**Problem 1 report**

1. **Estimated** **traversability map**

To obtain a traversability map that describes the crossable area for the rover, the local deviation from flatness of the given original image has to be measured. Utilizing the *stdfilt* command, the standard deviation of the 3-by-3 neighborhood around the corresponding pixel in the original map is obtained:

Moon = imread('moon.png');

MoonFil = stdfilt(Moon);

Assuming that the pixels with standard deviation value higher than 1.2 marks the uncrossable area. And the pixels of the crossable area is marked as 0 using

MoonBW = MoonFil > 1.2;

and the *MoonBW* contains the estimated traversability map (see Figure. 1), where white area refers to the crossable and the black uncrossable.

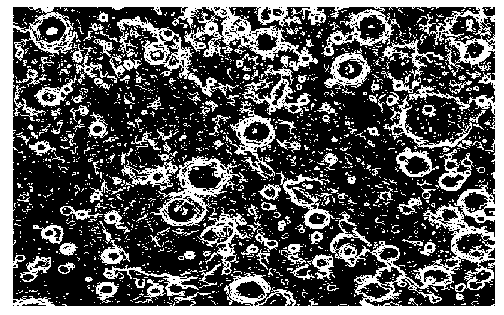


Figure. 1. The estimated traversability map.

1. **Implementation of Dijkstra’s algorithm**

Given the initial and final position of the rover, the Dijkstra’s algorithm is implemented to plan a shortest path that only lies on the crossable positions *Q*. The crossable positions *Q* on the map *MoonBW* are indexed using the indexes of pixels by

Q = find(MoonBW==0);

The cost is defined as the total distance of the planned path. The detailed code of the Dijkstra’s algorithm is given in *Dijkstra\_Optimized.m*. The initial and final position are chosen as [460, 30] and [310, 440], and the path planning result is shown in Figure. 2



Figure. 2. The path planning result using the Dijkstra’s algorithm.

Observing the Figure. 2, the planned path passes around the uncrossable area (dark colored) and finally reaches the final position with a total distance of 258.8061. The path seems the shortest and can ensure the safety of the rover.

1. **Implementation of A\* with a proposed heuristic**

A heuristic *h* is defined as the Euclidean distance between all positions and final positions, which is used to adjust the weights of the positions. With the assistance of the adjusted weights, the A\* algorithm prefer to choose a next position that is closer to the final position. When using the heuristic, the efficiency of the algorithm is improved.



Figure. 3. The path planning result using the A\* algorithm.

The initial and final position are also chosen as [460, 30] and [310, 440], which is the same as those for Dijkstra’s algorithm. The path planning result is shown in Figure. 3

Observing the Figure. 3, A\* algorithm with the proposed heuristic can also effectively plans a path with a total distance of 258.8061, which allows the rover to avoid the uncrossable area and reach the final position successfully.

1. **Comparison and discussion**

Comparing with the Figure. 2, the path in Figure. 3 seems much straighter than that in Figure. 2, thought they obtain the same cost. The execution time of the A\* algorithm is 571.58 [s], while the execution time of the Dijkstra’s algorithm is 714.13 [s]. The algorithm needs more computational efforts when using a heuristic; however, the A\* algorithm is more effective than Dijkstra’s algorithm with less time. The comparison details are given in Table. 1. Above all, the A\* algorithm with the proposed heuristic is better than the Dijkstra’s algorithm.

Table. 1. Comparison between A\* algorithm and Dijkstra’s algorithm.

|  |  |  |
| --- | --- | --- |
|  | Distance | Execution time |
| Dijkstra’s algorithm | 258.8061 | 714.13 |
| A\* algorithm | 258.8061 | 571.58 |