Lab Procedure for Python
Line Following

Setup

1. It is recommended that you review Lab 3 – Application Guide before starting this lab.
2. Launch Quanser Interactive Labs, scroll to the "QBot Platform" menu item and then select the "Warehouse" world.
3. Run observer.py first to initiate receiving data feeds from the virtual QBot.
4. On a separate terminal, run line_following.py. When the script is run successfully, the QBot and the Environment should be spawn in QLab and User LEDs on the virtual QBot will turn blue. Verify that the QBot is spawn as shown in Figure 1, then press "U" key and stop the script.

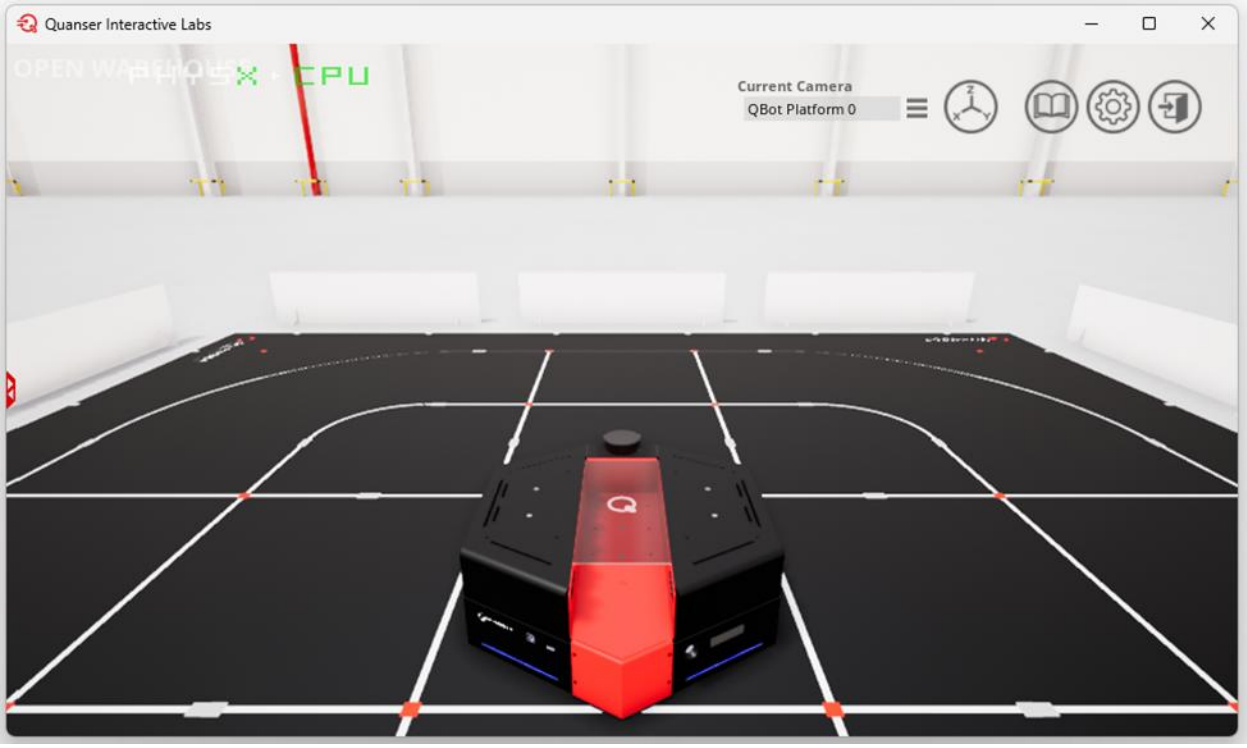

Figure 1. Successful set up of the Quanser Interactive Lab Workspace

Preparing the Image

1. In this lab you will be focusing on processing the information from the downward facing camera to allow the QBot to drive by itself. This will be done in Section D of the code in line_following.py.

2. Some code has been laid out for you, Section D.1 will first undistort the wide angle on the downward facing camera using camera properties and then resizing it for faster processing. Right click over **df_camera_undistort()** and click go to definition to see what the undistort function does. This will open the qbot_platform_functions.py, under the **QBPVision()** class, in which you will be writing the rest of the functions for this lab.

3. To begin processing downward camera feed, you will select the area of interest in the image by completing and using **subselect_and_threshold()** under **QBPVision()**.

    a. First create a sub-selection of the input image, stored in the variable *subImage.* Using the function arguments *rowStart* and *rowEnd,* the processed image should contain pixels from *rowStart to rowEnd.*

    b. Then create a binary or bitonal image stored in the variable *binary* where the color/area of interest (the line to follow) is white, and the rest of the image appears as black. Use the function arguments *maxThreshold* and *minThreshold*, and the OpenCV function called **threshold()** to create the binary image.

    c. Save your changes to the function and go to Section D of line_following.py. Use **subselect_and_threshold()** to complete Section D1, using 50 and 100 for *rowStart* and *rowEnd*, and appropriate thresholds. The output *binary* should be a 50x320 binary image.

