AI 3603 Artificial Intelligence: Principles and Techniques
Homework 1: Search Algorithm Due Oct. 9th 20:00 p.m.

Adhere to the Code of Academic Integrity. You may discuss background issues and general strategies with others and seek help from course staff, but the implementations that you submit must be your own. In particular, you may discuss general ideas with others but you may not work out the detailed solutions with others. It is never OK for you to see or hear another student's code and it is never OK to copy code from published/Internet sources. Moss (Measure Of Software Similarity) will be used for determining the similarity of programs to detect plagiarism in the class (`https://theory.stanford.edu/~aiken/moss/`). If you encounter some difficulties or feel that you cannot complete the assignment on you own, discuss with your classmates in Discussion forum on Canvas, or seek help from the course staff. Please complete the homework **individually**.

When submitting your assignment, follow the instructions summarized in Section 7 of this document.

1 Introduction

1.1 Description

In this homework, you will develop a path planning framework for a service robot in the unknown environment using A* algorithm. Suppose an autonomous robot DR20 exploring in an unknown scene, as shown in Fig. 1. The global map is unavailable for the robot at the beginning of exploration, but a laser scanner mounted on robot's body is utilized to scan the obstacles around the robot. The robot can gradually build the map as it explores, util it reaches the goal, which is shown as the gray block in Fig. 1.
This homework will be implemented in the CoppeliaSim simulator. A quick-start guide is provided for CoppeliaSim in the file list. For more details, please refer to `https://www.coppeliarobotics.com` and `https://www.coppeliarobotics.com/helpFiles/index.html`.

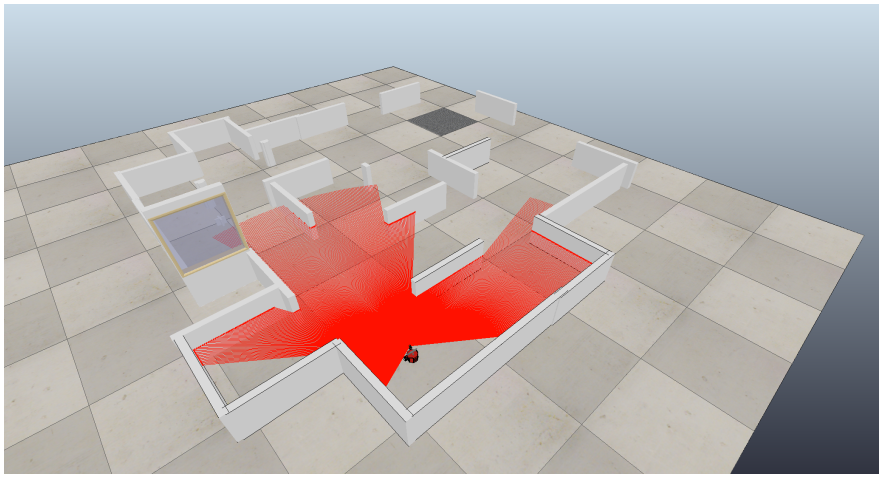


Figure 1: DR20 in the unknown environment.

1.2 Provided File List

The provided files for this homework including:

- 0-CoppeliaSim Installer (Download from jBox):

  Installer package for Windows: `https://jbox.sjtu.edu.cn/l/E1CNnE`

Installer package for Ubuntu 18.04: `https://jbox.sjtu.edu.cn/l/c1crqM`

Installer package for Ubuntu 20.04: `https://jbox.sjtu.edu.cn/l/b17kUk`

Installer package for Ubuntu 22.04: `https://jbox.sjtu.edu.cn/l/N1qS4d`

Installer package for MacOS 10.13: `https://jbox.sjtu.edu.cn/l/LFkJvZ`

Installer package for MacOS 10.15: `https://jbox.sjtu.edu.cn/l/J11ZRb`

- 1-HW1_Assignment.pdf: The introduction and description of homework 1.
- 2-CoppeliaSim_Tutorial.pdf: The quick-start guide for CoppeliaSim, including the introduction of the APIs.
- 3-Report_Template: A latex template for report.
- 4-Search.ttt: The scene file in CoppeliaSim for homework 1.
- 5-Example.py: An example code for CoppeliaSim and the APIs for homework 1.
- 6-Task_1.py: The code for basic A* algorithm to be completed.
- 7-Task_2.py: The code for improved A* algorithm to be completed.
- 8-Task_3.py: The code for self-driving planning algorithm to be completed.
- DR20-API: API for DR20 in CoppeliaSim.

