Students:
**Mir Mohibullah Sazid** [sazidarnob@gmail.com]
**Syma Afsha** [symaafsha.eece@gmail.com]

# Lab1: Potential Functions

## Introduction

The lab has been divided into two parts. In the part one, using the wavefront algorithm, the attraction function is created. Once the attraction function is created, the paths needs to find. In the potential function, it may be get trapped into local minima which needs to avoid by finding solutions. In the part two, bushfire algorithm is used to easily compute the distance to obstacles. Potential fields motion planning algorithms utilize attraction fields to move from the origin to the destination.

## Methodology

At first, using the wavefront algorithm, attraction function is build with 8 point connectivity. In this 8 point connectivity, a grid of an index has taken and it searches all the neighbouring indexes. For generating the attraction function the value was set to 2. By looping through the grid and checking the obstacles, the value of neighbouring cells are increased. Then normalization of the value between 0 and 1 is done to keep the values of all the cells in the grid map. For the resulting path, a function is created to find the optimal path from start to goal point. An array is initiated to track the whole path and another array is used to store the values of neighbouring cells. Using this array, the index of the original array is retrieved which has lowest value and shifted to that cell. Finally, the optimal path is plotted on the normalised attraction function map.

In the part two, from resulting path on the attraction function, some portion of the path is really near to the obstacle. For this reason, bushfire algorithm is implemented in the codes. Instead of avoiding obstacle, the obstacles are considered and the value of the neighbouring cells is increased. Then the repulsive function is created according to instruction manual. Finally, the repulsive function and the bushfire algorithm is merged to obtain the potential function and normalize the value between 0 and 1. After that, the resulting path function is run into the potential function map to obtain the optimum path without going too close of the obstacles.

The following flow chart shows the method used to build the code.

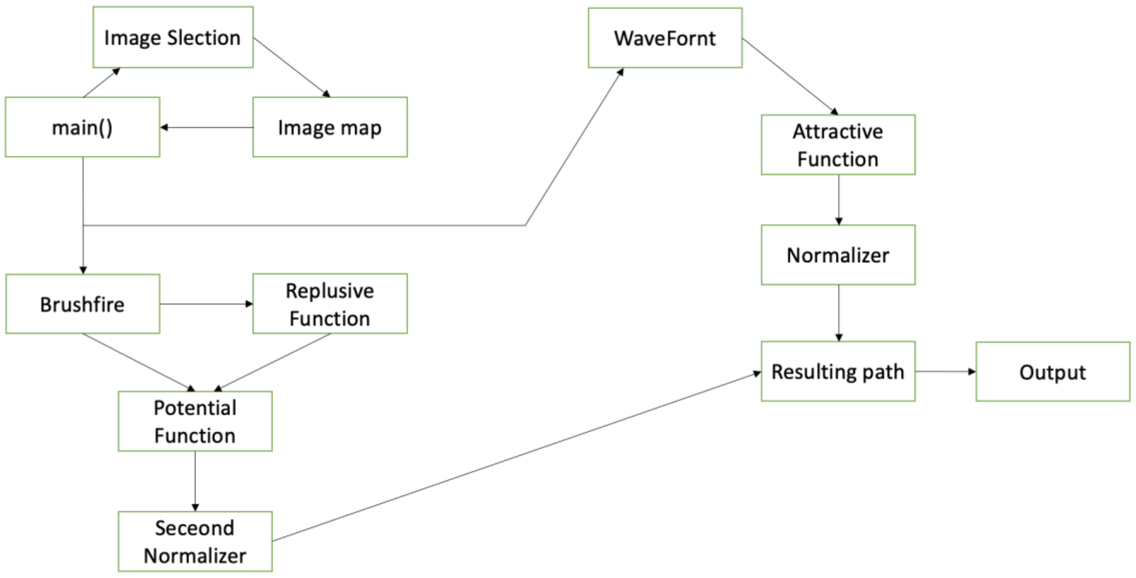

Figure 1.0: Flow chart of the code.

