CS 4758/6758: Robot Learning: Homework 3

Due: Mar 9 (Tuesday), 5pm

1 Localization using Computer Vision (35pts)

In this scenario, we are trying to locate Rovio robot using the images captured from a (different) camera.

1.1 Image denoising and filters

The images we receive from the camera are noisy (`receivedimage.jpg`, see Figure 1-left). We need to clean the image up. Create a filter to make the noise go away. Do a comparison with your result and the original image (`original.jpg`) shown in Figure 1-right. Hand in your code, give an explanation for the filter you applied, and the RMS error.

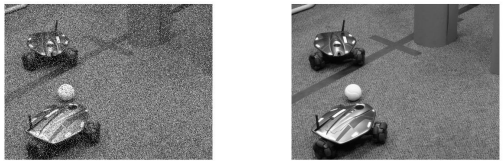


Figure 1: (a) Noisy image captured, (b) original image.

1.2 Localization

Rovio is navigating on a white platform (with other noise, etc.). How can we approximately estimate Rovio's location? In this question, you need to right a computer vision algorithm that can output the (x, y) location of Rovio in the image. Test it on three images: `rovio1.jpg`, `rovio2.jpg`, and `rovio3.jpg`.

Hint: You can implement an edge detection algorithm. After you find all the edges of the image, compute the histogram of the edges in the x and y directions. Look at the peaks and the mean of the histograms to get the x-y estimate for each of the three images.

Please hand in: (a) the image after the edge detection, (b) the x- and y-histograms, (c) the computed x and y locations. Explain your method (edge detection algorithm, pre-filters applied and method to get x-y from the histograms). Attach your code at the end of your homework.

Hint: You might need to apply a filter before computing the edges.

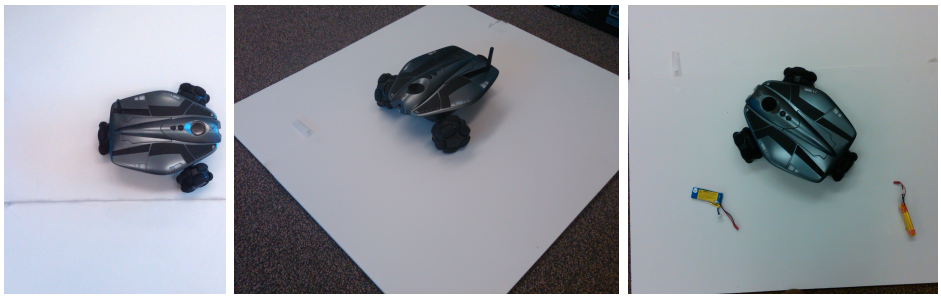
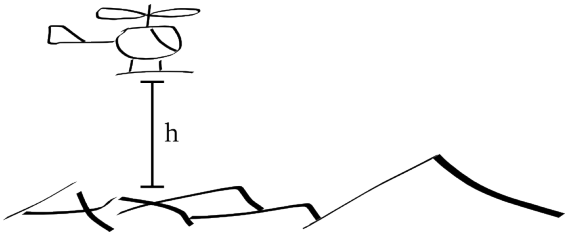


Figure 2: Estimate Rovio's position for these three images.



2 PID Control (35pts)

In this exercise you will design a PID controller for controlling the height of the helicopter (in simulation). We will assume that all the other motions are perfectly controlled, and the helicopter is restricted to vertical motion only.

The goal is to make the helicopter follow a set trajectory (i.e., different heights at different instants in time) by controlling its rotor speed. There are three MATLAB source files associated with this problem: `heli.m`, `control.m` and `testResults.m`. `Heli.m` simulates the helicopter dynamics and sensors, which you should not edit. `Control.m` is the controller for the helicopter; edit this to build a PID controller. `testResults.m` tests your controller on three trajectories and produces two plots: one showing the helicopter's actual vs. desired path, and one its rotor speed.

