UNIVERSITAT DE GIRONA

AUTONOMOUS ROBOTS

LABORATORY REPORT

Potential Functions

Author
Mohit VAISHNAV

Supervisor
Dr. Perez MARC
CARRERAS

March 13, 2018

1 Introduction

Autonomous robots have to develop the capacity of navigation and avoid any kind of obstacles. In some case robot has the complete knowledge of environment while in few others just the idea about the goal. Hence path planning is required to take the robot from one position to a particular location. One such algorithm is known as $PotentialFieldMethod$.

This lab was meant to understand techniques of path planning based on potential functions. Herein we have implemented the algorithm namely $Bushfire$ and $Wavefrontplanner$.

2 Environment

According to the lab sheet given, we have to design the functional implementation of these two algorithms and test them on different environmental setup with varied size of matrices already provided. All the implementation has been carried out in $MATLab$ setup.

First section of the implementation concerns with the $Bushfire$ algorithm which works on the repulsive potential method. Output of this function returns the $ValueMap$ which are labelled as the potentials gradient across the matrices circumferencing the given matrix following with we can know of the path with the minimal value.

Moving on to the next section, we are required to implement $Wavefront$ planner algorithm to compute the optimal path towards the goal.

3 Implementation

3.1 Bushfire

After struggling to find the optimum algorithm which could run fast enough to run on any kind of environment. I have created an iterating loop having the range of 1000 (assuming there could be at most 1000 last value loop).

The loop is starting from an initial position $(1, 1)$ and connectivity of 8 point. To check the condition of infinite loop, I have kept a flag value with the number as the number of 0 available in the matrix. Once the iterator checks the value being equal to flag value ($Temp_count$ in the code) it comes out of loop using *break* command.

3.2 Wavefront

Implementing this program was a tricky one because I had the starting position from which potential gradient has to be developed. During implementing I have designed two type of approach; while one was using just same as the above *brushfire* way where I divided the matrix in 4 different parts where a wave kind of gradient values were allotted to each pixel value.

In the second approach which is dynamic one, here the list is maintained, checking each individual value and assigning a flag value in that logical matrix, so that it is not repeated again. During the process I am also saving the path which is to be traversed to reach to goal point.

4 Code Execution

4.1 Brushfire

It takes the input as the map matrix and return the labelled map which can be displayed using various MATLab plot available. Here I have used the *heatmap* plot which displays the digits allocated to each cell of matrix.

4.2 Wavefront

There are two files as mentioned in the above section, one which follows the strategy of the *Brushfire* approach is named as $wavefront.m$ whereas the dynamic on is names as $wavefront_position.m$ (default).

5 Results

I have substantially optimized the code so that it could run without any ample lag. Below mentioned Table shows the run time in seconds for all the size of maze for different algorithms.

Run Time for Different Size Matrices (in *sec*)				
	Small	**Maze**	**BigMaze**	**ObstacleBig**
Brushfire	0.005	1.282	21.45	21.42
Wavefront	0.043	0.700	-	-

Run Time for Different Size Matrices (on *i7* processor) (in *sec*)				
	Small	**Maze**	**BigMaze**	**ObstacleBig**
Brushfire	0.012	0.37	6.16	6.38
Wavefront	0.002	0.21	-	-

6 Output

Output after execution of the code have been represented in the following figures.

7 Conclusion

Implementing this path planning basics was very much helpful to gain knowledge in building a higher system. Potential gradient tells us about the least energy path or to say most optimized path on which robot should move to have the maximum efficiency. Brushfire uses the gradient distance by taking the difference of the neighbouring cells. Wavefront finds the shortest path according to the metric available with the gradient distance and move on the lowest number available. In potential functions there is a uncertainty about reaching the global minima or to find if there are more global minima available or not.

