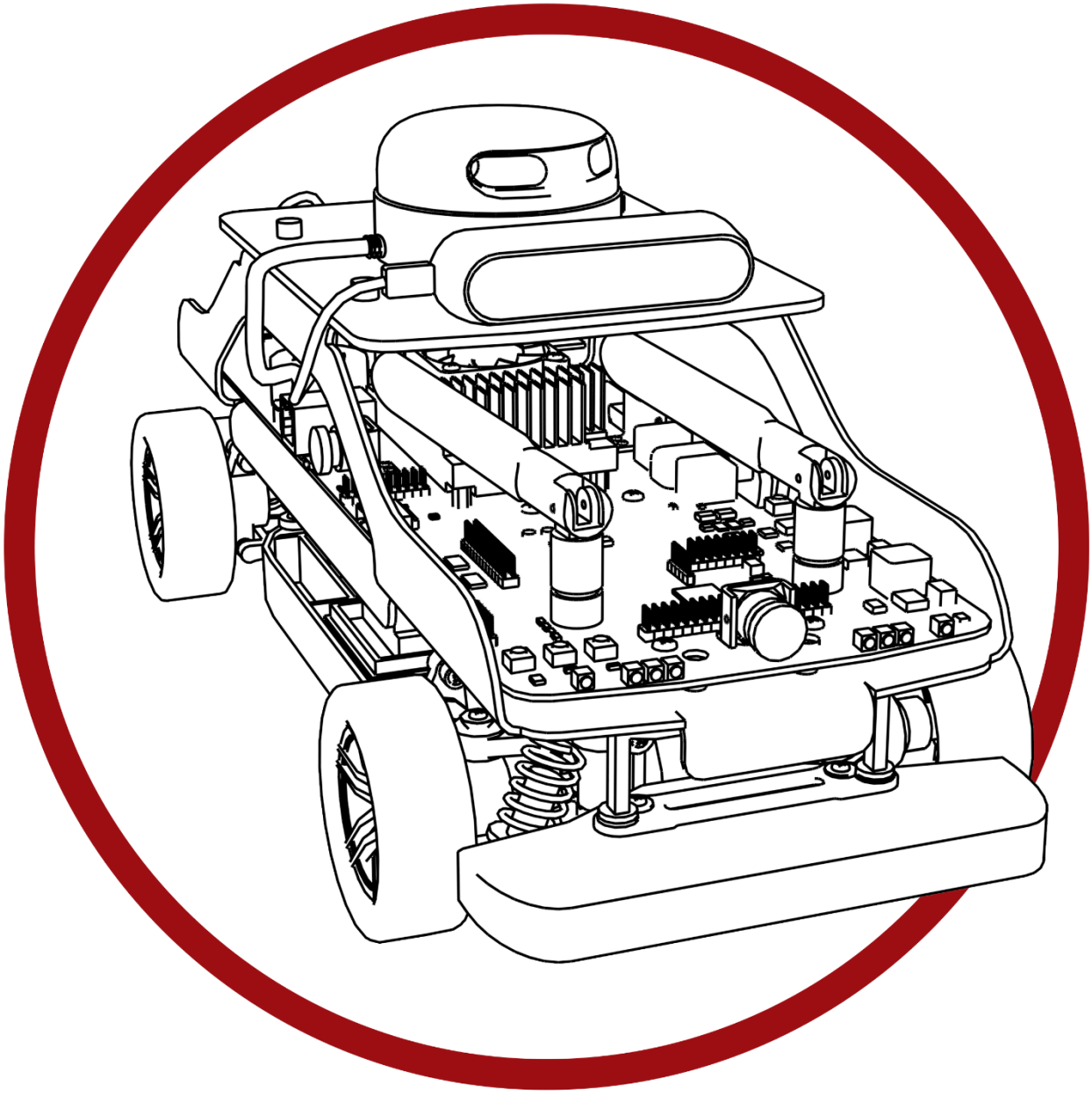
QUANSER
INNOVATE·EDUCATE

Self-Driving Car Research Studio



Lane Following – Python

V 1.0 (February 2021)

Table of Contents

I. System Description

In this example, we will use an analytical method to keep the car following a yellow lane. The camera image data is captured and we use image processing and analysis to output a steering command. The gamepad controller outputs a car speed and will overtake the steering if needed. The process is shown in Figure 1.

