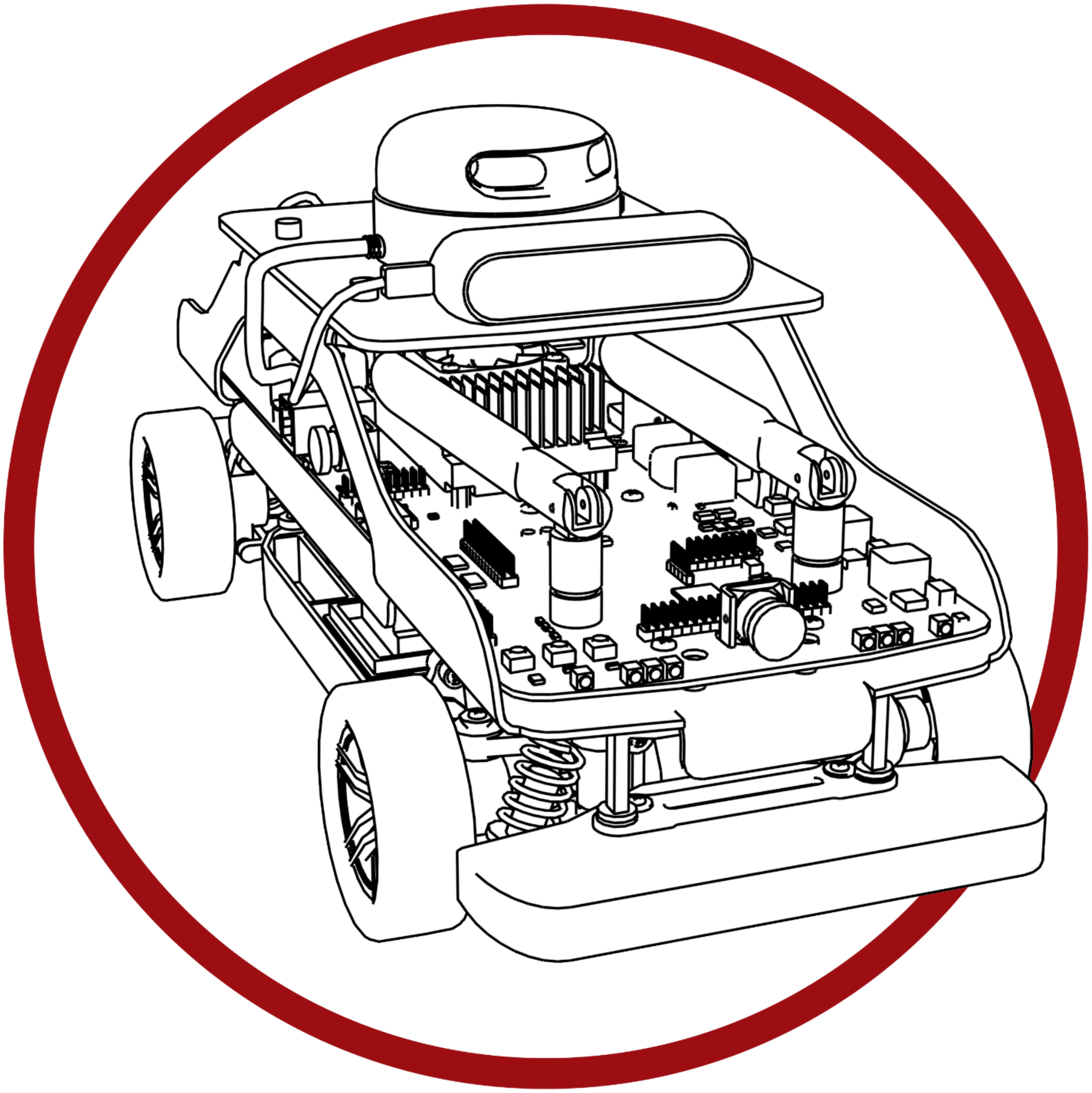
QUANSER
INNOVATE·EDUCATE

Self-Driving Car Research Studio



RGBD Imaging - Python

V 1.1 (November 2020)

Table of Contents

I. System Description

In this example, we will capture images from the Intel Realsense's RGB and Depth cameras. After thresholding the Depth image based on a minimum and maximum distance, a filtered binary mask is applied to RGB Image to only show the image within that range.

