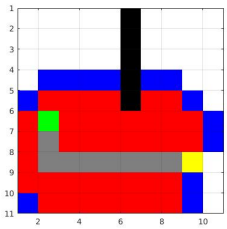

Figure 1: An Example A* result

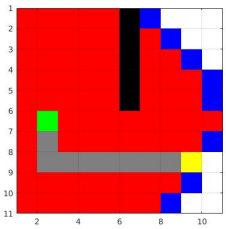

Figure 2: Example Dijkstra result

distanceFromStart This array encodes the current estimate for the distance between each node and the start node.

parent For every cell in the map this array records the index of its parent with respect to the algorithm, that is the next node along the shortest path to the start node.

On every iteration through the main loop the code finds the unvisited cell with the smallest distance value, i.e, the node that is the closest to the start node.
The part of this function that you need to write should consider the four neighbors of the cell - to the north, south east and west, and for each neighbor your function should decide if it needs to update the corresponding entries in the distanceFromStart, map and parent arrays. If you know how to use sub functions in Matlab you may find it useful to create one that you can then apply repeatedly to each of the neighboring cells, but this is not required.

When you complete the code and run the associated test script it should show you an animation (Fig.1, Fig.2) of the algorithms progress. On these plots the the *yellow* cell is the goal and the *green* cell is the start location. The *greyed* out region in the grid represents the path found by the planner. The *red* cells in the grid represent all the nodes that have been visited by the planner and maked as expanded while the *blue* cells are nodes that are currently on the list of cells that are being considered for expansion.

Part 2 : [15 points] A Star algorithm

The file AStarGrid.m contains some of the code needed to implement the A Star algorithm on a grid based environment. Your job is to complete the function by finishing the marked section of the file. As you will see the code is designed to maintain the following 2D arrays:

g This array encodes the current estimate for the distance between each node and the start node

f This array encodes the sum of the g value for each node and the value of the Heuristic function, H. The code that we have provided already computes an array, H, with appropriate heuristic function values for each node.

The *map* and the *parent* arrays carry out the same function as in DijkstraGrid.m (mentioned in previous section)

On every iteration through the main loop the code finds the unvisited cell with the smallest f value. The part of this function that you need to write should consider the four neighbors of the cell to the north, south east and west and for each neighbor it should decide if it needs to update

the corresponding entries in the f, g, map and parent arrays. If you know how to use sub functions in Matlab you may find it useful to create one that you can then apply repeatedly to each of the neighboring cells, but this is not required.

When you complete the code and run the associated test script it should show you an animation of the algorithm's progress.

Remember, the DijkstraGrid.m and the AstarGrid.m both need to return the total number of nodes visited by each of the planners. From the lectures we know which planner is more efficient and this is clearly reflected by the number of nodes traversed in order to reach the goal. This consequently makes the Dijkstra Planner much slower than A* and this is especially obvious on large maps. There is a variable called `drawMapEveryTime` that can be toggled to enable or disable the visualization of the planner's progress at each iteration.

2.3 Submission and Grading

To submit your result to our server, you need to run the command `submit` in your MATLAB command window. A script will then evaluate your planners and generate 2 output files, `AstarSubmission.mat` and `DijkstraSubmission.mat` to be uploaded to the Coursera web UI. If you pass our test cases, you will get the full score in this assignment. You may submit your result multiple times, and we will count only the highest score towards your grade.

Part	**Submitted File**	**Points**
Astar Planner	`AstarSubmission.mat`	15
Dijkstra Planner	`DijkstraSubmission.mat`	15