SUMMARY OF APPROACH

- ALGORITHM

  I have implemented a 2D weighted A* algorithm.

  Initially, a static goal is given to the planner. This goal is the last position on the trajectory of the moving target. The goal given to the planner remains constant until the robot reaches this goal. After the robot reaches this goal, the robot backtracks through the trajectory of the target (from the last position, to each preceding position on the trajectory), until it catches the robot.

  The weight is assigned to be equal to half of the collision threshold in each map.

  The algorithm is able to catch the robot in all the maps.

- DATA STRUCTURES USED

  A struct ‘node’ has been created to represent a node in the graph. This struct has the Following data members - x and y coordinates of the parent of the node, g value and f value.

  I have implemented the open list using a STL Set data structure. The open list is a set whose elements are of the type pair<float,pair<int,int>> . The first element of the pair denotes the f cost of a given node in the graph and the second element stores the (x,y) coordinates of the node. The elements in a set are in sorted order based on the value of f cost. Set provides O(logn) search, insertion and deletion.

  An implicit graph is built to store the nodes of the graph. All the nodes being created are stored in an STL ordered_map named nodes_created. The keys in the ordered map are of type pair<int,int> which stores the (x,y) coordinate of the graph node. The values in the map are of type struct node. The map data structure provides O(logn) search, insertion and deletion.

  The closed list has been implemented using a 2D array of booleans (closed_list).

- Heuristic

  I have used octile distance as the heuristic for my algorithm.

  H = D * (dx + dy) + (D2 - 2 * D) * min(dx, dy)

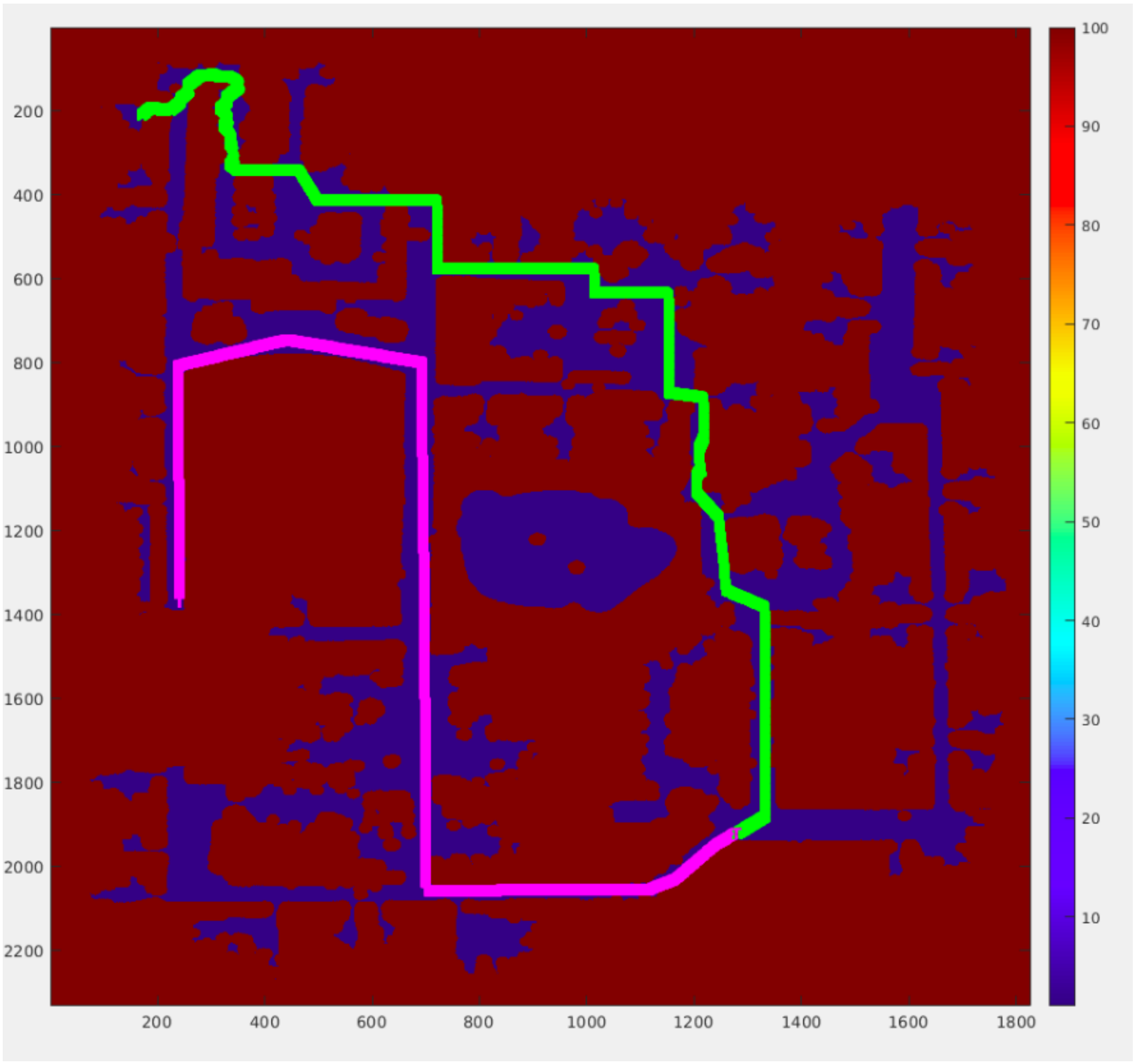
Where D2 is the cost of diagonal movement and D is the cost of horizontal/vertical movement. We take D = D2 = 1.