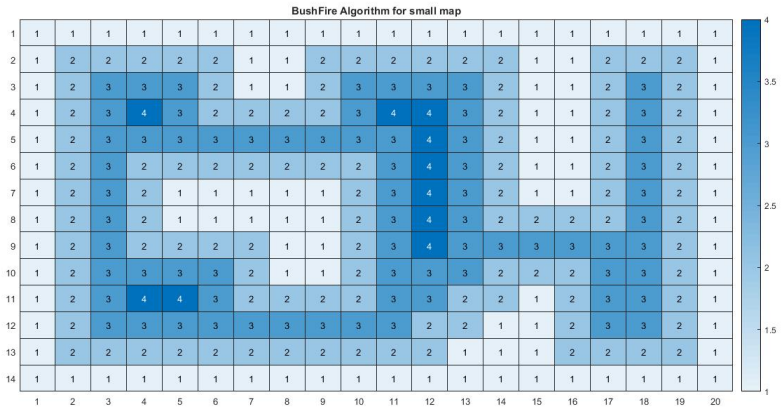

Figure 1: Small Matrix BrushFire and Path Planning

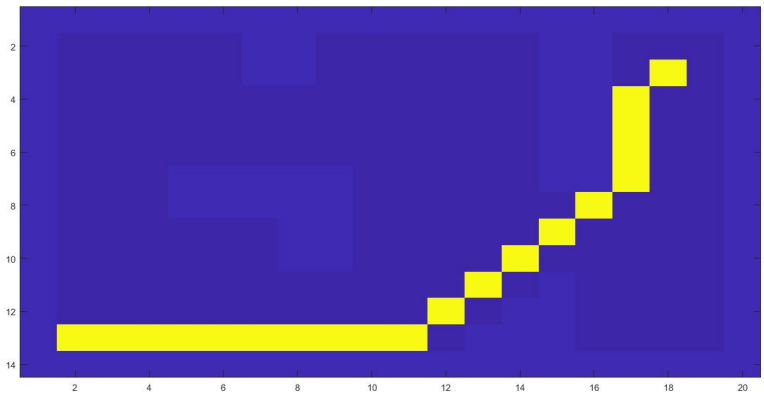

Figure 2: Path Planning for Small Matrix

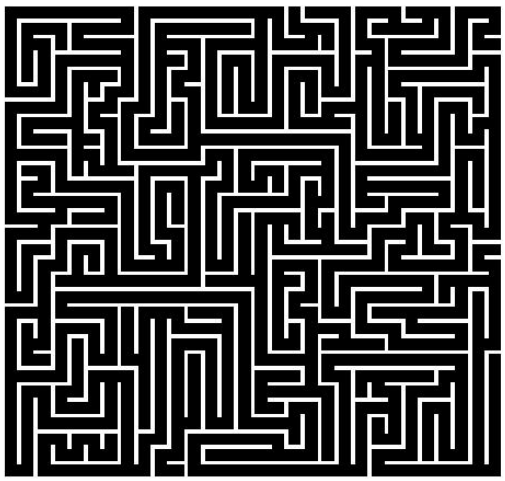

Figure 3: Maze Statement

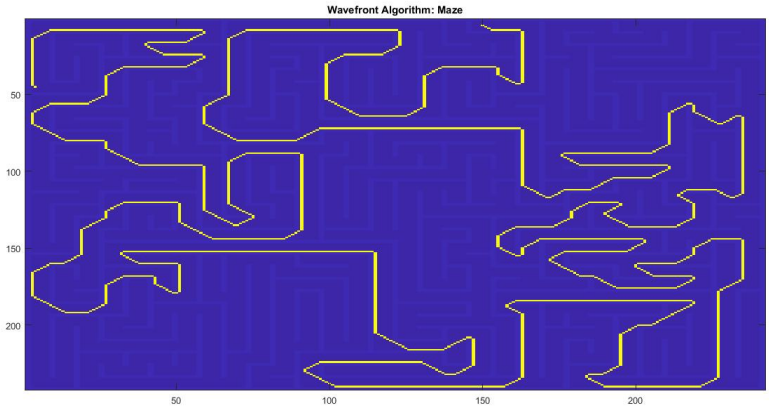