You will have to tune its coefficients to get good performance. For your finished design, please submit:

1. The plots and RMSE (root mean square error) from `testResults.m` for the three trajectories.
2. The weights assigned to the P, I and D terms.
3. Your source code (`control.m`)

3 Grid-based localization (20pts, plus 20pts bonus)

A robot (Figure 3) lives in a $2D$ square world W (Figure 4), which has been discretized via a fixed cell decomposition. The robot's world W can be represented as a 2-dimensional array of integers, with $W(i, j)$ indicating the cell in row i and column j. The robot can occupy a single open cell at any given time. We used a 3D gird (x, y, θ) to represent the robot's status. where $x \in \{1, 2, 3, 4, 5\}$, $y \in \{1, 2, 3, 4, 5\}$ and $\theta \in \{0^o, 90^o, 180^o, 270^o\}$. The shaded squares are obstacles,

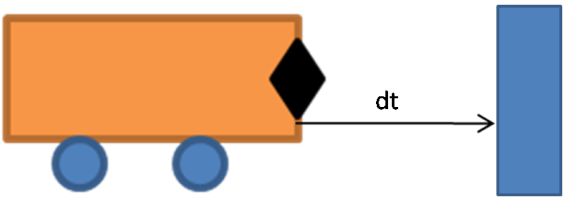


Figure 3: A simple robot with a single sensor

where the robot cannot move. And $\theta = 0^o$ represents the direction to the right. The outside blue boundary is the wall of world. The rest of the non-blue blocks are open spaces.

The Robot has only one sensor shown in Figure 3 and its sensor can measure distance to the wall or obstacles in front of it. At each time t, the sensor will report back a $d_t \in \{0, 1, 2, 3, 4\}$.

3.1 Perfect Sensors (20 pts)

We will start with a simple case where the distance sensor gives perfect information, i.e., it has no noise. For example, if $d_t = 0$, then that means the robot has to be either facing the wall or an

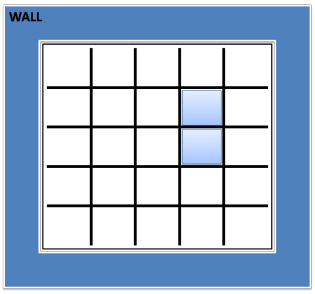


Figure 4: A map of the current World

obstacle. Also note that if the robot tries to move forward or backward, and there is a wall, the robot will not move and the sensor reading will remain the same. (i.e., you can use consecutive sensor readings to determine if it was facing away or towards an obstable)

Assume an initial probability that the robot is equally likely to appear on any open space blocks. The robot's sensor reads $d_0 = 0$ at time 0. Then the robot made the following two actions (actions are also perfect).

1. Action A: move backward one block without changing θ. The sensor returns $d_1 = 1$
2. Action B: move backward one block and turn left by 90^o, the sensor returns $d_2 = 0$

At each time step ($t = 0, 1, 2$), write out each probability distribution $P(x, y)$, which would be a 5×5 matrices.[1]

With the perfect sensors and perfect actions, does the robot know perfectly its location at $t = 1$? Does it know at $t = 2$, or should it move around and sense more? If it should move and sense more, write more actions that will let the robot know perfectly its position. (You get more points for fewer actions taken by the robot.) (Note that in one action, the robot can move only one block—forward or backward—and rotate by any angle.)

3.2 Imperfect Sensors (Bonus only: 20pts)

In reality, we do not have perfect sensors. (Neither do we have perfect action; but to make our life easy, we will still assume that the robot actions are perfect.)

In this part, when our sensors report d_t, we will not be 100% sure about the real distance d_{real}. Based on some previous experiments, the probability $P(d_t|d_{t,real})$ is given in Table 1:

Table 1: Table showing $P(d_t = x|d_{real} = y)$.

y =	0	1	2	3	4
x=0	0.9	0.1	0.05	0	0
x=1	0.1	0.8	0.1	0.05	0
x=2	0	0.05	0.8	0.05	0.1
x=3	0	0.05	0.05	0.8	0.1
x=4	0	0	0	0.1	0.8

Repeat the previous question, with this noisy sensor. I.e., report the probability after Action A and Action B.

How would you figure out the next best action to take such that the robot becomes more confident of its location. (Because the sensors are noisy, it would take a really long time for it to become 100% confident.)

[1] Hint: You would have to marginalize, i.e., sum over θ. We do not expect you to write code for this question.

4 Your Robot (10 pts)

For your robot and project, describe how would you do each of the following:

1. Robot Building
2. Perception/Sensing
3. Localization
4. Control
5. Planning

It will be evaluated on the basis of how precise and informative your answer are. (Not all of them will apply to your robot project.)

(Write briefly—not more than a few lines each.)