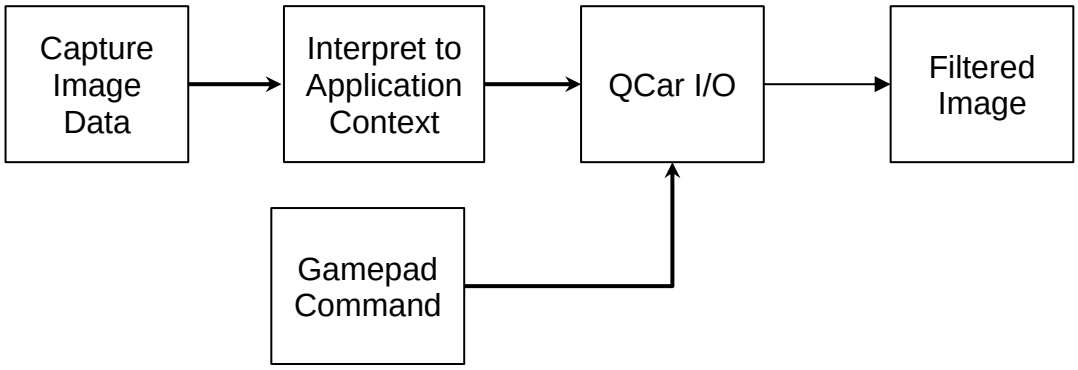


Figure 1. Component diagram

II. Running the example

Check the user guide **V - Software - Python** for details on deploying python scripts to the QCar as applications. Please make sure to assign the right event number to the joystick initializer **gamepadViaTarget()**.

Place your QCar on the right side of the yellow lane. Once you have set the controller number correctly for the gamepad (see Python Hardware Test documentation for details), run the script using a **sudo** flag. A **CV2** window shows the binary image of the yellow lane the QCar captures. If there is no lane or the lane is grainy, adjust the **HSV** upper and lower bounds. Press **X** to enable the automatic steering. Press **RT** to provide throttle. If the QCar failed to follow the lane, release **X** to have manual control to the QCar. Use the **left stick** to manually steer.

Note: If the manual steering does not appear to work, please ensure the **mode** light on the gamepad is **off**.

III. Details

1. Image Processing

   We leverage the functionality of **OpenCV** in this application. After the image is cropped to let it focus on the lower half of the image frame, we use **cv2.cvtColor** to convert the image format. Please visit **OpenCV** official website to check out more functionalities. After the image is transferred to HSV format, **binary_thresholding** takes the HSV image and thresholds it based on the defined boundaries. This is the line where we can change to threshold difference colors.

```
# Convert to HSV and then threshold it for yellow
hsv_buf = cv2.cvtColor(cropped_rgb, cv2.COLOR_BGR2HSV)
binary = binary_thresholding(hsv_buf, lower_bounds=np.array([10, 50, 100]),
upper_bounds=np.array([45, 255, 255]))
```

2. Performance considerations

   Raw_steering angle is controlled by the **slope** and **intercept** taking from **find_slope_intercept_from_binary**. Imagine a straight line crossing the first quadrant of an axis. The **slope** is the gradient which controls the turning angle. When the road is bending to the right, the **slope** decreases and vice versa. The **intercept** is the intercept value with the vertical axis which decides the distance that QCar tries to keep away from the yellow lane. The smaller the number, the closer it will get to the yellow lane. This is the line where we can adjust the **slope** and **intercept**.

```
# steering from slope and intercept
raw_steering = 1.5*(slope - 0.3419) + (1/150)*(intercept+5)
```