Project 1: A* search algorithm

PROBLEM DESCRIPTION [1]

The problem of planning a path given a **start** and a **goal** position is called the path planning problem. We often encounter such problems in the area of mobile robotics where the goal is to give the robot the ability to determine how it should navigate from its current location to some desired goal position. As such, we must consider the robot's **start** and **goal** positions as well as the geometry of the environment, i.e. the location of the obstacles, the size of the obstacles, and the such.

A common approach is to divide the workspace into cells. Once the workspace has been properly discretized, one can represent the layout of the workspace using what is called an occupancy grid. An occupancy grid is simply a data structure used to encode the layout of the workspace. One can interpret the occupancy grid as simply a map of the environment where free space is represented by clear cells and occupied space, i.e. space where obstacles reside, is represented by solid cells. Figure1(a) shows an occupancy grid for a workspace that has been tessellated into cells of varied size and shapes while Figure 1(b) shows an occupancy grid obtained using a fix grid decomposition of the workspace. From these two examples, we see that it is possible to represent any given occupancy grid as arrays of 0s and 1s where 0s denote free space and 1s denote occupied space. Furthermore, by representing our workspace using an occupancy grid, we can easily find the shortest path between any two free cells in the environment by determining the shortest sequence of free cells that connects the two cells.

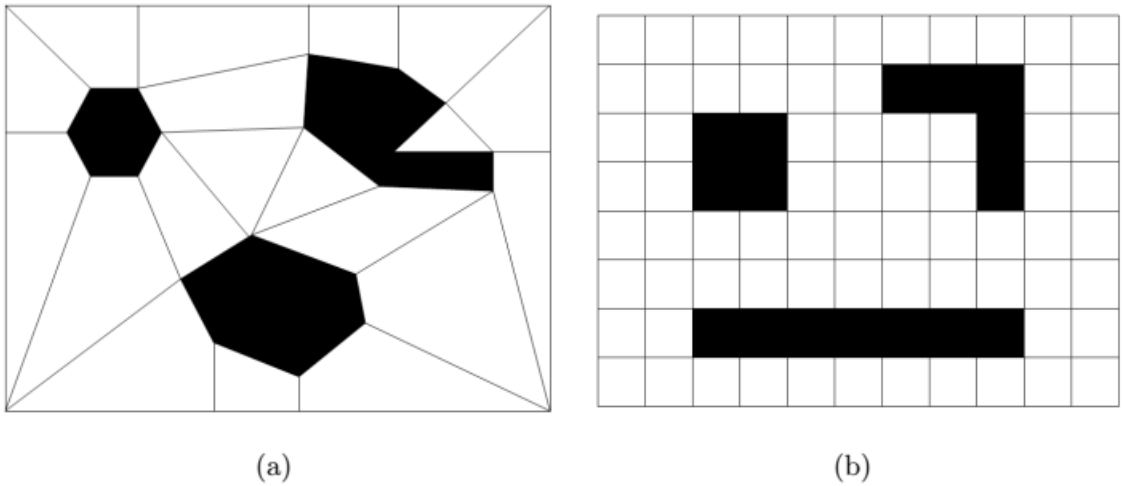

Figure 1[1]: (a) General division of an environment. Such methods will result in an occupancy grid with cells of varying sizes and shapes. (b) A grid decomposition of an environment. These results in square cells of the same dimensions.

Cost of the best route that goes through each node. And it employs this heuristic estimate when determining which node to visit next in its search process. As such, A* is an example of a best–first search algorithm [5]. Furthermore, if we choose h(x) = 0, we can show that Dijkstra's algorithm is simply a special case of A*. Lastly, A* is both complete and optimal. This means that A* will always find a path if a path exists and report failure if a path does not exist and the path that A* returns will be optimal in terms of the heuristic function.

PROGRAMMING

1. **Task 1.**

   You have to implement the A* algorithm to enable a robot to find an optimal path from its starting position to any goal location specified by a human operator for a given environment. You can code this up in **Python**. Euclid distance is used as a heuristic between a current cell and the goal. The ordered expansion of **8 adjacent cells** is shown in the figure below. (note that the orange cell is the current cell)

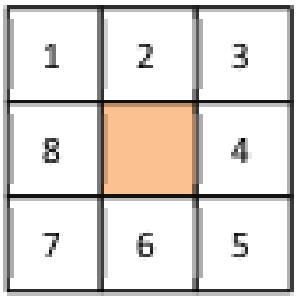

Figure 2. The ordered expansion of adjacent cells.

The syntax to run this program is <Program name> <Input> <Output>

- **Program name:** is the name of your program.
- **Input**: is input file.
- **Output:** is output file.

Both the Input format and Output file format are shown in the next section.

2. Input format and Output format.

a. Input format.

The input file is the text file with some information as a table below.

M N # the map size is M rows and N columns *Sx Sy # start position* *Gx Gy # Goal position* *//The map with M rows and N columns. 0: free cell and 1: obstacle cell. Each value is separated by a space.*	7 7 0 0 6 6 0 0 0 0 0 0 1 0 0 0 0 0 1 1 1 1 0 0 0 0 1 0 1 1 0 0 0 0 0 1 1 0 0 1 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0

b. Output format.

The output file is also the text file with the map and the optimal path.

On the one hand, if you cannot find out the path from start position to goal position. The output file has only one value (-1).

-1

On the other hand, the output file has some information as an example below.

```
8 # the number of steps to reach the goal.
(0,0) (1,1) (2,2) (3,3) (4,4) (5,4) (6,5) (6,6)
# the position of each step in the optimal path.
# the map with the optimal path.
S - - - - - o
- x - - - o o
o o x - - - o
- o o x - - -
- o o - x o -
- o - - x o -
- - - - - x G
```

From start (S) to goal (G), we reach the goal by 8 steps from the **start** position. "-" are free cells, "o" are obstacle cells. "x" are cells which belong to the optimal path.

3. **Task 2.**

   You have to implement a python application with GUI (graphical user interface) to show step by step to reach the goal from start position of the robot.

4. **For your report.**

   The report must have all requirements below.

   - Name and Student ID.
   - How many percentages of the project did your group finish?
   - The contribution of each member in this project.
   - Test case (at least 3 different test cases with different features.)
   - The diagram which shows the flow of your program.
   - Description of the data structure which used in your program.
   - Description of the algorithm.
   - References.
   - What I need to do to run your code. Please submit your code with your report.

5. **Regulations.**

   - **Maximum members in a group: 3.**
   - Deadline: 2 weeks (Following our Moodle).
   - Program name and folder name are StudentID1_StudentID2_....Project1.

- The submission included source and report.
    - **Source**: your programs.
    - **Report:** the report file (.docx, .pdf, .doc)
- Before submitting the project 1. Please compress all file into one file (.zip/ .rar). Example: 1510001_1510002_Project1.zip/.rar

References

[1] Ani Hsieh, “A* Search Algorithm”, course E28, Mobile Robotics.
[2] Maxim Likhachev, Geoff Gordon and Sebastian Thrun, "ARA*: Anytime A* with Provable Bounds on Sub-Optimality," Advances in Neural Information Processing Systems 16 (NIPS), MIT Press, Cambridge, MA, 2004.