AUTONOMOUS ROBOTS

Lab report 1: Potential Functions

Author: **RAABID HUSSAIN**

Objective:-

The objective of the lab-work was to get familiarized with the basic forms of potential functions. We were to implement brush-fire and wave-front planner algorithms using MATLAB. For both of these algorithms, different maps were provided to test on.

Introduction:-

Path planning is an important feature of robots performing autonomous navigation allowing them to find the optimal trajectory to the goal position. There are different parameters that can be optimized to produce the ideal trajectory. These may include distance, number of turns etc. A path planning algorithm should always find a path (if it exists) in finite amount of time.

In path planning it is necessary for the robot know a map of the environment and to be aware of where it is in the map. A representation of the environment map can be achieved by potential functions. The aim in this approach is find a complete path between the starting and the goal position while avoiding obstacles. The values of the potential functions can be thought to represent energy, gradient of which yields force that will ultimately be used to drive the robot to the goal position. The starting position and the obstacles are represented as highest potentials whereas the goal position is represented as the lowest potential. This potential difference produces the repulsive forces, moving the robot to its goal position as everything in the world prefers to be in a state of low energy. The robot always follows the negative potential gradient.

In this experiment, two potential functions are implemented using MATLAB: brush-fire and wave-front algorithms. It is conventional to discretize the map first and then find the potential at each location.

Methodology:-

Brushfire Algorithm:

In implementation of the brushfire algorithm, we were provided with a discretized map in which the obstacles were represented by 1 and the free areas by 0. We were to represent the free areas by their respective potentials. The starting position of the robot is provided as input argument to the brushfire function. In brushfire, the potential decreases as the robot moves away from the obstacles.

The simplest approach was to start a counter at 2 which would represent the potential value at pixels. For each counter value, all the pixels were gone through. If there was a pixel, with one less value than the counter value, in the neighborhood of the current pixel and the pixel value in the map was zero then the

counter value was assigned as the pixel value in that pixel in the map. 8 connectivity neighborhood was used here instead of 4 to allow for lesser potential values and smoother distribution of the potential. Similarly all pixels/discretized locations in the map were checked for the same counter value. This counter values are used as labels for each location representing their potential. After scan of the entire map, the counter value was incremented and the process repeated until all the locations in the map had been assigned a label for their respective potentials. The result for one such map is shown in figure 1.

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	2	2	2	2	2	1	1	2	2	2	2	2	2	1	1	2	2	2	1
1	2	3	3	3	2	1	1	2	3	3	3	3	2	1	1	2	3	2	1
1	2	3	4	3	2	2	2	2	3	4	4	3	2	1	1	2	3	2	1
1	2	3	3	3	3	3	3	3	3	3	4	3	2	1	1	2	3	2	1
1	2	3	2	2	2	2	2	2	2	3	4	3	2	1	1	2	3	2	1
1	2	3	2	1	1	1	1	1	2	3	4	3	2	1	1	2	3	2	1
1	2	3	2	1	1	1	1	1	2	3	4	3	2	2	2	2	3	2	1
1	2	3	2	2	2	2	1	1	2	3	4	3	3	3	3	3	3	2	1
1	2	3	3	3	3	2	1	1	2	3	3	3	2	2	2	3	3	2	1
1	2	3	4	4	3	2	2	2	2	3	3	2	2	1	2	3	3	2	1
1	2	3	3	3	3	3	3	3	3	3	2	2	1	1	2	3	3	2	1
1	2	2	2	2	2	2	2	2	2	2	2	1	1	1	2	2	2	2	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 1: Output of the brushfire algorithm function for a small map

Please note here that the areas close to the obstacles should be represented as the highest potentials. However this is not the case here. So to represent the potentials in a graphical representation, the values were inverted. This means that to obtain the actual potential, the potential label for each location was subtracted from maximum potential value.

