The function for the lidar simulation is showing the range the robot can see:

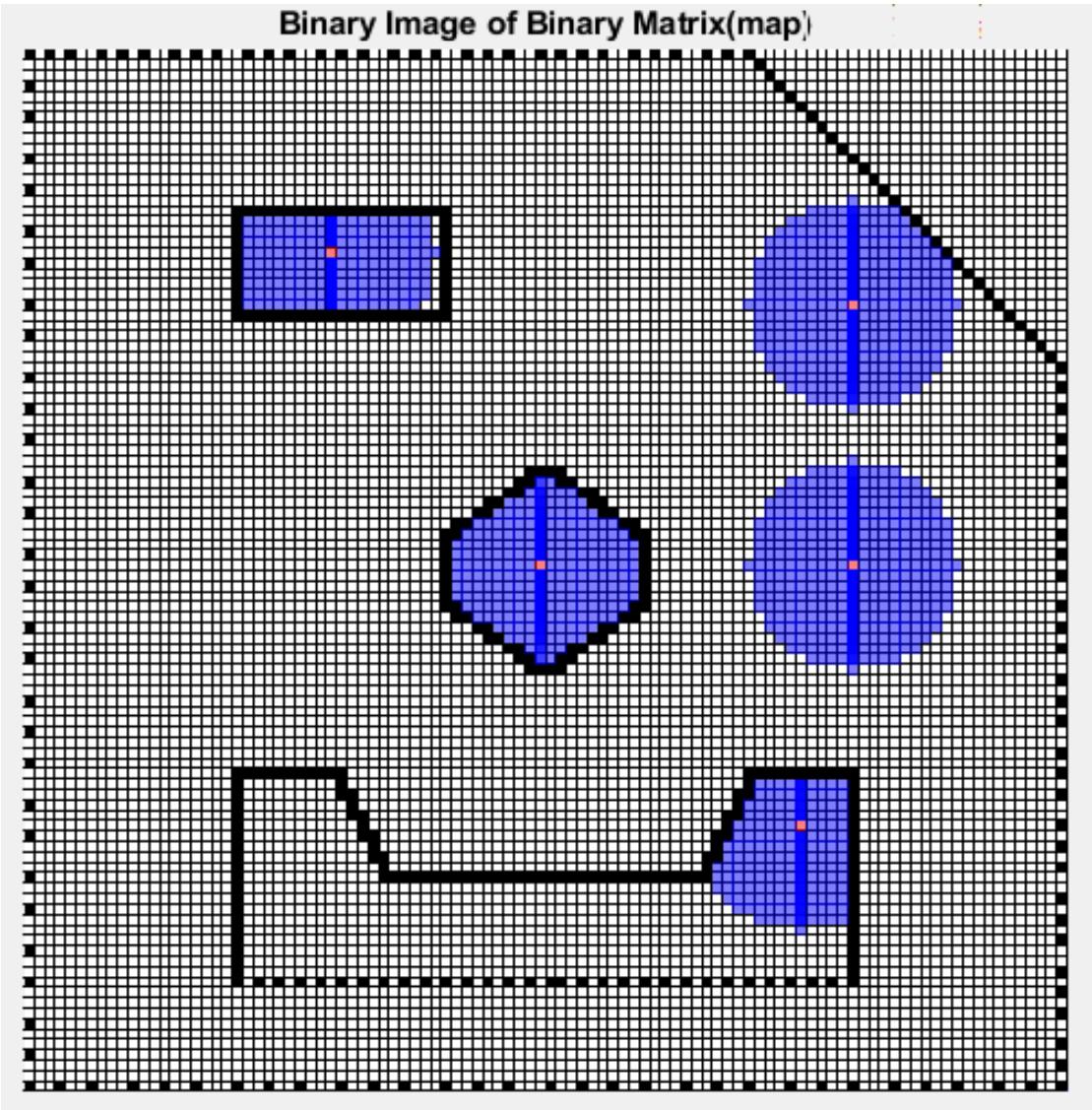


Figure.1: This figure shows the lidar simulation of the robot for different positions([30,20], [50,50], [75,75], [80,25], [80,50]). This code also takes into account if there are other robots and make them see each other and update the map data. This enhancement is not needed for the use of only one robot but if you want to use more than 1 robot you just need to add another robot position.

Assumptions:

I defined the robot movement function as follows:

- It can choose to go up, down, left, or right with the probability of 0.25 for each.
- It can move 1,2,3,4 or 5 cells at each cycle.
- The robot is able to move in any direction it wants, but when it moves in a direction, with a 5% chance it moves 1 or 5 cells, with a 15% chance it moves 2 or 4 cells and with a 60% chance it moves 3 cells.

These conditions yields following movement capability:

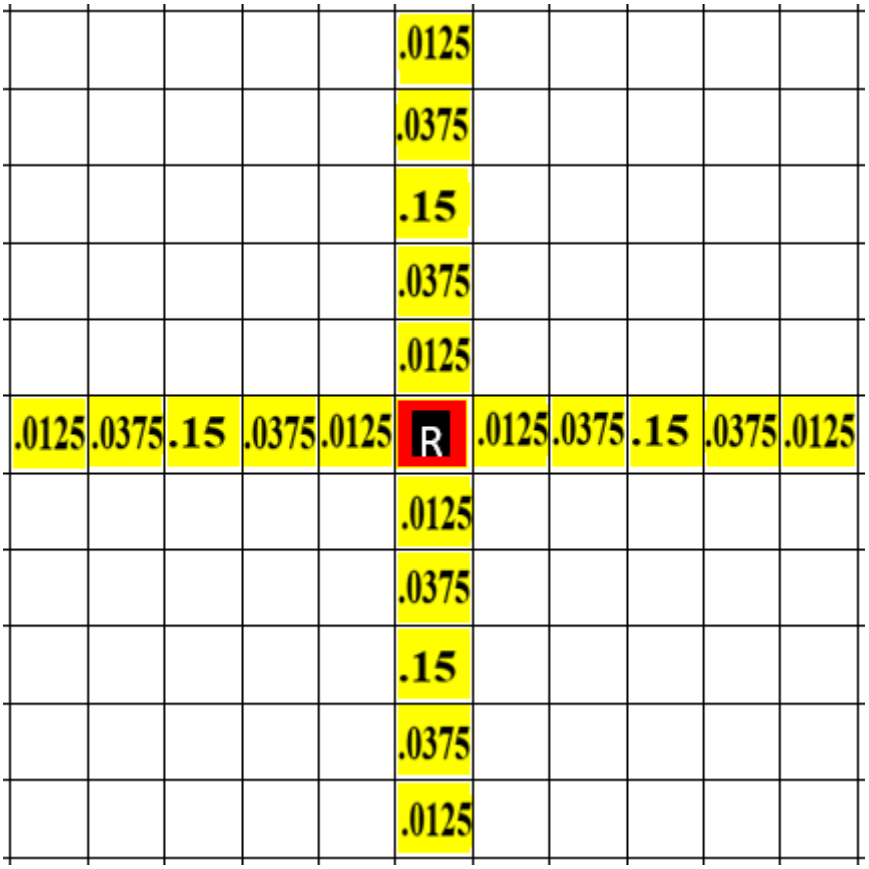


Figure.2: Probabilities of the cells that the robot may be inside in the next position

The lidar sees an obstacle at its proper location with a 40% probability whereas it sees an obstacle in an adjacent cell has a probability of 7.5%.