1.3 Submission File List

- Task_1.py: The completed code for A* algorithm.
- Task_2.py: The completed code for improved A* algorithm.
- Task_3.py: The completed code for self-driving planning algorithm. Provide the folder "DR20API" if you revise the code in it.
- Task_1.mp4: Demo video for A* algorithm.
- Task_2.mp4: Demo video for improved A* algorithm.
- Task_3.mp4: Demo video for self-driving planning algorithm.
- HW1_report.pdf: Report for homework 1.

2 Task 1: Basic A* Algorithm [60 points]

In this section, you will implement basic A* algorithm for the autonomous robot DR20. The coordinate of the goal is known for the robot but the global map is unavailable at the beginning. You can control the robot move forward, backward, left and right. As long as the robot enters the gray block, the mission is considered successful. To reach the goal, the robot can execute a loop similar to the following

```
Input the goal position.
Initialize the position of the robot and the map of the world.
repeat
    Plan a path based on the current map using A* algorithm.
    Move the robot along the a part of the path.
    Obtain the current position of the robot.
    Update the map based on the current information of laser scanner and get the updated map.
until the robot reaches the goal.
```

To implement A* algorithm, the world is discretized into a grid map, where a grid indicates a 0.1m×0.1m block in the world. The coordinate system of the world is defined in Fig. 2. The configuration of the robot is indicated as (x_r, y_r), where x_r and y_r are two integers indicating the coordinate of the robot in the grid map.

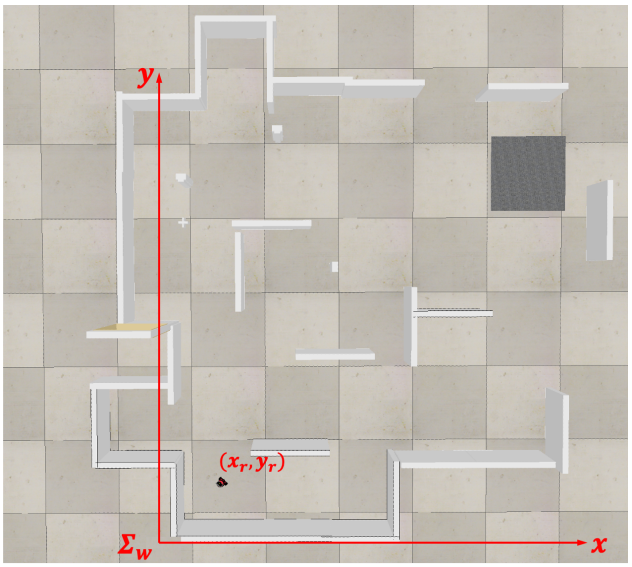


Figure 2: Coordinate system of the world.

3 Task 2: Improved A* Algorithm [30 points]

Although A* algorithm can theoretically plan an optimal (or shortest) path, you may find that the path by A* algorithm in Section 2 may not be good enough, such as frequent turns and close distance to the obstacles. Hence in this section, you will implement an improved A* algorithm to improve the performance of A* planner. Specifically, you are asked to formulate and incorporate the following factors into A* algorithm:

- Possibility of moving towards upper left, upper right, bottom left, bottom right, shown as Fig. 3(a);
- The distance between the robot and the obstacles to avoid collision, shown as Fig. 3(b);
- The cost of steering to reduce the frequency of turning, shown as Fig. 3(c).

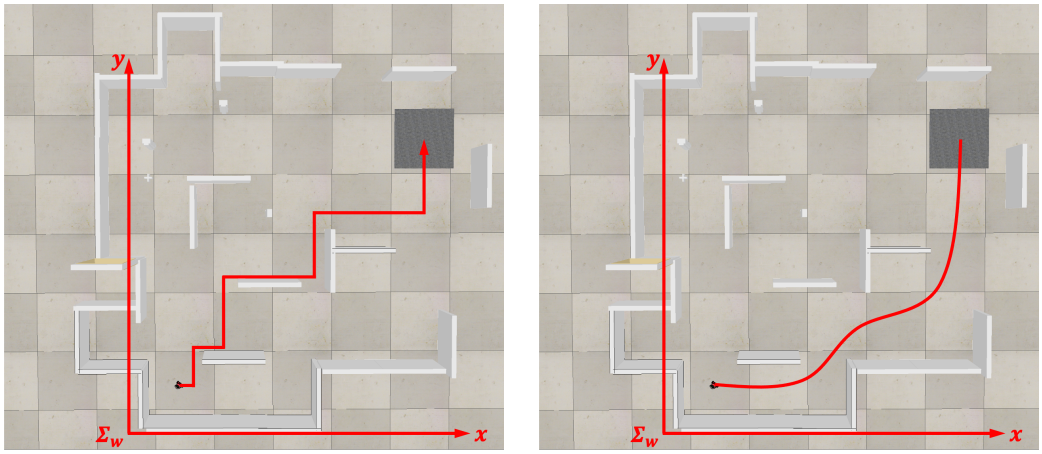
(a) (b) (c)

Figure 3: Improvements on A* algorithm.

