Lab Procedure for Python
Line Following

Setup

1. It is recommended that you review Lab 3 – Application Guide before starting this lab.
2. Turn on the QBot Platform by pressing the power button once. To ensure the robot is ready for the lab, check the following conditions.
    a. The LEDs on the robot base should be solid red.
    b. The LCD should display the battery level. It is recommended that the battery level is over 12.5V.
    c. The Logitech F710 joystick's wireless receiver is connected to the QBot Platform. Before use, **always make sure the switch on top is in the X position and that the LED next to the Mode button is off.**
    d. Make sure your computer is connected to the same network that the QBot Platform is on. If using the provided router, the network should be `Quanser_UVS-5G`.

Test connectivity to the QBot. Using the IP displayed in the robot's LCD display, enter the following command in your local computer terminal and hit enter: `ping 192.168.2.x`

3. Open line_following.py. In Section A change the value of the variable "ipHost" to the IP address of your local Windows machine. To get this IP, open a Command Prompt and enter `ipconfig`. Your IP can be found under IPv4 address and should look like the one the QBot has, `192.168.2.X`.

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.* Note: Python uses Numpy array which assume (height x width). OpenCV functions use (width x height)

    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 the code as per the steps in the Running Code section. Press the left button (**LB**) to arm the robot and keep it pressed. You should be able to drive it using the joysticks as you did in the previous lab.

5. On your windows 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 D1 and run the code again using the steps in the Running Code section. Do this as many times as needed.

Running Code

Note: You will be coding on your local Windows machine and then transfer the code to the QBot Platform. You must go through the steps in this section every time a change is made in the code.

1. Open WinSCP, enter the IP address of the QBot Platform for Hostname, and enter "nvidia" for both Username and Password, then click the login button.

2. In the WinSCP window, transfer required files to run the application:

    a. navigate to "~*Documents**Quanser**Mobile_Robotics*" and create a new folder "*Lab3*", as shown in Figure 1. Then copy the updated line_following.py to "*Lab3*". In addition, copy the qbot_platform_driver_physical.rt-linux_qbot_platform over. (You can just drag the files over).

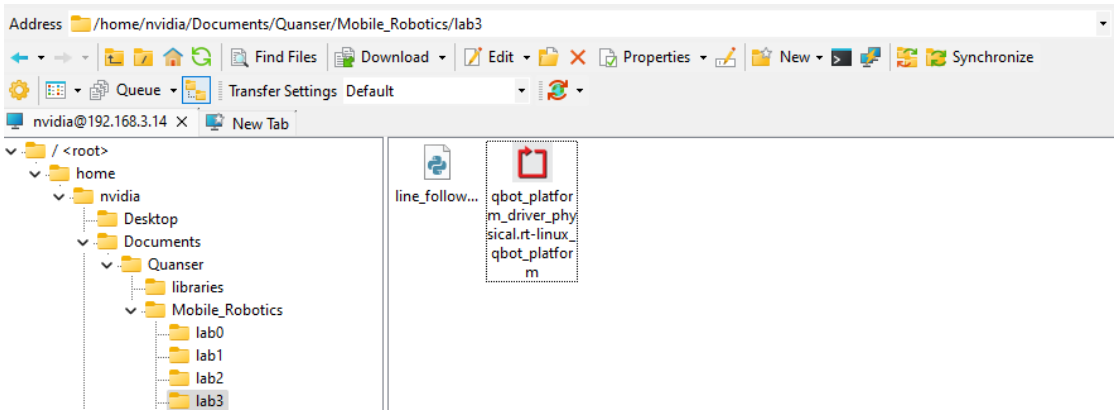

Figure 1. Correct directory for python script

    b. navigate to "~\Documents\Quanser\libraries\python" on the QBot Platform as shown in Figure 2, create these directories if they don't exist. Copy over the "hal" and "pal" folder from your local Windows machine, replacing existing files on QBot.