This condition can be interpreted as follows:

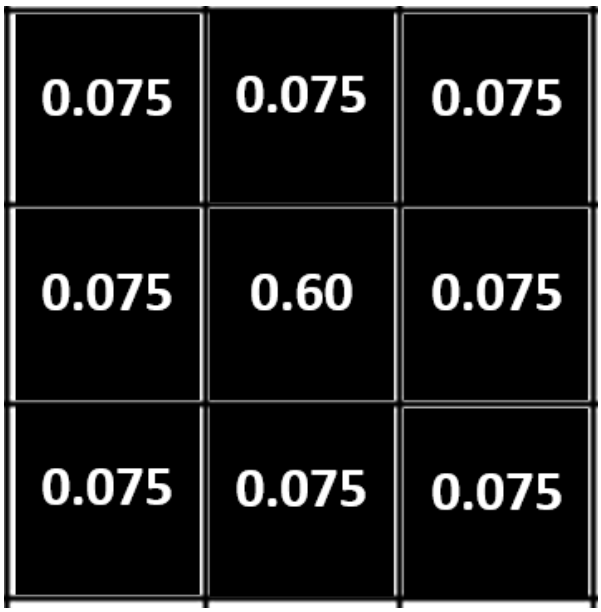


Figure.3: The probabilities that the robot may see where an obstacle cell is. The middle cell is the actual cell.

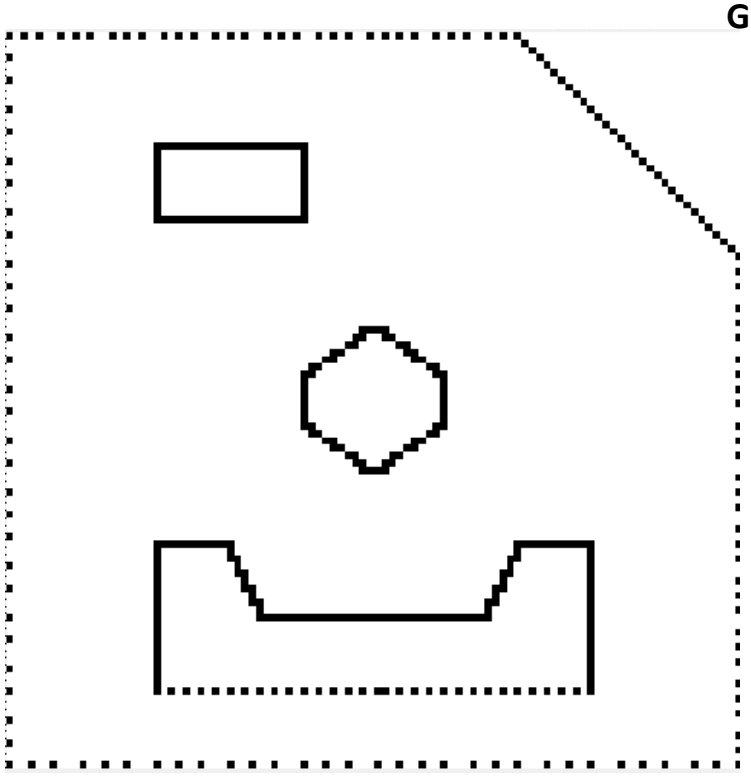


Figure.4: This figure shows how the map interpretation would be for the robot if it was able to see an obstacle 100% in where the obstacle actually is.

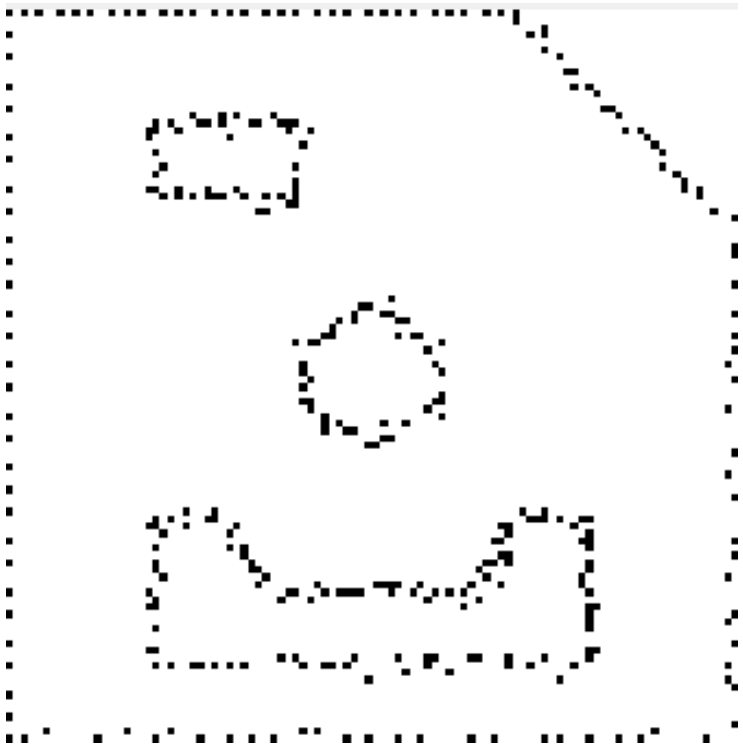


Figure.5: This figure shows how the map interpretation is for the probability distribution given in Figure.3. This would change if you run the code again because of the probability distribution.