## Result

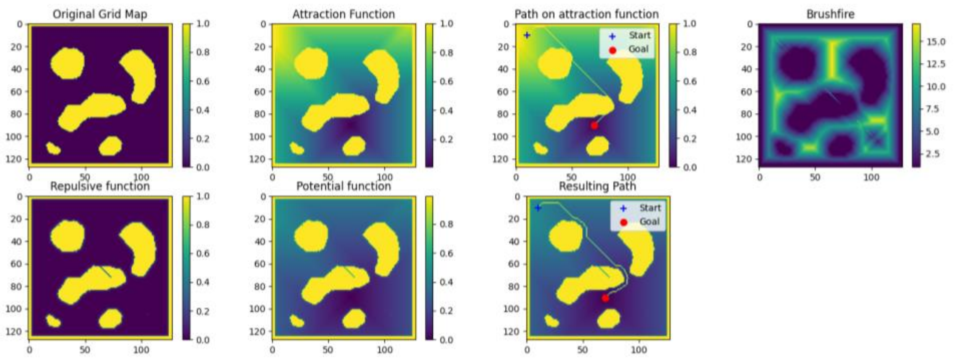

Figure 1.1: Map0

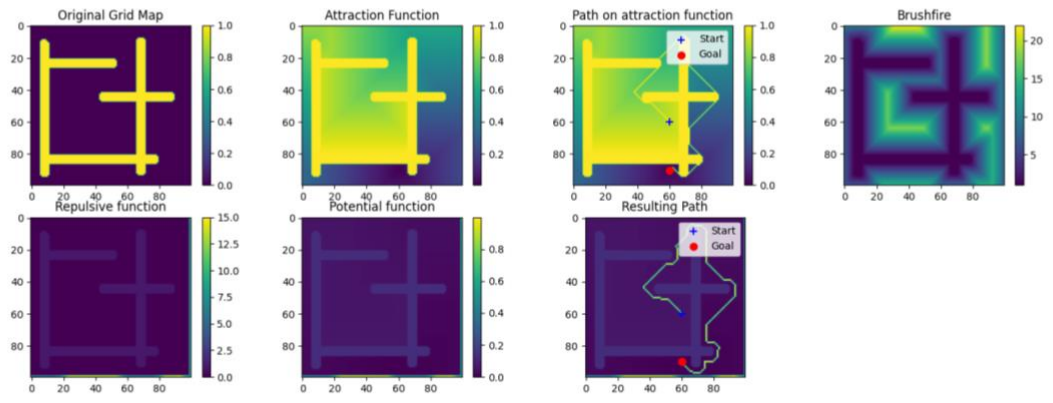

Figure 1.2: Map1

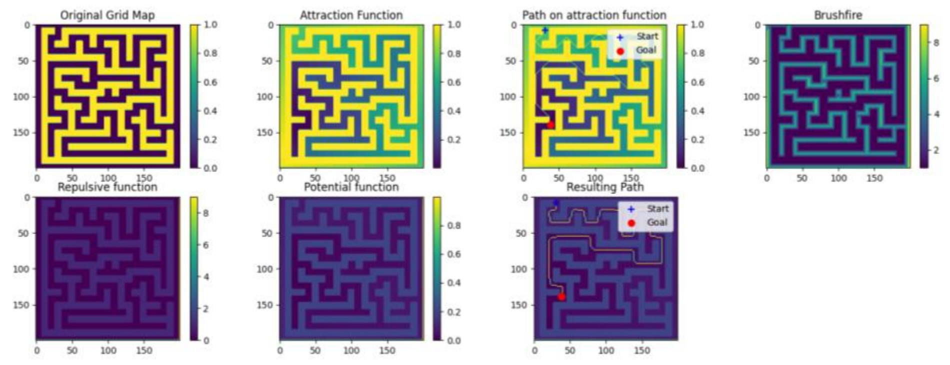

Figure 1.3: Map2

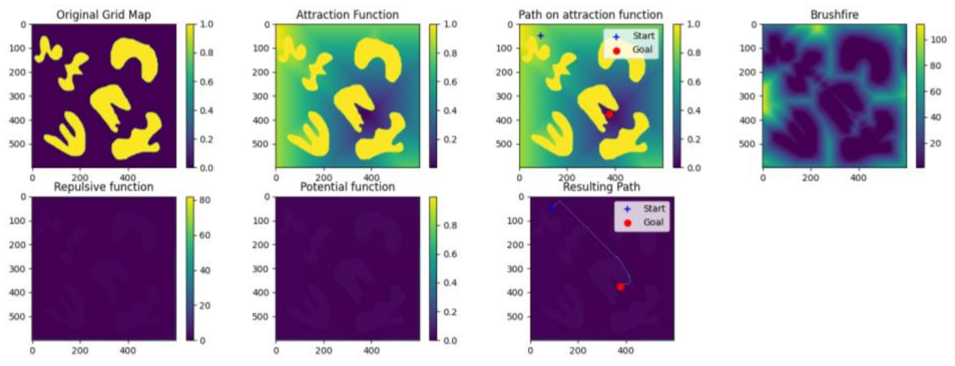

Figure 1.4: Map3

In the second set of images in Figures 1.1, 1.2, 1.3, and 1.4, it is clear that cells closer to the objective have lower values, while those further away have higher values. Once we've converted the image map to a binary representation, assigning obstacles a value of 1 and open areas a value of 0, we observe that cells located farther from the objective tend to have values near 1.

While analyzing the same set of images (Figures 1.1, 1.2, 1.3, and 1.4), we observe that our algorithm generates the path with the shortest distance. However, in an effort to select the smallest available values, it occasionally brings the robot closer to obstacles.

As depicted in the fourth image of Figures 1.1, 1.2, 1.3, and 1.4, we have integrated the brushfire algorithm to calculate distances from obstacles in order to reduce potential uncertainties and prevent collisions. In addition, we have added a repulsive function and combined these components to produce a potential function. As depicted in the final images of Figures 1.1, 1.2, 1.3, and 1.4, this potential function enables the robot to determine an optimal path while keeping a safe distance from obstacles.

## Problems and Conclusion

Throughout our algorithm development sessions, we ran into optimization obstacles. Initial mapping was founded on Euclidean distances. Although it was possible to generate these maps, the resulting route was not optimal. Due to the fact that our algorithm calculated direct distances from the objective, the robot moved in a variety of directions and occasionally navigated in circles around obstacles. To address this issue, we rewrote our algorithm and added eight-way connectivity to our path planning solution.