Wave-front Planner Algorithm:

In implementation of the wave-front planner algorithm, we were provided with the same discretized map in which the obstacles were represented by 1 and the free areas by 0. We were to represent the free areas by their respective potentials. The starting and goal positions of the robot are provided as input arguments to the wave-front planner function. In wave-front planner, the potential decreases as the robot moves away from the starting position. The obstacles in the map have the highest potential followed by the starting location. In this algorithm, the potentials are highly dependent on the starting position, so even if the map remains the same, the potentials need to be re-calculated to deal with new/different starting and goal positions.

The approach for computing the wave front planner was very similar to the brushfire approach. A similar counter was initialized to represent the potential at the particular location. Simply, the starting position of the robot was assigned a label value of 2 and the counter was started at 3. The same procedure was repeated as in the brush-fire algorithm until all the pixels had been assigned a label for their potential. Representing the starting position as 2 was the key here as doing so, enabled us to use the same approach as in the brushfire algorithm and it turned out to be a matter of copy and paste to implement this algorithm. A sample map for a small map and a certain starting position is shown in figure 2.

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	13	13	13	13	13	1	1	14	14	15	16	17	18	1	1	22	22	22	1
1	12	12	12	12	12	1	1	13	14	15	16	17	17	1	1	21	21	21	1
1	11	11	11	11	11	11	12	13	14	15	16	16	16	1	1	20	20	20	1
1	10	10	10	10	10	11	12	13	14	15	15	15	15	1	1	19	19	19	1
1	9	9	9	9	10	11	12	13	14	14	14	14	14	1	1	18	18	19	1
1	8	8	8	1	1	1	1	1	13	13	13	13	14	1	1	17	18	19	1
1	7	7	7	1	1	1	1	1	12	12	12	13	14	15	16	17	18	19	1
1	6	6	6	6	6	7	1	1	11	11	12	13	14	15	16	17	18	19	1
1	5	5	5	5	6	7	1	1	10	11	12	13	14	15	16	17	18	19	1
1	4	4	4	5	6	7	8	9	10	11	12	13	14	1	16	17	18	19	1
1	3	3	4	5	6	7	8	9	10	11	12	13	1	1	17	17	18	19	1
1	2	3	4	5	6	7	8	9	10	11	12	1	1	1	18	18	18	19	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 2: Output of the wave-front planner algorithm function for a small map

This algorithm is actually used to plan the path for the robot. So, the next thing was compute the trajectory for the robot depending on the starting and the goal position. This was also fairly simple. A dynamic 2D array was created to represent all the points in the path. A point to notice here is that since we can easily find out the number of locations/pixels in the trajectory by subtracting the potential of the goal and the starting position, a static array could have been easily used in order to speed up the time. However, I went with the choice of dynamic array as implementing a dynamic array is fairly simple in MATLAB and the time difference was not that big on testing.

The goal position was input as the last element in the list. The neighbors of the goal position were checked to see which ones had lower label value than the label value of the current pixel. This lower label position was input to the 2d array. Preference was given to the 4 connectivity neighbors as in practice it is easier to turn 90 degrees rather than 45 degrees. The process was repeated until the starting position was reached. This approach will always converge to the starting position as potential labels increase with respect to the staring position and all the pixels are connected to it. A typical trajectory output for the above map can be seen in figure 3.

13	12	11	11	11	11	11	11	10	9	8	8	8	8	8	7	6	5	4	3
2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	18	18	18

Figure 3: Trajectory produced for the small map

Experimental Results and Simulations:-

After successfully implementing the algorithms, it was important to represent their output in a serviceable manner. A MATLAB script was created that loads a map and calls both the functions and represents their output is a graphical representation. To represent the results, different illustrations were used:

First black and white image is produced that represents the input image in which the white areas represent the walls and the obstacles whereas the black regions represent the free areas for which potential needs to be determined. Next a grayscale image is produced to represent the potentials for both the algorithms. Here whiter regions represent higher potential areas and vice versa. This is followed by a 3D representation of the potentials and the trajectory produced by the wave-front planner for the given inputs. The results for different maps are represented in the following figures:

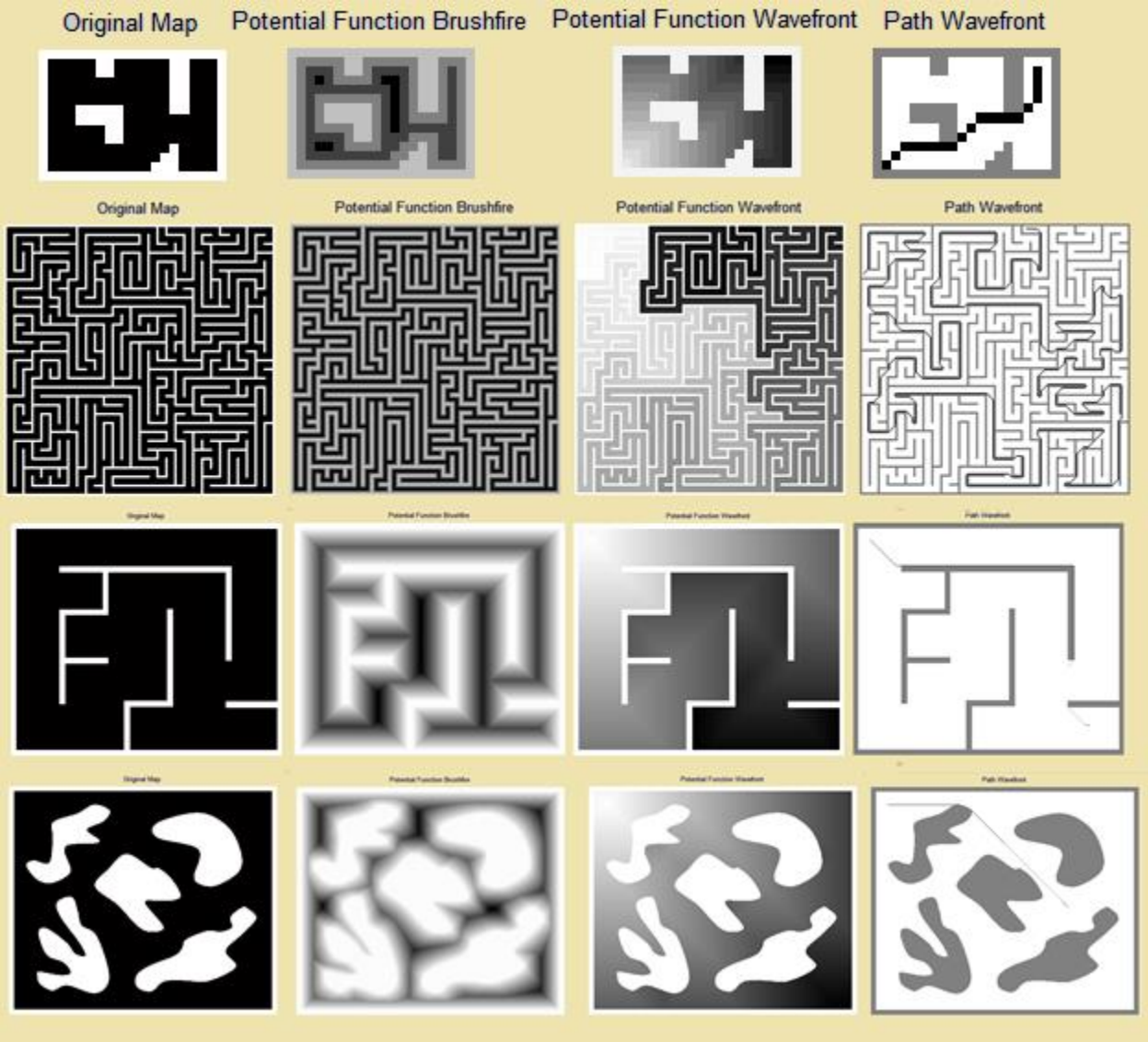

Figure 4: Output of the MATLAB script for 4 different maps (from left to right): Original Map; Potential produced by brush-fire algorithm; potential produced by wave-front planner algorithm; trajectory produced by the wavefront planner

Please note that in some of the figures, the path from the wave-front planner appears broken. This is just due to the resolution issues and if we zoom on the actual images, the completed path can be easily seen. Also, the trajectories may differ depending on the preferences of the turns, the basic path will always be the same.