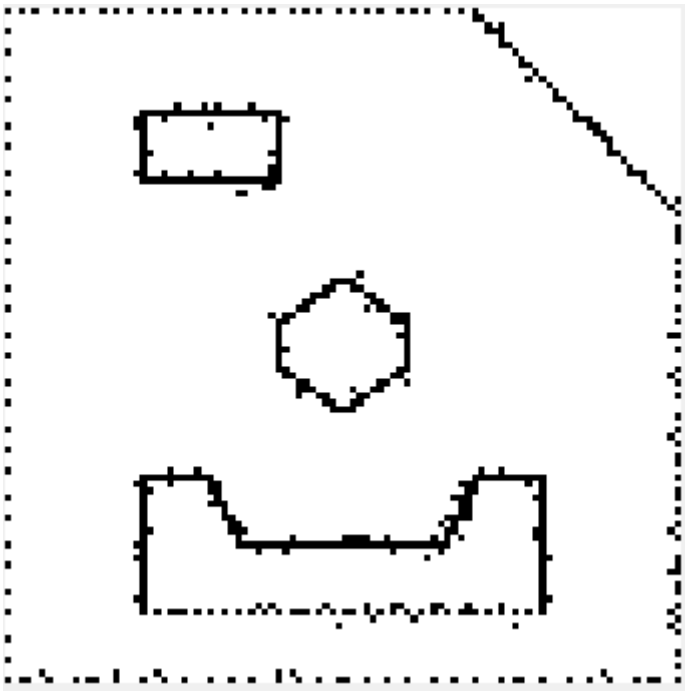


Figure.6: This figure shows the combination of matrices shown in Figure.4 and Figure.5. I created this map in order to apply obstacle avoidance. Even if the lidar couldn't see the obstacle where it is if the robot goes in that direction and meets with an obstacle it chooses to change its direction. By doing so, the robot is not allowed to go beyond the actual wall cells as well as the measured wall cells because the robot believes there is a wall even if there isn't.

Cycle Size = 100 (Initial position = [50, 50]):

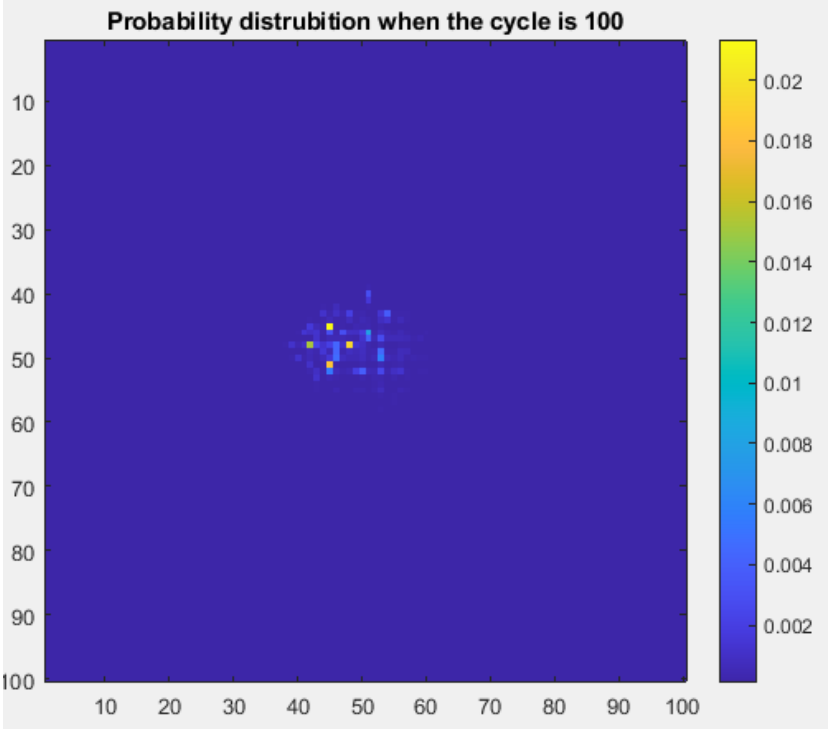


Figure.7: This shows the probability distribution of the robot position belief. When you run the code you will see the result of each cycle, this figure is the final.

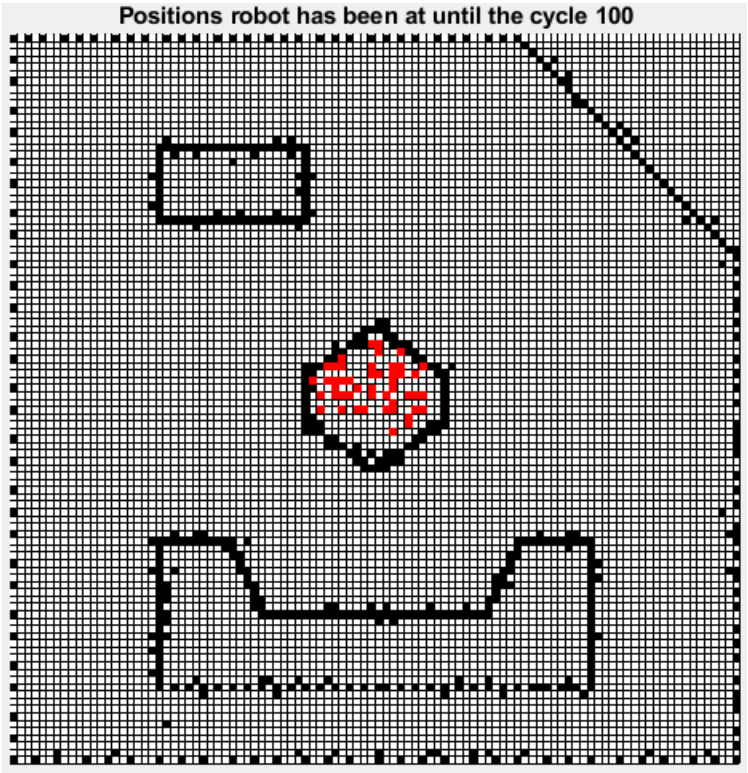


Figure.8: Positions that the robot has been at until cycle 100. When you run the code you will see the result of each cycle, this figure is the final.

Cycle Size = 100 (Initial position = [80, 50]):