- HOW TO COMPILE CODE

    - I have tested my code using MATLAB in Ubuntu 16.04. The code has been tested for g++ compilation.
    - Change to the code directory in MATLAB.
    - Run - mex planner.cpp
    - Run - runtest('map.txt')

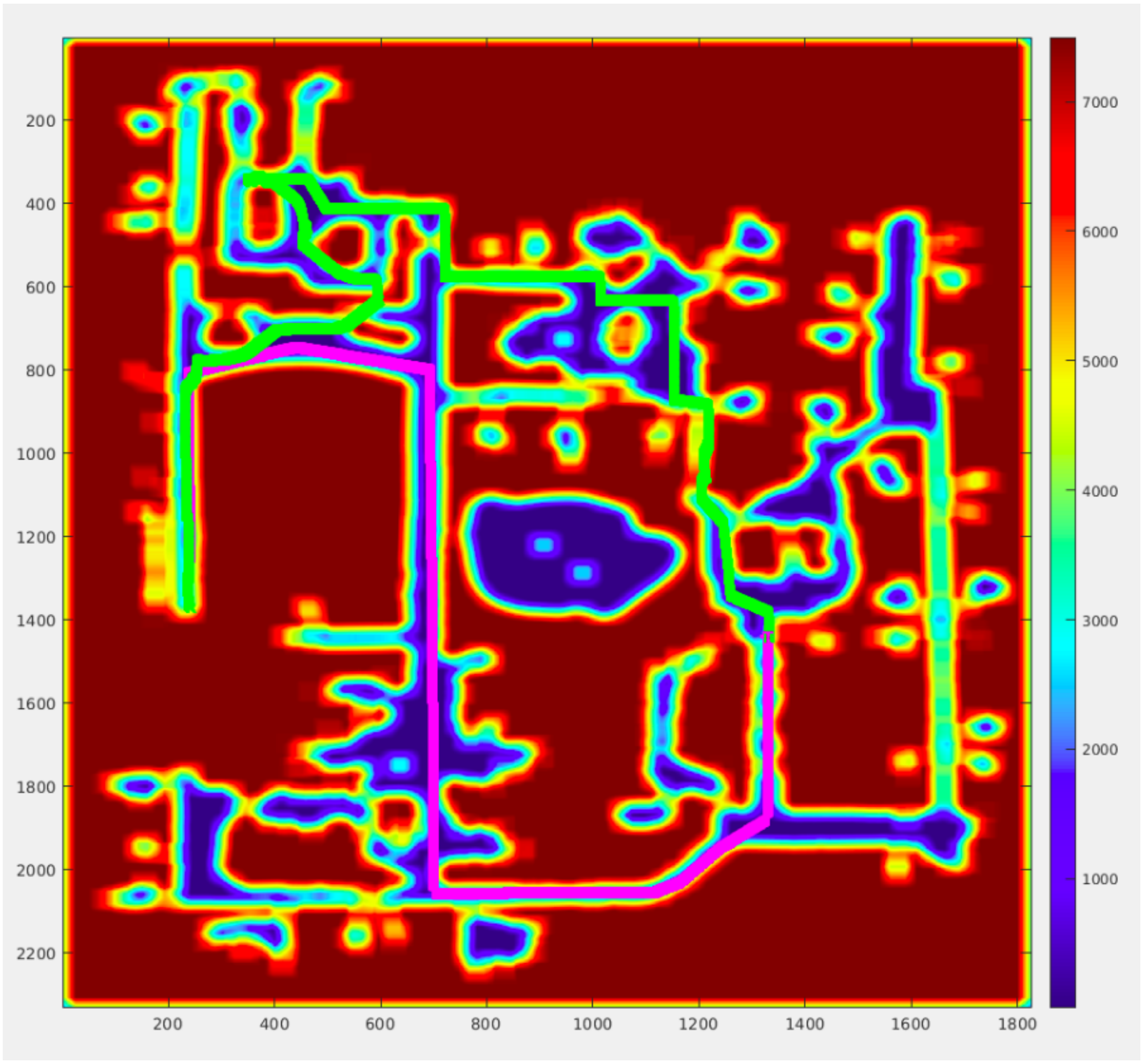
RESULTS OF TEST MAPS

1. MAP 1



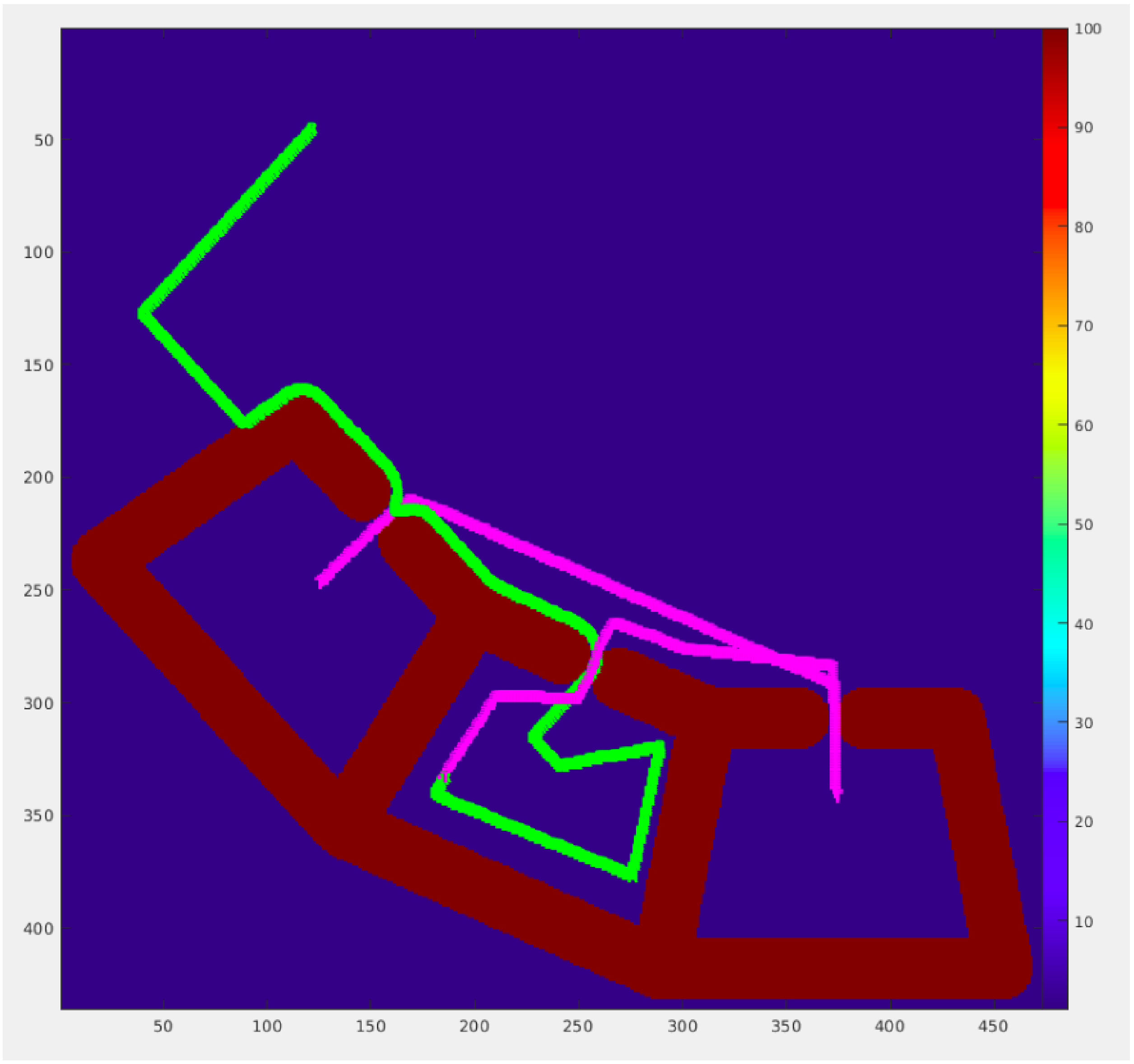
Target caught = 1
Time taken (s) = 2885
Moves made = 2883
Path cost = 2885