4. Run observer.py first, then run line_following.py. Press the Space Bar to arm the robot and keep it pressed. You should be able to drive it using the keyboard as you did in the previous lab.

5. On your computer, observe the output of your binary image, and verify that you properly highlighted the area of interest. The line should look white while the rest of the image black. If it is not correct, change the thresholds input to the function in Section D2 and run the code again. Do this as many times as needed.

Find Objects of Interest and Line Follow

1. Now that the image is captured and the area of interest is highlighted, we will now localize the blob of interest to enable line following. We will use the function **image_find_objects()** in the **QBPVision()** class in qbot_platform_functions.py.

2. You will have to complete this function to extract the blobs of interest using the OpenCV function **connectedComponentsWithStats()**. Separate the values from its output to their corresponding variables (labels, ids, values, centroids). Uncomment the lines underneath. This will ensure your output is the properties of the largest blob in the image.

3. Use this function to complete Section D1 of line_following.py. You should use *binary* as the input to the function, as well as appropriate connectivity, minimum and maximum number of pixels. Keep the maximum blob size under 3000 as that will ensure the function works properly.

4. Section D2 is where we use the centroid of the line in the binary image to inform the QBot how to drive.

5. Go back to the qbot_platform_functions.py and navigate to **line_to_speed_map()** under **QBPVision()**. You will add controls to this function to enable line following.

6. Using the input parameter *col*, find the *error* to bring the QBot back center of the line. Consider the image is 320 pixels wide and you want to minimize the distance between the blob center and the image center. Figure 2 will help you visualize this.

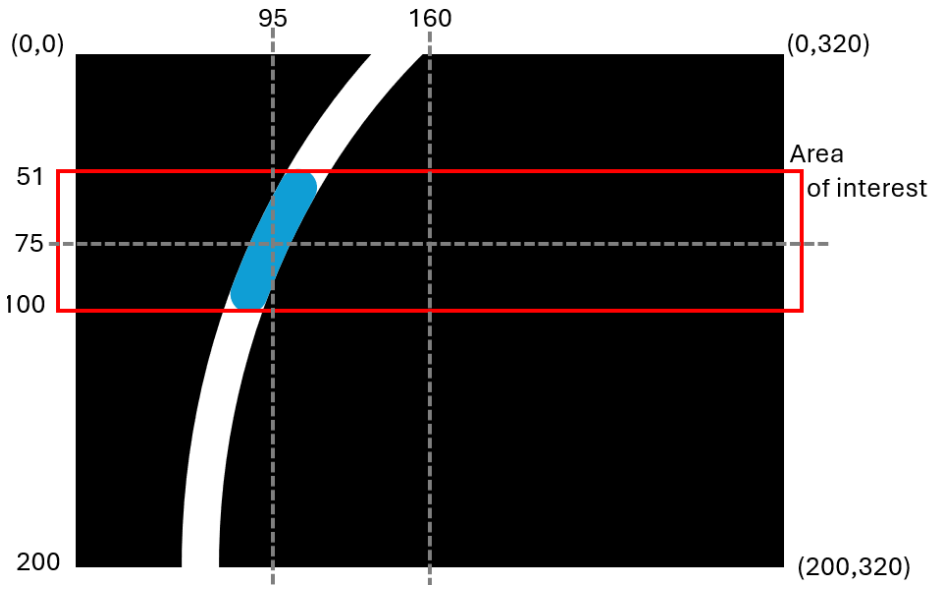

Figure 2. Image with the blob of interest in blue.

7. Using the variable *angle*, find the error angle of the blob center based on the bottom center of the image. Use Figure 2 as a reference to find the correct geometric function to use.

8. The *turnSpd* variable uses the parameters *kD* and *kP*. These two gains create a proportional-derivative controller for the robot (also referred to as Visual Servoing). You will tune these gains as your robot moves.

9. Set your forward speed to $3 * \cos(angle)$. This will make sure that the speed is inversely proportional to the error and the QBot drives faster when there is less error.

10. Add the *offset* to the *error*. This compensates for curves when finding the center of the line.

11. Go back to line_following.py. On section D4 change the predefined send values of 0, which correspond to the *kD* and *kP* gains for the QBot movement PD controller, to .2 and 0 respectively. You will tune these as you test your code.

12. Run observer.py first, then run line_following.py. While you keep the QBot armed, drive the robot so it sees the line to follow. Then press and hold the "7" key, and your line following code will be enabled.

13. Does your robot turn when it sees a line? Why? Stop pressing the "7" key and use the keyboard to bring the robot back on top of the line. Try this a couple of times.

14. Stop the code, increase the gains for *kD* and *kP* to tune the PD controller and test until you like the results. If the robot ever misses the line, let the "7" key go, drive over the line again manually and repeat the tuning process.

15. Observe your line following behaviour and let it run it for a bit.

16. When you are happy with the line following behavior, stop the code by pressing the "U" key. Ensure that you save a copy of your completed files for review later. Close Quanser Interactive Labs.