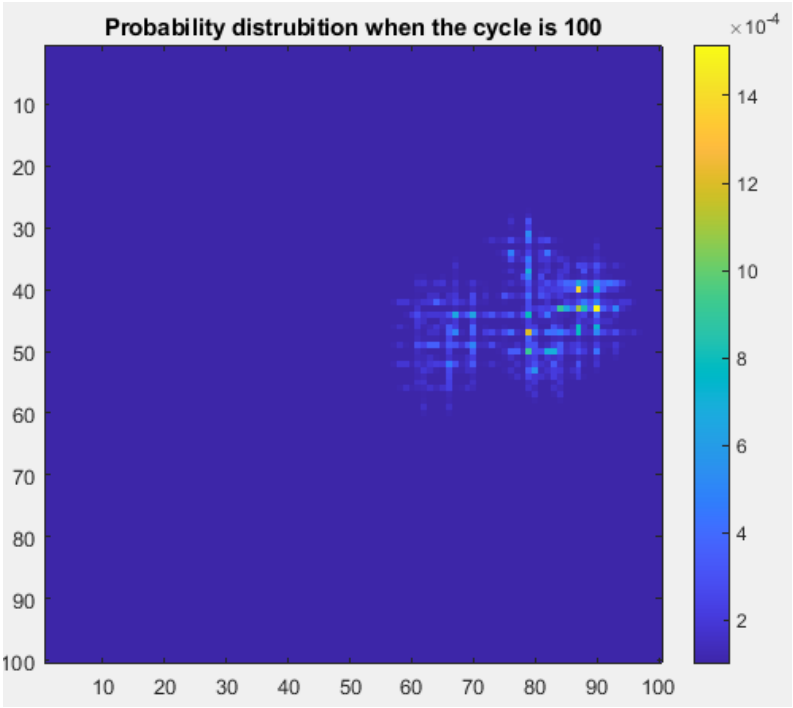


Figure.9: This shows the probability distribution of the robot position belief. When you run the code you will see the result of each cycle, this figure is the final. The probabilities of this map are more spread to the map and much less certain than the previous map. The reason is, fort he previous map, there were walls surrounding the robot so it visualized the very similar environments for each cycle. Therefore it learnt faster.

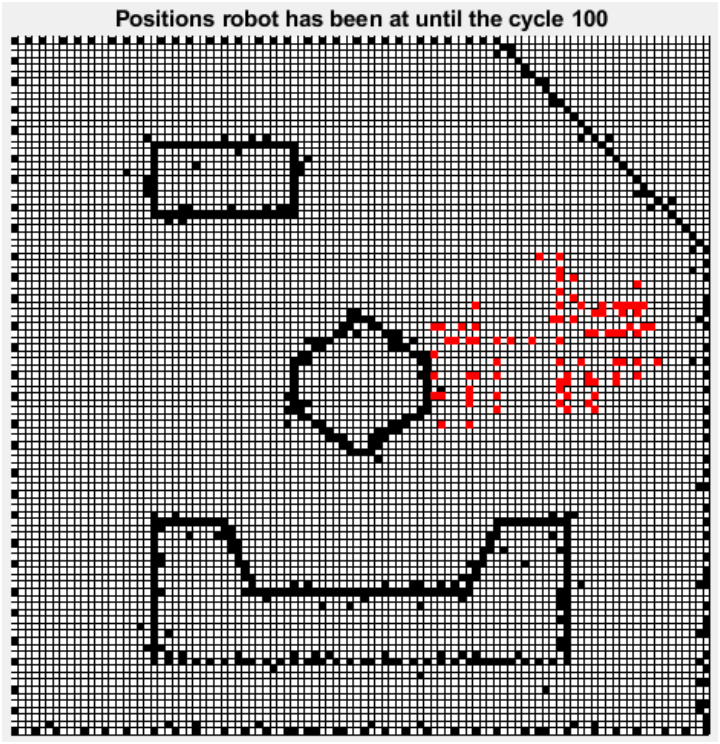
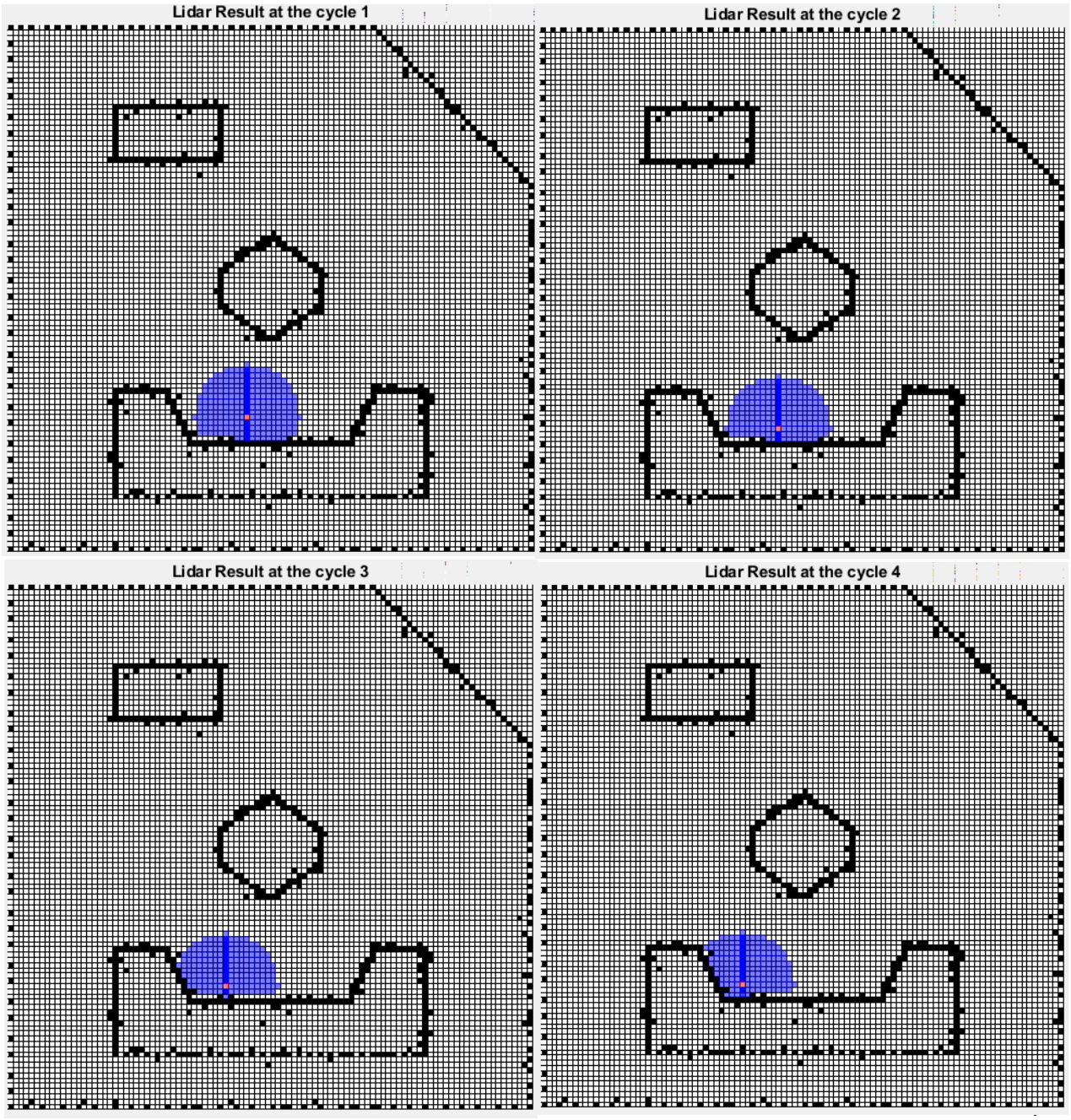