2. MAP 2



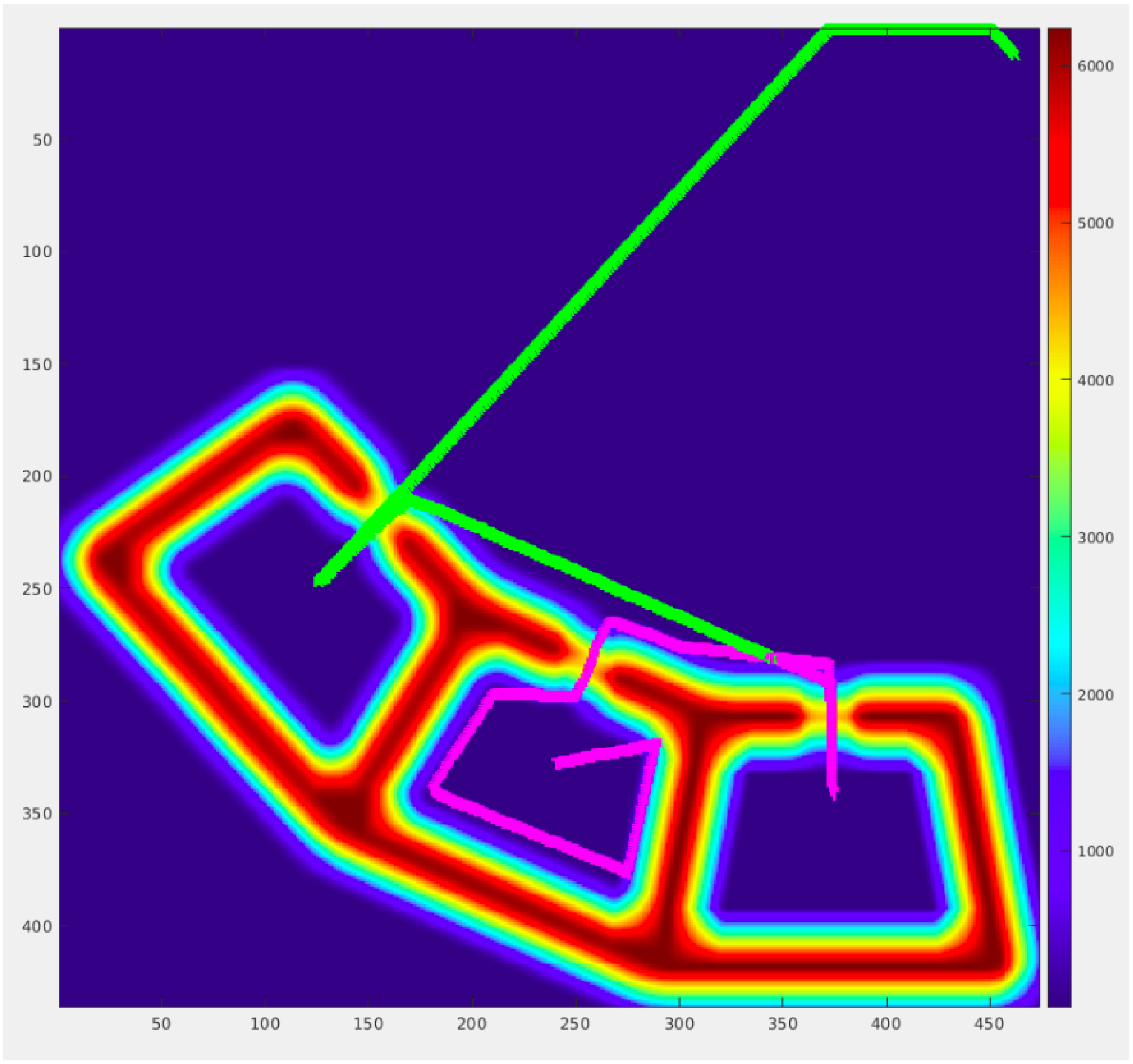
Target caught = 1
Time taken (s) = 3283
Moves made = 3280
Path cost = 6498484

