Aerial Robotics Kharagpur Documentation
Localization in known environment

Kumar Anant 22MA10030

***Abstract*— This documentation belongs to Task 3 (Localization in the known environment) of perception task round for the selection of 1st years in ARK**

I have completed Task 3 by completing the strategy() function in the code. My code is giving the exact position of the drone in almost 80% of 100 runs of code; for the rest 20%, I am getting relatively small errors in the position of the drone. However, I have specified in the output of the code that if the resultant position of the drone is exact or if there may be some small error. I will explain in this documentation how my code is able to recognize if the resultant location is exact or not, as the code lacks information about the drone's initial position.

I. INTRODUCTION

This task aims to find the drone's location of a drone in a known environment, which is a maze. It is allowed to move the drone vertically and horizontally by any desired value of pixels. By running $getSnapShot()$, the drone camera turns on and returns 51×51 images surrounding the drone. I have to localize it using snapshots and movements.

Another thing I want to discuss here is that I observed that $move\_vertical()$ is returning how much the drone moved vertically, Suppose the initial drone's location is (i, j)(we don't have this, right!) and the location at which the initial snapshot matches in the Map is (m, n)(we can find this) then, I prompt the drone to move 800 pixels vertically downward then it may be possible that it encounters a wall and moved only 5 pixels downwards this makes location drone to $(i, j+5)$ (again, we don't have this too). Now, this 5 pixels can be either found by just storing the returned value of move_vertical() function or by again prompting the drone to give a snapshot and again doing a template match in the whole map; suppose we got the location of the template match of snapshot after move downward as (m, n') (we can find this) then $(n' - n)$ must be equal to 5. My point is just to make you sure that instead of doing a template match every time the drone moves, I have used returned values of move_vertical() and move_horizontal().

II. OVERVIEW

To make the explanation easier, let's call drone location as:

$$(x_d, y_d)$$

Basically, what I am doing is I am prompting the drone to move vertically down by 800 pixels and then back to the original position by getting how much the drone actually moved downward. And repeating this in an upward, rightward and leftward direction, simultaneously storing how much it moved in every direction. This will help me to get the maximum movement drone can make in a particular direction. Suppose UP, $RIGHT$, $LEFT$, and $DOWN$ are the variables storing how much the maximum drone can move in upward, rightward, leftward, and downward, respectively. After doing this, The information I get is that there is Wall or, technically, the pixel value at $(x_d, y_d + UP + 1)$ is 0. Similarly, for Right, left and downward direction.

III. FINAL IMPLEMENTATION

Now, what I have done is that I have started a loop start from the co-ordinate of the Template match of the snapshot when the drone was in its original position (say it is (m, n)) up to $(m + 51, n + 51)$ one by one checking each pixel that if pixel value at $(i, j + UP + 1)$, $(i, j + DOWN + 1)$, $(i + RIGHT + 1, j)$, $(i + LEFT + 1, j)$ is 0 or NOT where (i, j) is the location of active pixel while loop is running.

if NOT, then check the next adjacent pixel. if YES, then it may be possible that it is the drone's position, I am mentioning "it may be possible" because there may be some other position in matrix (m, n) to $(m + 51, n + 51)$ where pixel value at $(i, j + UP + 1)$, $(i, j + DOWN + 1)$, $(i + RIGHT + 1, j)$, $(i + LEFT + 1, j)$ is 0. In fact, after running the above algorithm there are about 600-700 pixels in $(51 \times 51) = 2601\, pixels$ where pixel value at $(i, j + UP + 1)$, $(i, j + DOWN + 1)$, $(i + RIGHT + 1, j)$ and $(i + LEFT + 1, j)$ is 0. Now, only one pixel belongs to the drone's location in those 600-700 pixels. To Refine that, I have checked that there should be no wall or black pixel in between (i, j) to $(i, j + UP + 1)$, $(i, j + DOWN + 1)$, $(i + RIGHT + 1, j)$ and $(i + LEFT + 1, j)$ if (i, j) is location of drone. This makes one in 600-700 possible locations shrink to one in 2-7 possible locations and sometimes even get only one value of (i, j), which is the drone's location.

Now, At this point, for 100 samples of the maze, there are about 55 in which I got only one value of (i, j).

To further refine this, I have prompted Drone to do some other types of movements, like moving clockwise once and storing how much it moved right, left, up, and down and similarly checking if there is no black pixel in the path created by the drone for each value of (i, j).

Now, At this point, after running the code 100 times and storing in how many cases, I got only one value of (i, j); it is 80 cases where I got single (i, j) which will be the exact location of the drone with no error.

Rest 20 cases give only 2-3 possible values of $(i,\ j)$, which I observed are **very close** to each other. So, I simply took the average value of those 2-3 values of (i, j) and printed "There may be some small error in drone's position" as output.

Please look at the code to get more of what I have done!

IMPROVING ACCURACY

So, as I mentioned above, my code gives the drone's position with no error in 80% of 100 random samples of the maze and a relatively small error in the drone's position in the other 20% cases. It can be noted that if I had prompted the drone to do more movements, the 80% can be taken close to 100%.