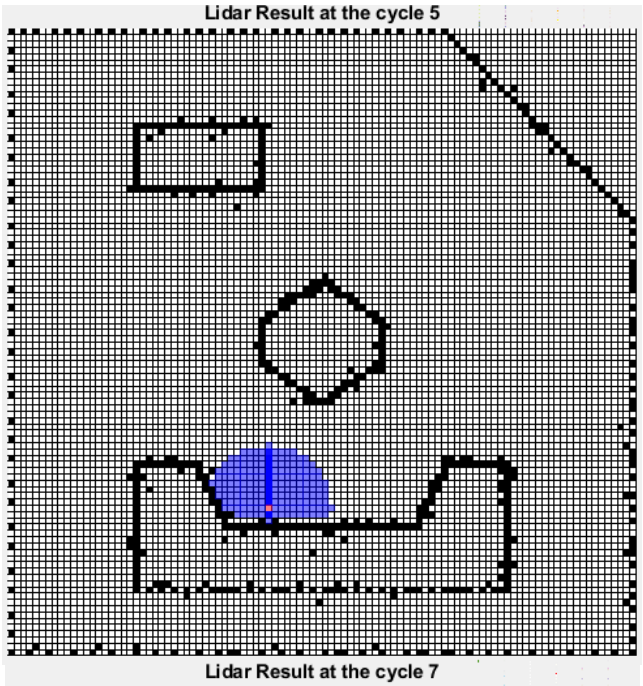
Figure.10: Positions that the robot has been at until cycle 100. When you run the code you will see the result of each cycle, this figure is the final.

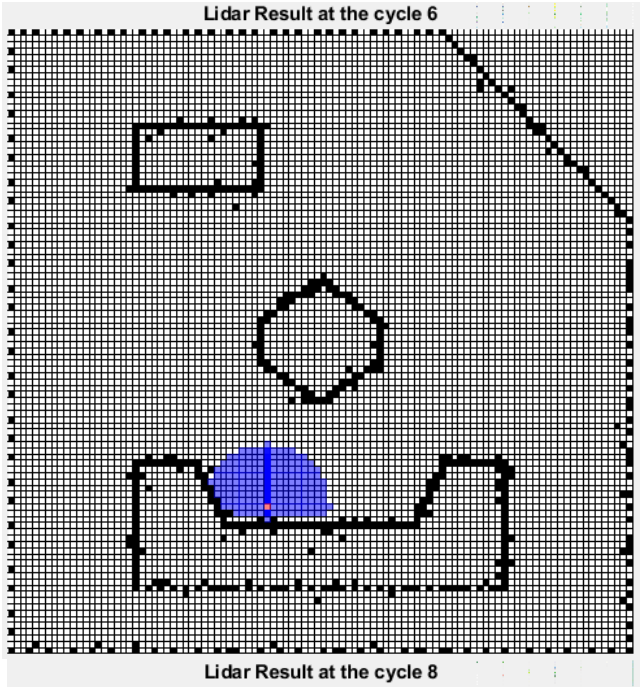
First 20 Lidar Data for the Second Example Initial position = [50, 75]

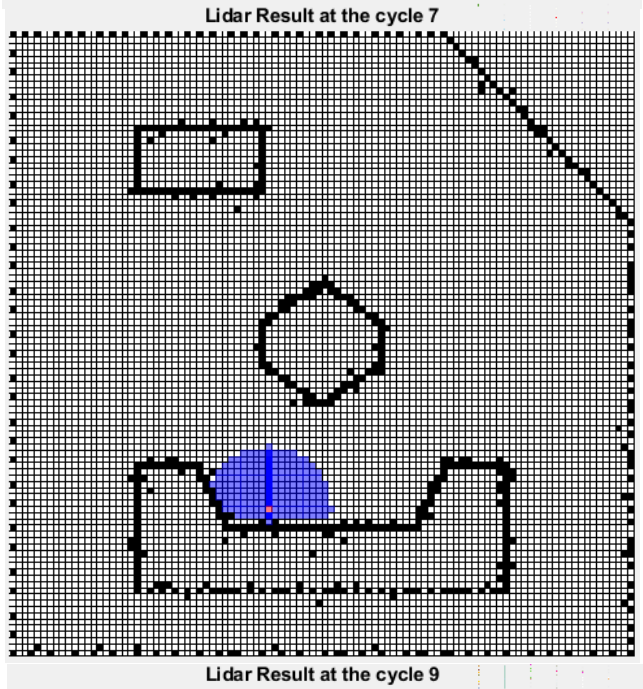
```
346 %     fh = figure();
347 %     fh.WindowState = 'maximized';
348 %     subplot(11,10,1:100)
349 %     imshow(matrixForLidar) % Showing combined matrix
350 %     hold on
351 %
352 %     for i = 1:100
353 %         line([0.5,100.5], [0 + i+0.5,0 + i+0.5],"Color","k","Linewidth",0.5)
354 %         hold on
355 %         line([0 + i+0.5,0 + i+0.5],[0.5,100.5],"Color","k","Linewidth",0.5)
356 %         hold on
357 %     end
358 %     cycleNumber = 3;  %You should change the cycleNumber to see lidar data of different cycles
359 %     title("Lidar Result at the cycle "+string(cycleNumber))
360 %     lidardata([positionarray(cycleNumber,1),positionarray(cycleNumber,2)],matrixForLidar);
361 %     hold on
```

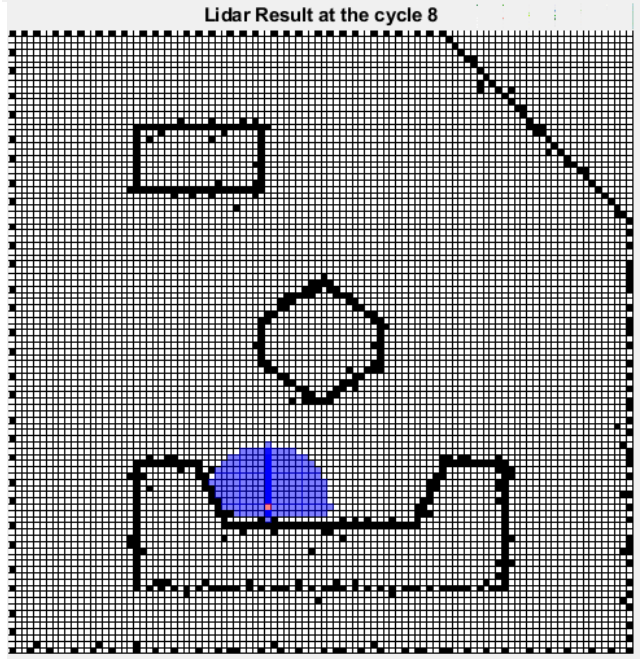
Figure.11: This section of the code allows you to see lidar data for different cycles. After running the code please uncomment and paste it to the command window, it will automatically execute itself, otherwise press enter.