3. MAP 3



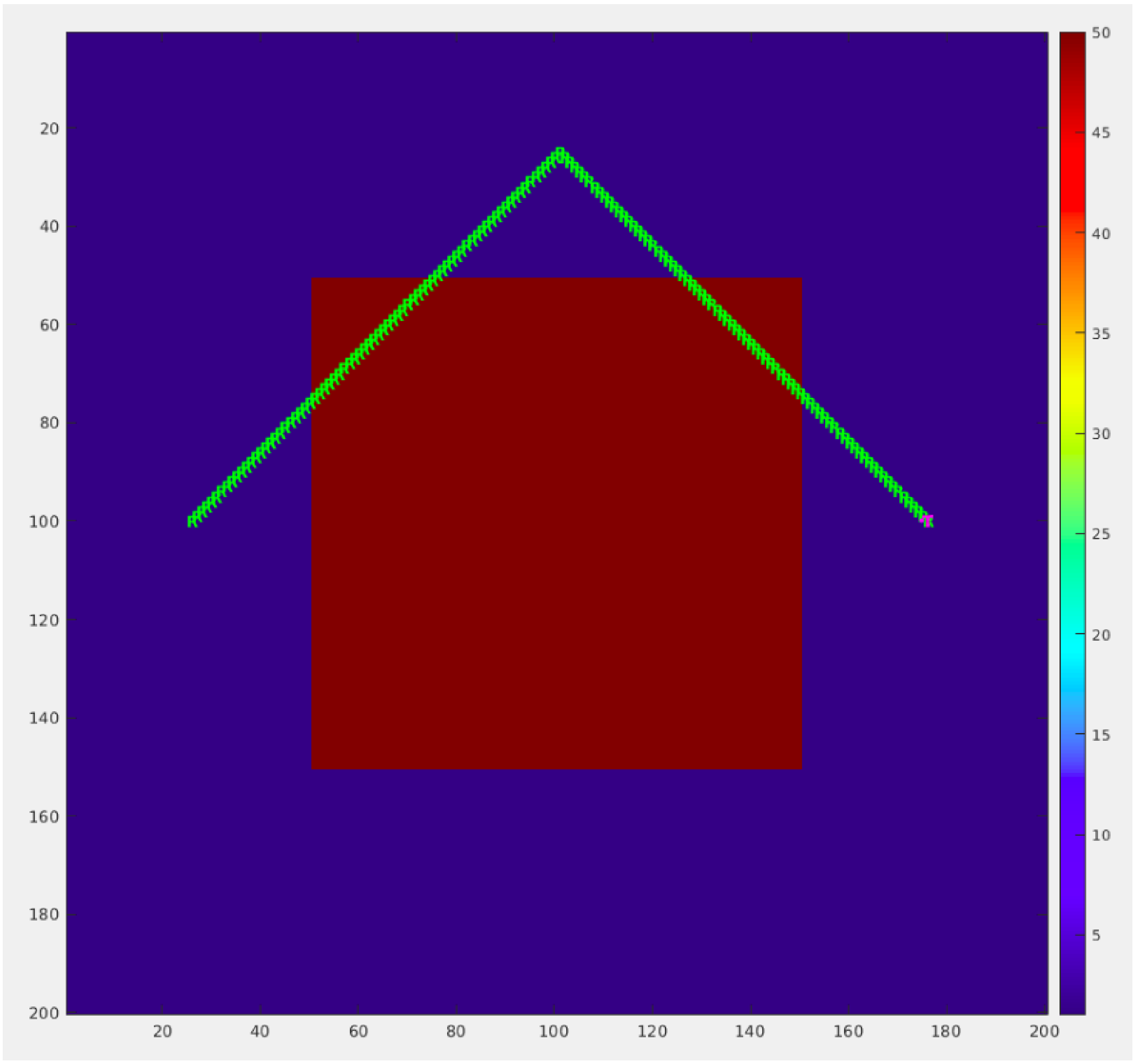
Target caught = 1
Time taken (s) = 583
Moves made = 581
Path cost = 583

