Lab Guide
360 Vision

Content Description

The following document describes a lane following implementation in either python or MATLAB software environments.

Lab Description

In this example, we will capture images from the four CSI cameras at the same resolution and frame rate. These will be stitched together, and passed to a display module.

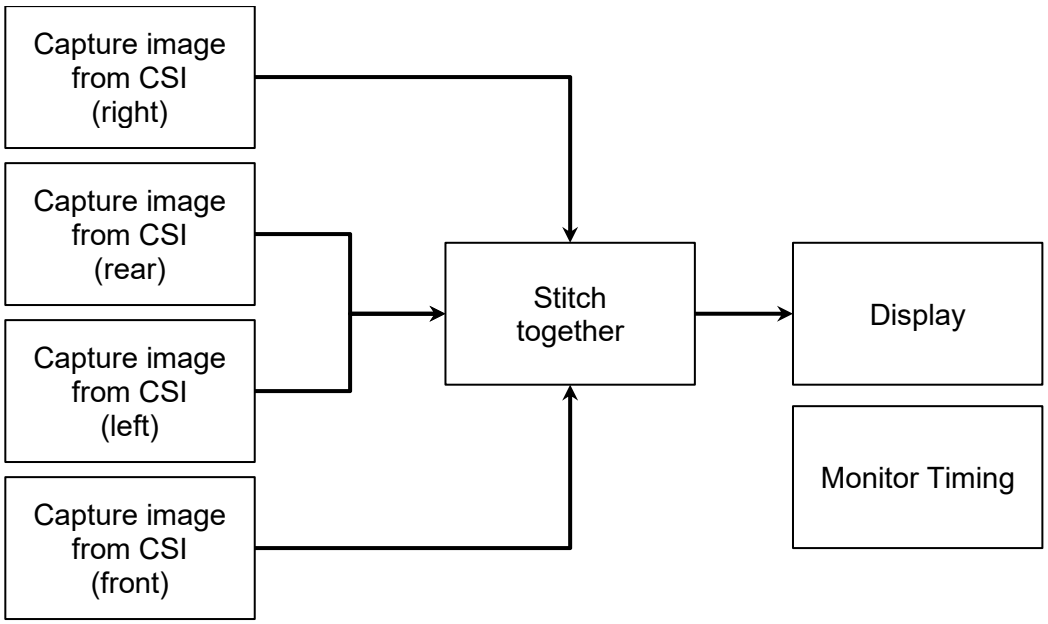

Figure 1. Component diagram

MATLAB

The Simulink implementation is displayed in Figure 2 below.

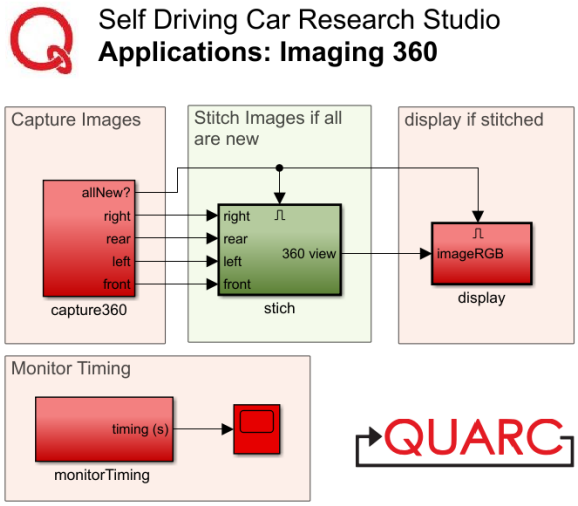

Figure 2. Simulink implementation of 360 Vision

Running the example

Check User Manual – Software Simulink for details on deploying Simulink models to the QCar as applications. The output in the **Video Display** block should look like Figure 3. (The order of stitching is **Right, Rear, Left,** and **Front)**

Figure 3. 360 view output showing images stitched together

Details

The implementation has a few important elements that help boost performance. The model is deployed at 30 Hz with each camera streaming a 640 x 480 RGB (3 channels) unsigned integer image (8 bits or 1 byte per data sample). The data rate can be estimated as,

$$30\ \frac{steps}{s} \times (640 \times 480 \times 3)\ \frac{samples}{image} \times 1\ \frac{byte}{sample} \times 4\ \frac{images}{step}$$

This ends up as $105.47\ Mbps$ of data streamed over Wi-Fi. However, asynchronous image acquisition between the four capture loops might yield cases when at least 1 of the 4 images might not be new, and this step can be skipped.

In the implementation provided, the images are stitched together only when all the images are new. The **Video Capture** block in the **capture360** module also outputs a **new** signal that is high (**1**) when the image is new. The four **new** signals are passed into a logical AND, which is then used to stitch images and display them, improving performance.

Note that a simple matrix concatenate method is used here to stitch the images together side by side. Alternatively, you may consider lens distortion correction on the raw images first, and using feature matching to remove translational and rotational discrepancies in the placements of the 4 cameras.

Python

Running the example

Check User Manual – Software Python for details on deploying python scripts to the QCar as applications. The output in the **cv2.imshow()** function should look like Figure 2.

Figure 2. 360 view output showing images stitched together

Details

In this example, we move the front camera feed to the centre of the 360 degree video stream and split the rear camera stream to the left and right extremes. This aligns the driving orientation of the vehicle with the camera feed, making first-person-view driving easier. We also place black padding around the camera feeds to ease differentiating them, as seen in the snippet below.

```
# Stitch images together with black padding
        horizontalBlank     = np.zeros((20, 4*imageWidth+120, 3), dtype=np.uint8)
        verticalBlank       = np.zeros((imageHeight, 20, 3), dtype=np.uint8)

        imageBuffer360 = np.concatenate(
                                        ( horizontalBlank, np.concatenate
                                          (
                                             (  verticalBlank,
                                            myCam2.image_data[:,320:640],
                                               verticalBlank,
```

```
                                        myCam3.image_data,
                                        verticalBlank,
                                        myCam4.image_data,
                                        verticalBlank,
                                        myCam1.image_data,
                                        verticalBlank,
                                    myCam2.image_data[:,0:320],
                                        verticalBlank
                                     ), axis = 1
                                ),
                    horizontalBlank )
                        , axis=0 )
```

The script runs in a **while** loop at 30Hz The **cv2.waitKey(msSleepTime)** will pause the loop for the time difference between the **sample time** and **computation time** (please refer to User Manual – Software Simulink **Section B. Timing** for more information.)