Figure 4: Path given the initial and final position

Figure 5: Big Maze Matrix

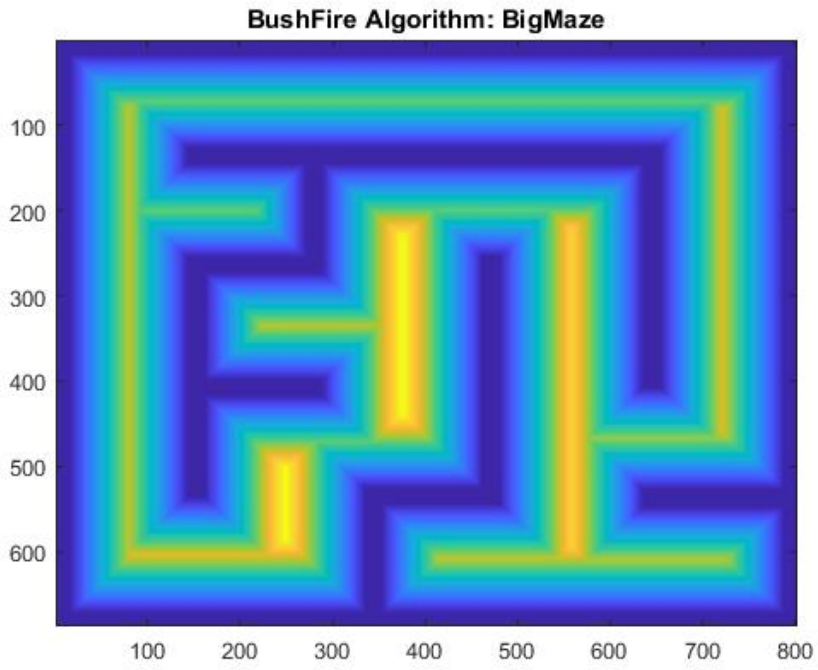

Figure 6: BrushFire display for the Big Matrix

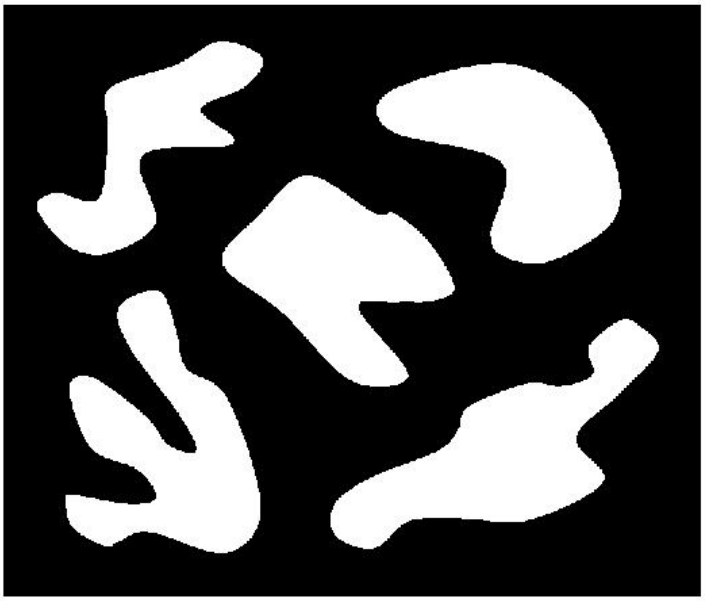

Figure 7: Big Obstacle Image

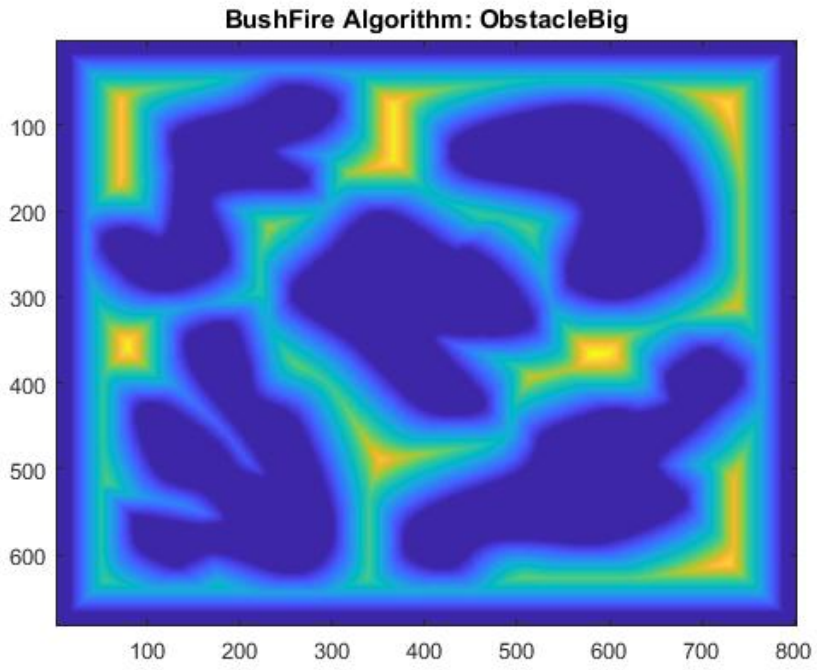

Figure 8: Big Obstacle BrushFire output