ENGNX627 Lab Tutorial: Seek a Dot

In this lab tutorial, you will work along with the demonstrator to write a matlab script that makes your robot drive toward a dot on the floor. Many of the tools and techniques in this tutorial are highly relevant to the line following task. The floor of the room will be as shown in Figure 1

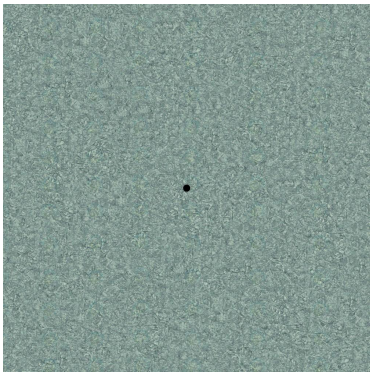

Figure 1: The floor image for this exercise.

Begin by starting the simulator and placing the robot at $(1.5, 2.0)$. Next, capture an image with the camera using `img = pb.getImage()`. The output image should look like Figure 2.

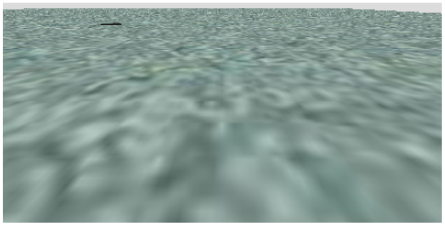

Figure 2: An image captured from the robot camera.

Now we will do some image processing. To drive toward the dot, we will need to identify it within the image. This is easily done with image thresholding. Convert the image to grayscale using `rgb2gray`. Then you can use `imbinarize` to threshold the pixel values of the image. The result of this process is shown

Figure 3: A binarized image from the camera.

in Figure 3. If you are not sure how to use these functions, you can type `help rgb2gray` or search for the documentation online or in the MATLAB help browser. Knowing where to find documentation and understanding it is an important skill, so make sure to practice it.

From here, we want to compute the centre of the pixels. This is simply the average of all pixel coordinates with a value of 'true'. That is,

$$p_{\text{avg}} := \frac{1}{n} \sum_{p \in W} p,$$

where W is the set of all white (true) pixels, and n is the number of pixels in W. This can be implemented in a number of ways in MATLAB, so have a go and see what you can think of!

Now that we have the average pixel coordinate of the dot, it is nice to normalise the coordinates. One easy way is to try and restrict the x and y values to the range $[-1, 1]$. Using `size(img)` will tell you the image has 200 rows and 400 columns. Knowing this, you can normalise the pixel coordinate by using

$$p_{\text{norm}} := \begin{pmatrix} 2/400 & 0 \\ 0 & 2/200 \end{pmatrix} \left(p_{\text{avg}} - \begin{pmatrix} 400/2 \\ 200/2 \end{pmatrix} \right).$$

Now x_{norm} and y_{norm} must lie between -1 and 1.

Finally, it is time to design a control strategy. This means choosing a linear velocity u and angular velocity q to guide the robot to the dot. If x_{norm} is negative, the dot is to the left, so the robot must spin to the left, i.e. q must be positive, and vice versa. A proportional control is a simple but effective strategy, that is, set $q = -k_q x_{\text{norm}}$, for some constant gain $k_q > 0$. As for the linear velocity, note that, as long as the dot is visible, it must lie somewhere in front of the robot. One option is to set $u = k_u$ for some constant $k_u > 0$. Another is to slow down when the dot is far to the left or right of the image (how might you do this?).

Set up your simulation loop and test your code. Did you manage to drive onto the dot? What if you can't see the dot from where you start? How could you find it? Does your code still work when the dot is not visible? Does your robot stop right on top of the dot, or not quite? How could you improve this?