4. MAP 4



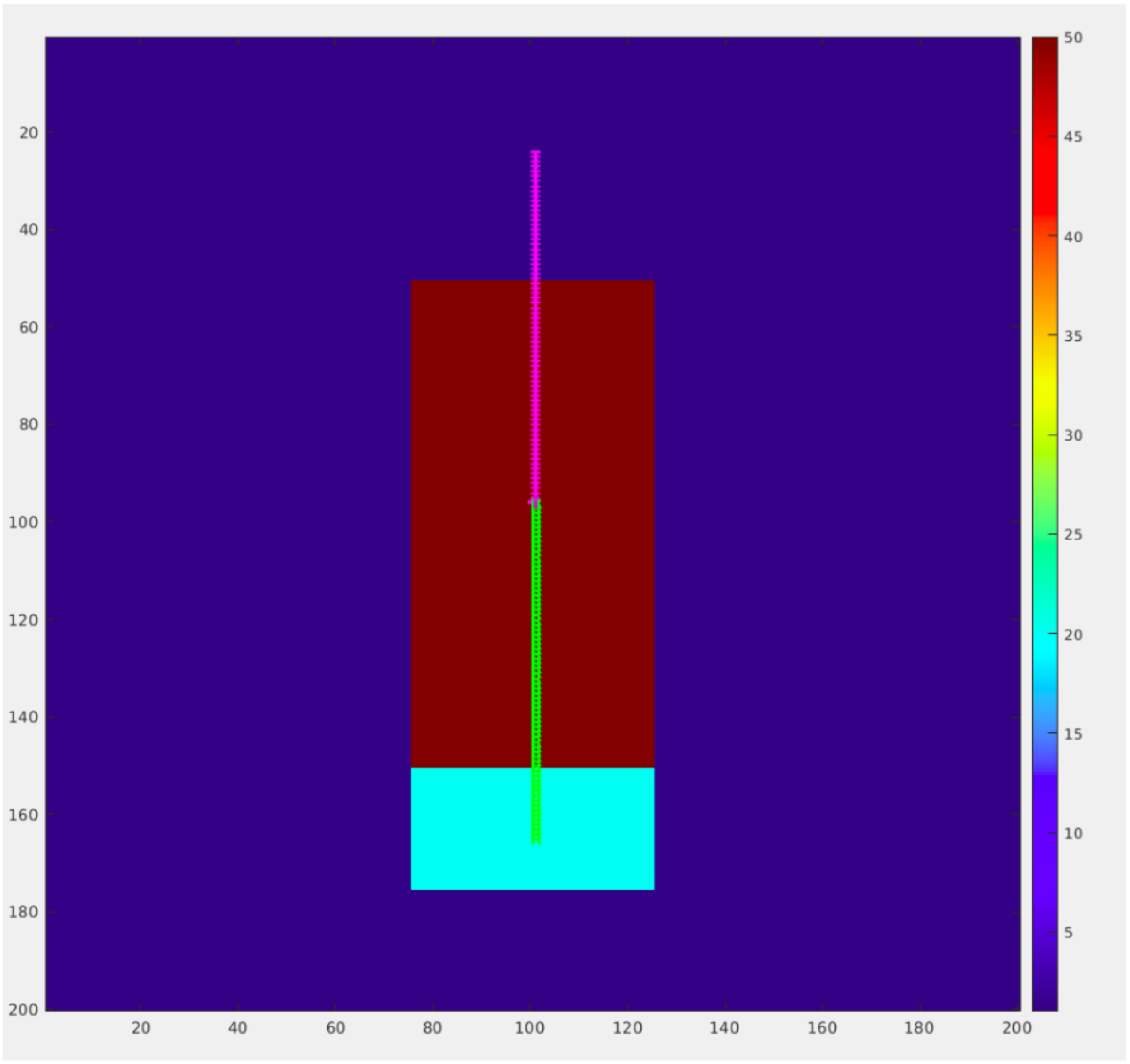
Target caught = 1
Time taken (s) = 565
Moves made = 563
Path cost = 195728

5. MAP 5



Target caught = 1
Time taken (s) = 150
Moves made = 150
Path cost = 2551

6. MAP 6



Target caught = 1
Time taken (s) = 72
Moves made = 69
Path cost = 3150