4 Task 3: Path Planning for Self-driving [10 points]

Due to the discretization of the map, the path obtained by the methods in Section 2 and 3 are not smooth. However, smoothness is usually required to guarantee the comfort and the energy efficiency for self-driving cars. In this section, you are asked to improve the smoothness of the path. In this task, we have no restrictions on code changes, that is, you can modify the existing framework, re-write the existing functions and APIs, and use other libraries. Some keywords for reference are provided below. You can implement one of these methods or conduct your own ideas.

- Polynomial interpolation
- Bézier curve
- Hybrid A* algorithm
- Lattice planner



(a) (b)

Figure 4: Smoothness of the path.

5 Code, Demo Video and Report

Code: For Task 1 and 2, you can edit your code between "### START CODE HERE ###" and "### END CODE HERE ###" in Task_1.py and Task_2.py. Please ***DO NOT*** revise other parts of the code in Task 1 and 2.
For Task 3, you can edit your code anywhere, including "DR20API". Provide the folder "DR20API" if you revise the code in it for Task 3.
Only common python packages are available for basic function and data structure, such as numpy, pandas, math, queue, heapq and matplotlib. Any code related to the algorithm implementation cannot be completed through packages.
The code block to be completed is described below.

```
### START CODE HERE ###
# This code block is optional. You can define your utility function and class in this block
                                                    if necessary.



###  END CODE HERE  ###
```

```
def A_star(current_map, current_pos, goal_pos):
    """
    Given current map of the world, current position of the robot and the position of the
                                                    goal,
    plan a path from current position to the goal using A* algorithm.

    Arguments:
    current_map -- A 120*120 array indicating current map, where 0 indicating traversable
                                                    and 1 indicating obstacles.
    current_pos -- A 2D vector indicating the current position of the robot.
    goal_pos -- A 2D vector indicating the position of the goal.

    Return:
    path -- A N*2 array representing the planned path by A* algorithm.
    """
```

```
def Improved_A_star(current_map, current_pos, goal_pos):
    """
    Given current map of the world, current position of the robot and the position of the
                                                    goal,
    plan a path from current position to the goal using improved A* algorithm.

    Arguments:
    current_map -- A 120*120 array indicating current map, where 0 indicating traversable
                                                    and 1 indicating obstacles.
    current_pos -- A 2D vector indicating the current position of the robot.
    goal_pos -- A 2D vector indicating the position of the goal.

    Return:
    path -- A N*2 array representing the planned path by improved A* algorithm.
    """
```

```
def reach_goal(current_pos, goal_pos):
    """
    Given current position of the robot,
    check whether the robot has reached the goal.

    Arguments:
    current_pos -- A 2D vector indicating the current position of the robot.
    goal_pos -- A 2D vector indicating the position of the goal.

    Return:
    is_reached -- A bool variable indicating whether the robot has reached the goal, where
                                                    True indicating reached.
    """
```

Demo video: Please record your screen to show the results of your A* planner, improved A* planner and self-driving planner. The videos should be in mp4 format and a 10 MB max in total (you can speed up and compress the videos), named as Task_1.mp4, Task_2.mp4 and Task_3.mp4.
Report: Summarize the process and results of the homework using the provided report template in English, including but not limited to:

- The description of the implementation of A* and improved A* algorithm.
- The formulation and implementation of the heuristic function of improved A* algorithm.
- The comparison between A* and improved A* algorithm, such as computational time, safety, optimality, etc.
- The description, formulation and implementation of your algorithm to improve the smoothness of the path, and the comparison with basic A* and improved A* algorithm.

6 Discussion and Question

You are encouraged to discuss your ideas, and ask and answer questions about homework 1. A new post for this assignment "Homework 1 Discussion" is opened in the Discussion Forum on Canvas. If you encounter any difficulty with the assignment, try to post your problem for help. The classmates and the course staff will try to reply. `https://oc.sjtu.edu.cn/courses/49425/discussion_topics/107259`.

7 Submission Instructions

1. Zip your python code files **Task_1.py**, **Task_2.py**, **Task_3.py**, demo videos **Task_1.mp4**, **Task_2.mp4**, **Task_3.mp4** and the report file **HW1_report.pdf** to a zip file named as **HW1_Name_ID.zip**.

2. Upload the file to "Homework 1: Search Algorithm" on Canvas:
   `https://oc.sjtu.edu.cn/courses/49425/assignments/182362`.
   Due: Oct. 9th 20:00 p.m.