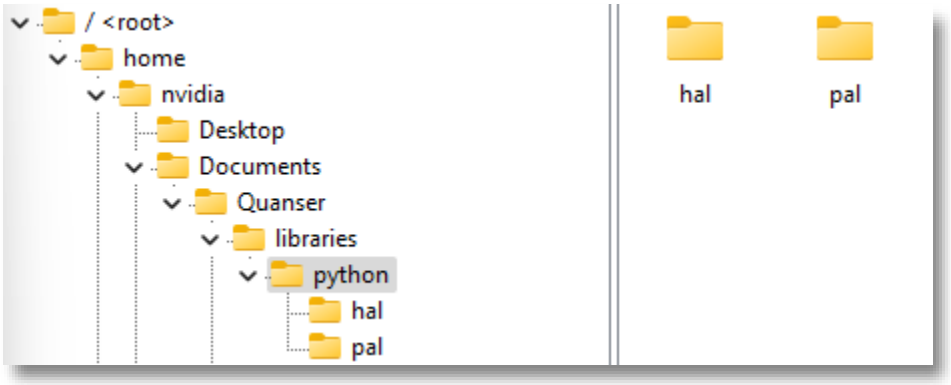

Figure 2. 'hal' and 'pal' folder in correct directory

3. Run observer.py on your local Windows machine first to initiate receiving data feeds from the QBot Platform.
4. Run line_following.py on QBot Platform:
    a. Open a PuTTY session by clicking  in the top bars of the WinSCP window.
    b. In the PuTTY terminal, enter the password "nvidia".
    c. Navigate to the python script directory as shown in step 2.a by using the `cd` command.
    d. Run the script using the following command:

        ```
        sudo PYTHONPATH=$PYTHONPATH python3 line_following.py
        ```

    e. When the script is run successfully, User LEDs will turn blue.

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 D1 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* representing the difference between the center of the line and the center of the image, which you want to minimize. Figure 3 will help you visualize this.

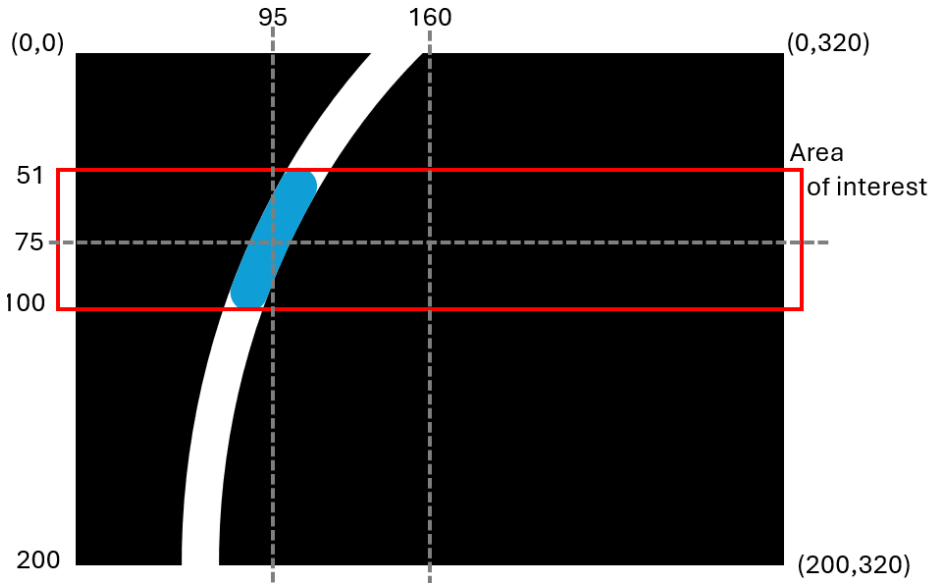

Figure 3. Image with the blob of interest in blue.

7. Using the variable *angle*, find the error angle of the blob center based on the bottom center of the image. Use Figure 3 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 $0.3 * \cos(angle)$. 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 D2 change the predefined send values of 0, which correspond to the *kP* and *kD* gains for the QBot movement PD controller, to 1 and 0 respectively. You will tune these as you test your code.

12. Run the code as per the steps in the Running Code section. While you keep the QBot armed, drive the robot so it sees the line to follow. Then press and hold the "A" button, and your line following code will be enabled.

13. Does your robot turn when it sees a line? Why? Stop pressing the A button and using the joysticks, 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 A button go, drive over the line again manually and repeat the tuning process. Every time changes are made, the steps in the Running Code section must be repeated.

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 right button (RB). Ensure that you save a copy of your completed files for review later.

17. Turn OFF the robot by single pressing the power button (do not keep it pressed until it turns off). Post shutdown, all the LEDs should be completely OFF.