



Birla Institute of Technology & Science, Pilani

Work-Integrated Learning Programmes Division

MTech in Artificial Intelligence and Machine Learning

S2_2025-2026, AIMLCZG557/AECLZG557

Assignment 1 – PS6 - Rescue Robot - [Weightage 12%]

Read through this entire document very carefully before you start!

Drone Path Planning using A* Search

An autonomous drone must determine the shortest and safest path from a starting location to a destination in a grid-based terrain environment. The drone can move in eight possible directions:

- East
- West
- North
- South
- North-East
- North-West
- South-East
- South-West

The terrain consists of:

- 0 → Blocked regions/Obstacles
- 1 → Valid flying regions

The objective is to implement the *A* Search Algorithm* using both traditional and problem-specific heuristic functions to optimize drone navigation.

Input Type

Binary matrix, source indices, destination indices

```
terrain = [[1, 1, 1, 1, 1, 1, 1],  
          [1, 0, 0, 1, 0, 0, 1],  
          [1, 1, 1, 1, 1, 0, 1],  
          [0, 1, 0, 0, 1, 1, 1],  
          [1, 1, 1, 1, 1, 0, 1],  
          [1, 0, 0, 1, 1, 1, 1],  
          [1, 1, 1, 1, 0, 1, 1]]
```

start = (0, 0)

end = (6, 6)

Logic / Search Technique

A* Search Algorithm

Heuristic Functions

1. Euclidean Distance Heuristic

Used to estimate the geometric shortest distance between the current node and the goal node.

$$h(n) = \sqrt{(x_{goal} - x_n)^2 + (y_{goal} - y_n)^2}$$

2. Problem-Specific Obstacle Heuristic

Design a custom heuristic based on the **number of obstacles encountered near the current path or node**. The heuristic should penalize routes passing through highly obstructed regions.

Example idea:

$$h'(n) = h(n) + \alpha \times O(n)$$

Where:

- $h(n)$ = Euclidean distance heuristic
- $O(n)$ = Number of nearby obstacles encountered
- α = Obstacle penalty factor

This heuristic helps the drone:

- Avoid densely blocked regions
- Improve flight safety
- Reduce collision risk

- Find smoother and more efficient paths
- a. Implement the A* Search Algorithm for drone navigation.
- b. Design and compare:
 - i. Euclidean heuristic
 - ii. Obstacle-aware heuristic

Output Type

Set of pairs of integers representing the optimal path followed by the drone.

Example

```
[(0, 0),
 (1, 0),
 (2, 1),
 (2, 2),
 (2, 3),
 (3, 4),
 (4, 4),
 (5, 5),
 (6, 6)]
```

Test Case 1

Input

```
terrain = [[1, 1, 1, 1, 1, 1, 1],
           [1, 0, 0, 1, 0, 0, 1],
           [1, 1, 1, 1, 1, 0, 1],
           [0, 1, 0, 0, 1, 1, 1],
           [1, 1, 1, 1, 1, 0, 1],
           [1, 0, 0, 1, 1, 1, 1],
           [1, 1, 1, 1, 0, 1, 1]]
```

```
start = (0, 0)
```

```
end = (6, 6)
```

Expected Output

```
[(0, 0),
 (1, 0),
 (2, 1),
 (2, 2),
 (2, 3),
 (3, 4),
 (4, 4),
 (5, 5),
 (6, 6)]
```

Test Case 2

Input

```
terrain = [[1, 1, 1, 0, 1, 1],
           [1, 0, 1, 0, 1, 1],
           [1, 1, 1, 1, 1, 0],
           [0, 1, 0, 1, 1, 1],
```

```
[1, 1, 1, 1, 0, 1],  
[1, 0, 1, 1, 1, 1]]
```

```
start = (0, 0)
```

```
end = (5, 5)
```

Expected Output

```
[(0, 0),  
(1, 0),  
(2, 1),  
(2, 2),  
(3, 3),  
(4, 3),  
(5, 4),  
(5, 5)]
```

Deliverables:

- PDF document **designPSXX_<group id>.pdf** detailing your solution.
- **[Group id] _Contribution.xlsx** mentioning the contribution of each student in terms of percentage of work done. Columns must be “Student Registration Number”, “Name”, “Percentage of contribution out of 100%”. If a student did not contribute at all, it will be 0%, if all contributed then 100% for all.
- **inputPSXX.txt** file used for testing
- **outputPSXX.txt** file generated while testing
- .py file containing the python code. Create a single notebook. Do not fragment your code into multiple files
- Zip all of the above files including the design document in a folder with the name:
 - **[Group id] _A1_PSXX_XXXXXXXXXX.zip** and submit the zipped file.
 - Group Id should be given as **Gxxx** where **xxx** is your group number. For example, if your group is 26, then you will enter **G026** as your group id.

Instructions:

- It is compulsory to make use of the data structure(s) / algorithms mentioned in the problem statement.
- Ensure that all data structure insert and delete operations throw appropriate messages when their capacity is empty or full. Also ensure basic error handling is implemented.
- For the purposes of testing, you may implement some functions to print the data structures or other test data. But all such functions must be commented before submission.

- Make sure that you read, understand, and follow all the instructions
- Ensure that the input, prompt and output file guidelines are adhered to. Deviations from the mentioned formats will not be entertained.
- The input, prompt and output samples shown here are only a representation of the syntax to be used. Actual files used to evaluate the submissions will be different. Hence, do not hard code any values into the code.
- Please note that the design document must include:
 - One alternate way of modeling the problem with the performance implications.
 - Writing a good technical report and well documented code is an art. Your report cannot exceed 4 pages. Your code must be modular and quite well documented.
 - You may ask queries in [this dedicated discussion section](#). Beware that only hints will be provided and queries asked **in other channels like MS Teams, emails, Taxila inbox will not be responded to.**

Deadline

1. The strict deadline for submission of the assignment is **8th June 2026 11:55 PM IST (Monday)**.
2. The deadline has been set considering extra days from the regular duration in order to accommodate any challenges you might face. No further extensions will be entertained.
3. Late submissions will be evaluated as below. Submissions made between:
 - a. 8th June 2026 11:56PM to 9th June 2026 11:55PM → Deduct 2M for late submission and evaluation for 10 marks.
 - b. 9th June 2026 11:56PM to 10th June 2026 11:55PM → Deduct 6M for late submission and evaluation for 6marks.
 - c. Beyond 10th June 2026 11:55PM → No evaluations, 0 marks will be awarded.

How to submit

- This is a group assignment.
- Each group has to make one submission (only one, no resubmission) of solutions.
- Each group should zip all the deliverables in one zip file and name the zipped file as [Group id] _A1_PSXX_XXXXXXXXXX.zip and submit the zipped file.
- Group Id should be given as Gxxx where xxx is your group number. For example, if your group is 26, then you will enter G026 as your group id.

- Assignments should be submitted via Taxila > Assignment section. Assignments submitted via other means like email etc. will not be graded.

Evaluation

- The assignment carries **12 Marks**.
- Grading will depend on:
- Fully executable code with all functionality working as expected
 - Well-structured and commented code
 - Accuracy of the design document.
 - Every bug in the functionality will have negative marking.
 - Marks will be deducted if your program fails to read the input file used for evaluation due to change / deviation from the required syntax.
- We encourage students to take the upcoming assignments and examinations seriously and submit only original work. Please note that plagiarism in assignments will be taken seriously. Refer to the detailed policy [here](#).
- Source code files which contain compilation errors will get at most 25% of the value of that question.

Readings

Artificial Intelligence: A Modern Approach Textbook by Peter Norvig and Stuart J. Russell, 4th Edition.