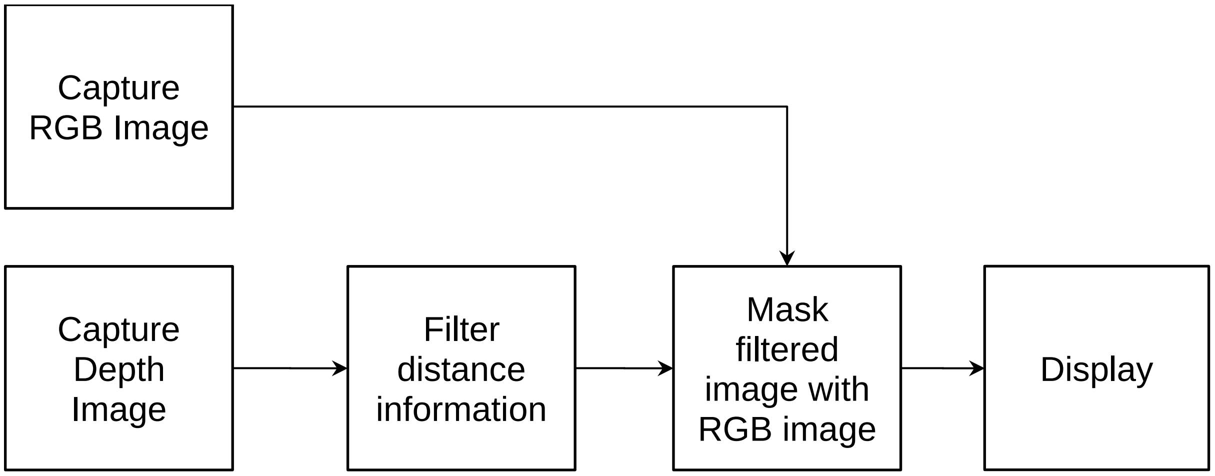

Capture RGB Image
Capture Depth Image
Filter distance information
Mask filtered image with RGB image
Display

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.

In this example, users only need to define the image frame size and the maximum & minimum distance for the depth image filtering. After executing the script on the QCar, the results should look similar to Figure. 2. (Another QCar is sitting in the front as an object.)

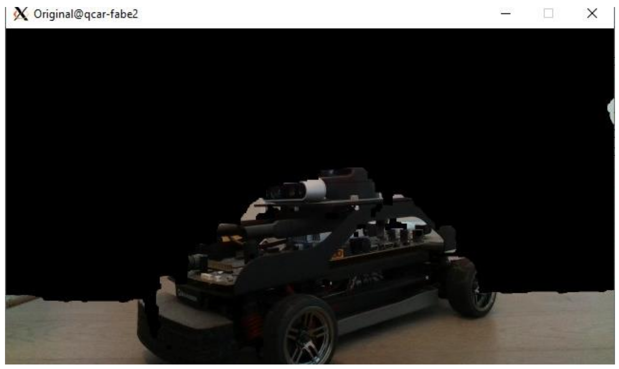


Figure 2. RGBD outputs showing the filtered image based on distance thresholds

III. Details

1. Depth image thresholding

   binary_thresholding() (that is provided with the **q_interpretation.py** core library) automatically detects 3-color or grayscale images and correspondingly thresholds them using the bounds provided. In this example, it thresholds based on the maximum and minimum distance that users define. The result is a binary image (numpy array) that is **1** within the threshold bounds and **0** elsewhere.

```
binary_now = qi.binary_thresholding(myCam1.image_buffer_depth_m, min_distance,
max_distance).astype(np.uint8)
```

2. Clean up the noise from the RGB-D sensor

   To clean up the noisy depth image, two filters are applied. This first consists of a **temporal difference** filter to remove single frame random noise. By calculating the difference between the current and previous frames, and removing those pixels in the current frame, we remove noise between frames. For a new pixel to get registered, it must last at least 2 frames. This introduces a maximum delay of **sample_time** in the system, which is set to 1/30 s. Next, a **spatial closing filter** (morphological dilation and erosion sequence) is applied to fill holes and clean up edges.

```
binary_clean = qi.image_filtering_close(cv2.bitwise_and(
cv2.bitwise_not(np.abs(binary_now - binary_before)/255), binary_now/255 ),
dilate=3, erode=1, total=1)
```

3. Image correction between the depth camera and RGB camera and mask

   The field of view is different on both cameras and there is also a physical distance between them. The first line handles this transformation. This adjusted mask is then applied to the RGB image.

```
binary_clean = cv2.resize(binary_clean[81:618, 108:1132], (1280,
720)).astype(np.uint8)

masked_RGB = cv2.bitwise_and(myCam1.image_buffer_RGB, myCam1.image_buffer_RGB,
mask=binary_clean)
```