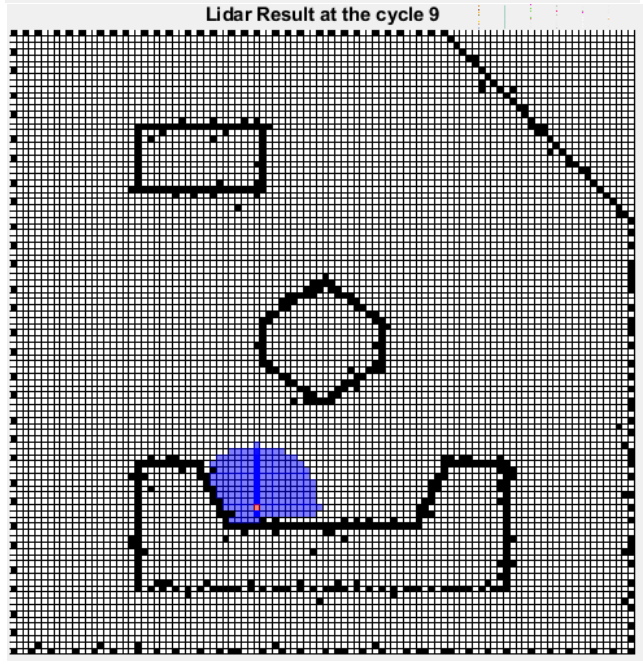


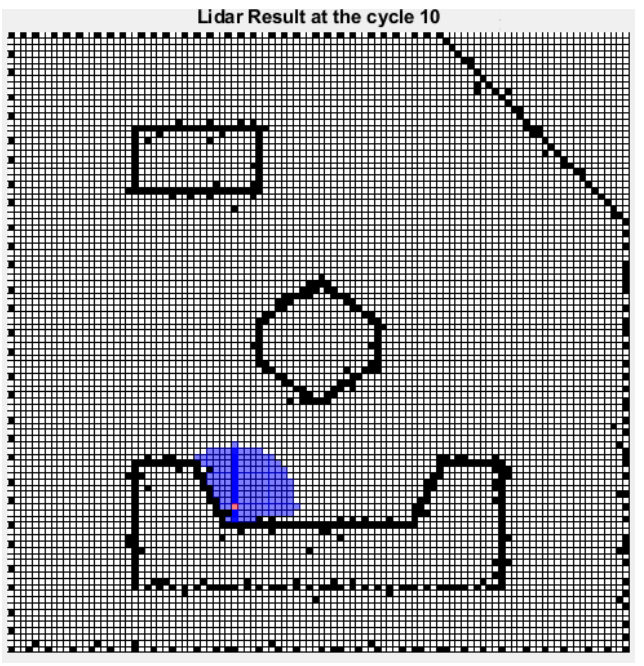

Lidar Result at the cycle 5

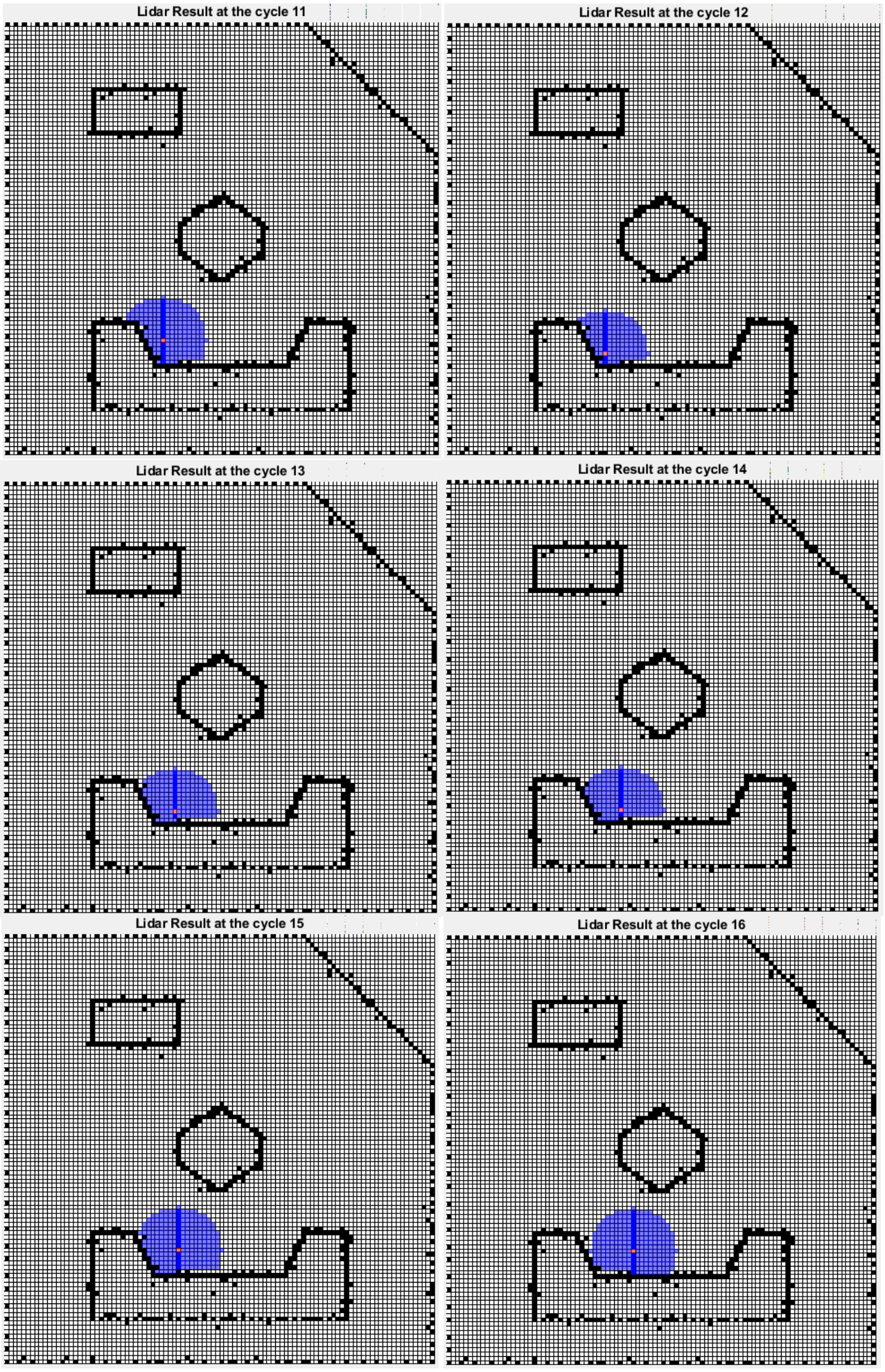

Lidar Result at the cycle 6


Lidar Result at the cycle 7


Lidar Result at the cycle 8


Lidar Result at the cycle 9


Lidar Result at the cycle 10


Lidar Result at the cycle 11
Lidar Result at the cycle 12
Lidar Result at the cycle 13
Lidar Result at the cycle 14
Lidar Result at the cycle 15
Lidar Result at the cycle 16