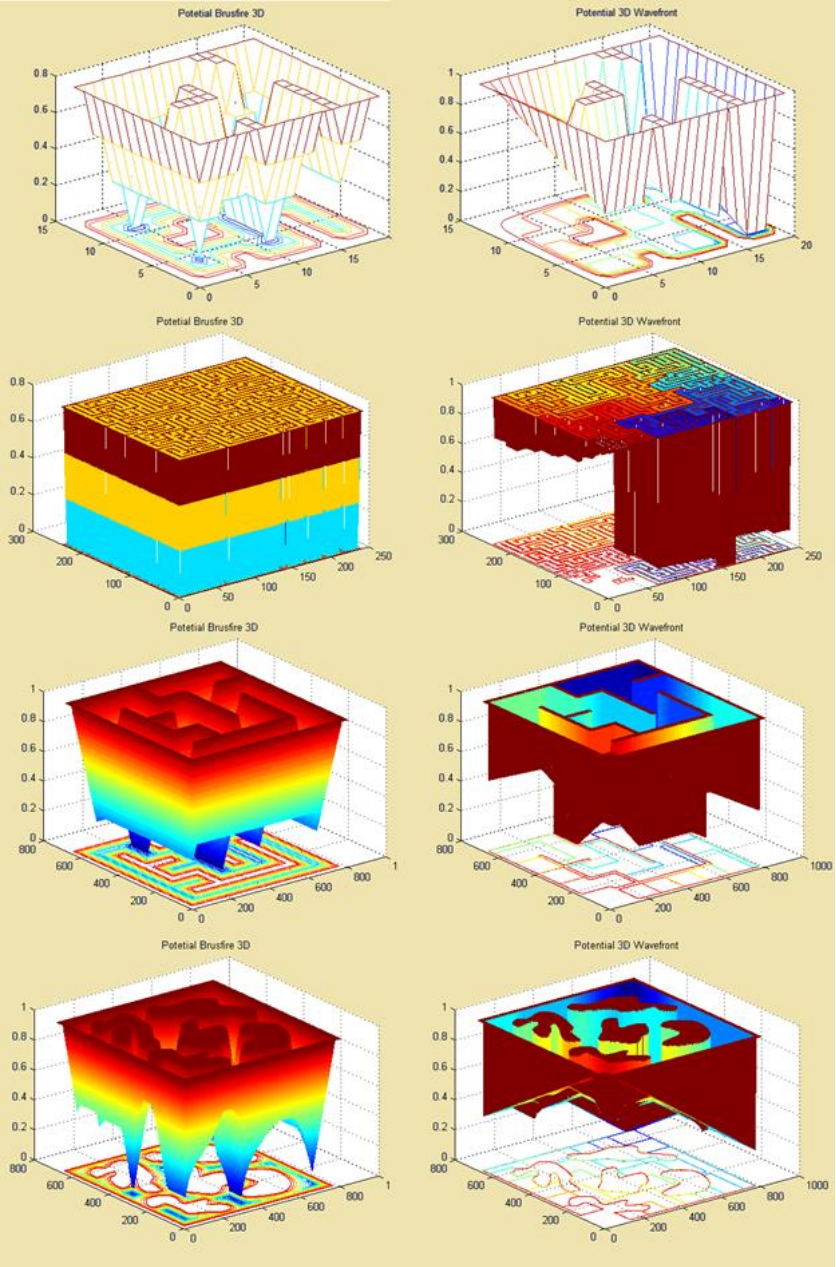

Figure 5: 3D representation output of the potentials of the MATLAB script for 4 different maps (from left to right): Brush-fire algorithm; Wave-front planner algorithm

Conclusion:-

In this lab work, basic path planning algorithms were implemented using MATLAB. Potential functions were used in these algorithms to define a possible path for the robot to follow. Since these were simple algorithms, no major problems were faced and the entire programming part was completed during the lab hours and the coming week was used to write the report. The lab was a great learning experience for the start of motion planning algorithms.