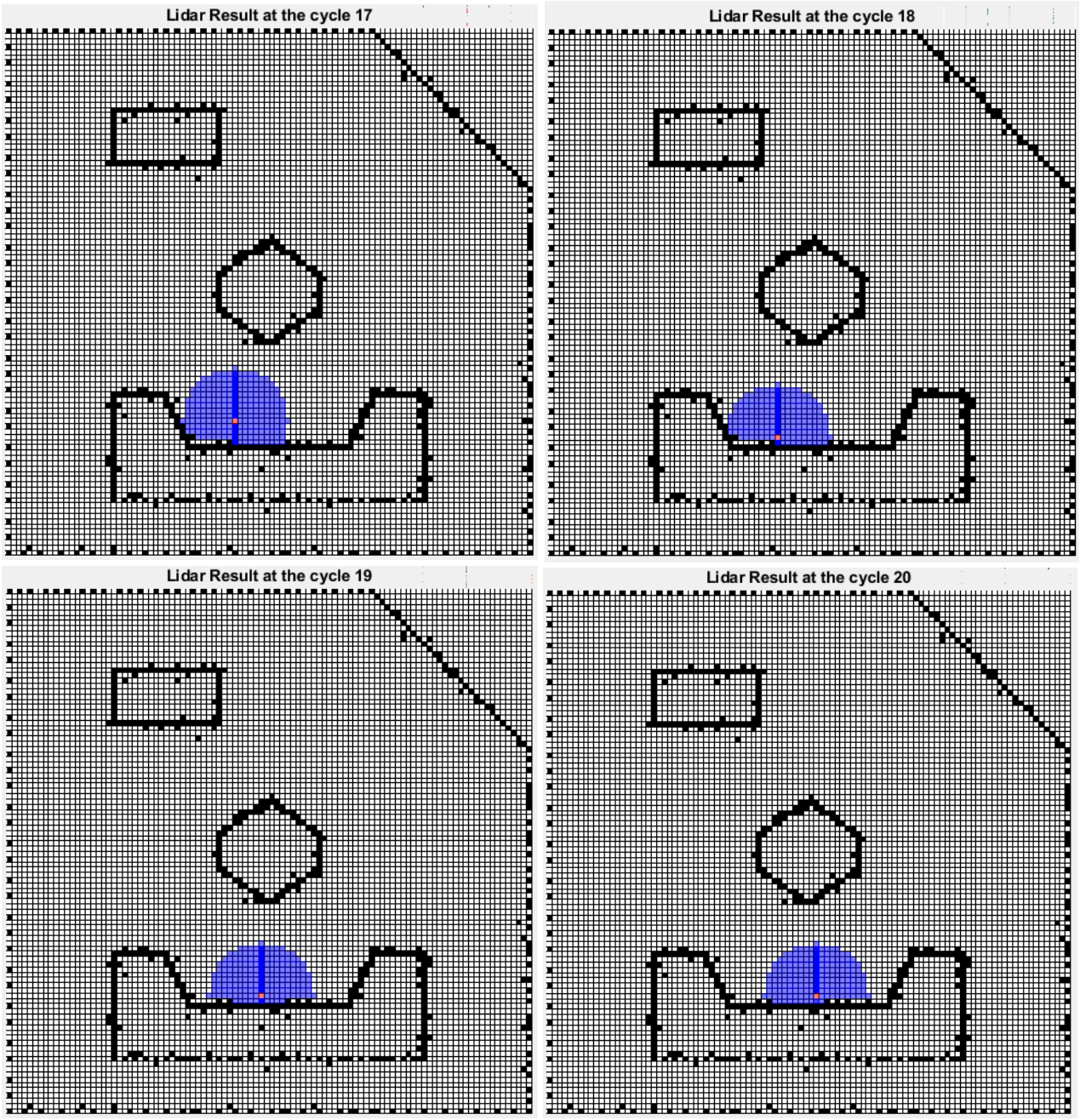


Figure.[12:31]: Lidar data for cycles 1 to 20.

If the robot's belief is 100% it stops moving and gives a message:

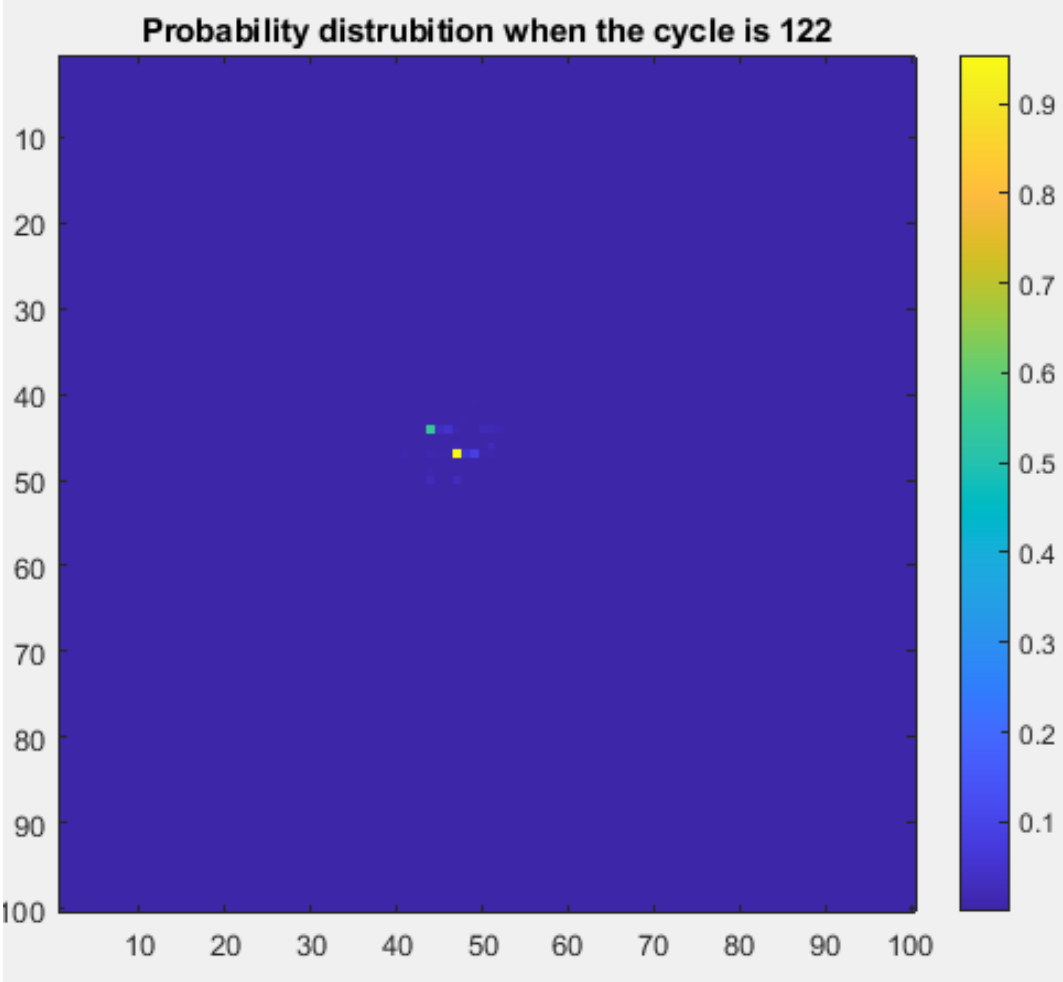


Figure.32: When the robot's belief get very close to one, it stops moving. Basically, it knows where it is. The initial position is [50,50] at middle, between the walls. I entered 350 cycle but it learnt where it is and stopped moving further.

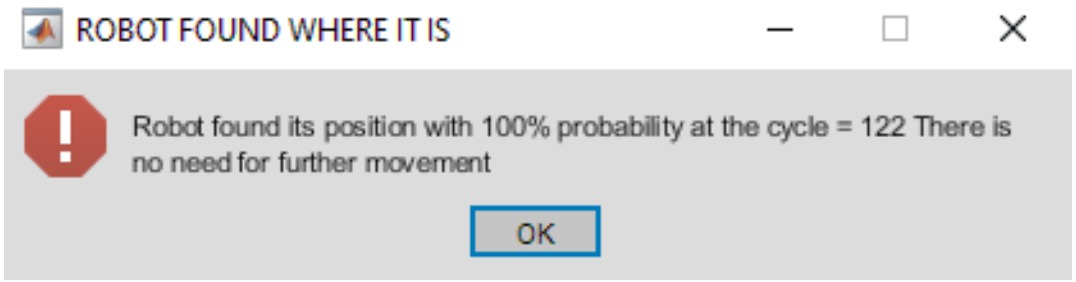


Figure.33: This window is opened when the robot learns where it is.

If the robot leaves the map, a warning is given:

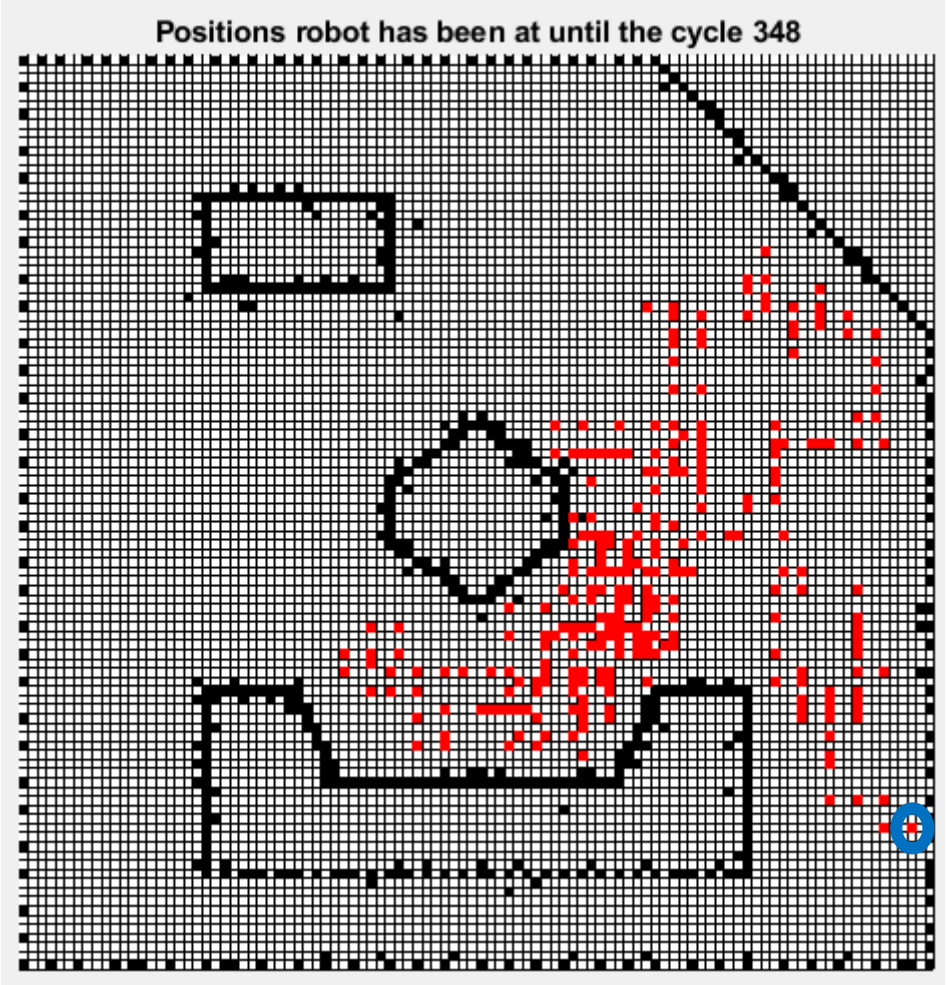


Figure.34: The initial position is given as [80,50]. The robot tried to escape from the map at cycle 348 so further execution of the code is forbidden.

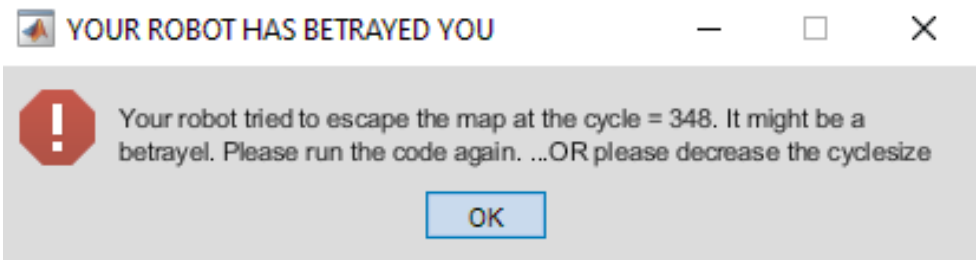


Figure.35: This window is opened when the robot tries to go out of the borders.