ENPM 809T – Autonomous Robotics

HW 3

Bala Murali Manoghar Sai Sudhakar

116150712

Step 1: HSV Masking

1. Image is captured using Raspberry Pi camera as RGB image (Figure 1 a)
2. Captured image is converted into HSV space for robustness since color and intensity are separate in HSV space. (Figure 1 b)
3. A mask is generated by tresholding the for green color. The mask is used to separate required region of captured image (Figure 1 c)

(a) (b) (c)

Figure 1

Step 2: Contours and Bounding circle

1. On the masked image in (Figure 1 c), Eroding and diatation are done to remove small green areas.
2. Contours are found on masked image.
3. Enclosing circle is found for the identified contour.
4. Moment for the identified countour is found.

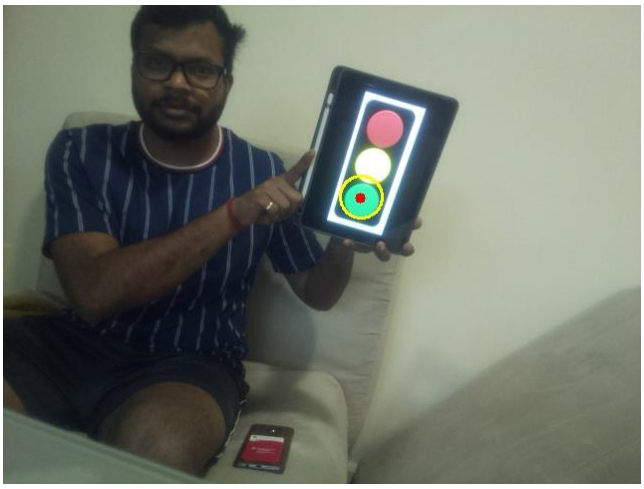

Figure 2

Step 3: Video Feed

1. The functions created for above steps are imported.
2. Frames are captured continuously from raspberry pi camera.
3. The captured frames are processed to identify the green patch.
4. The processed image is saved to a video file.

Step 4: Video Upload

Video File Link - https://youtu.be/RtK8KMuwdvE

Step 5: Analysis of hardware perfomance limitaions

1. Using date time module, time required to process one frame is calculated.
2. The timing metrics are logged to text file.
3. Perfomace graphs are generated as shown below

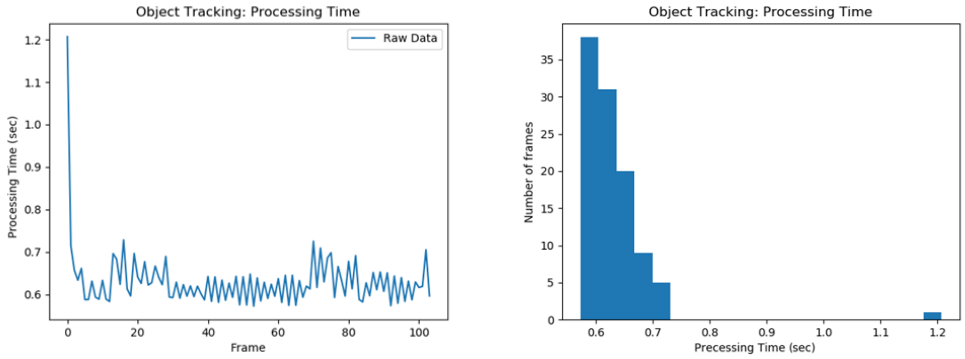

Figure 3 – Metrics with object detection

It can be noted that on an average using raspberry pi and the pipeline specified in above steps, we are able to achieve ~0.6 fps. Considering frame rate of normal videos being ~60 fps, the fame rate we got is very low. When we use this small frame rate for safety critical applications such as obstacle avoidance if there are fast movements, we will not be able to detect and avoid the obstacle. Thus, this method is suitable for taking high level decisions such as goal point detection etc, but the entire navigation stack cannot rely on this method.

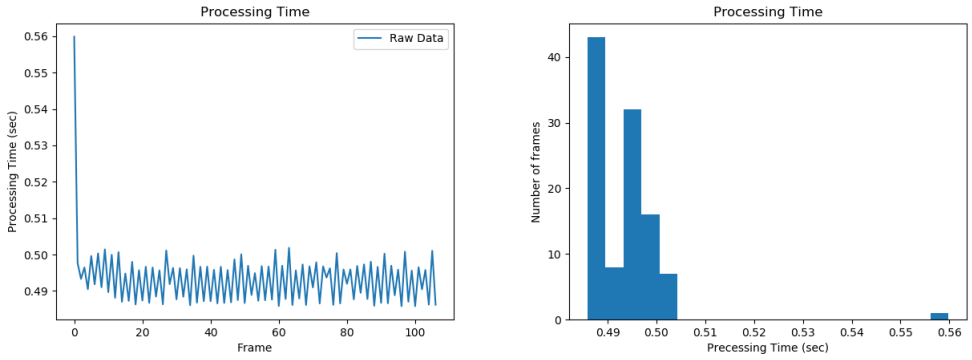

Figure 4 – Metrics without object detection

The Figure 4 shows the metrics without object detection, display of captured frames and saving to video. It can be seen that on an average the frame rate is ~0.45 fps which is 150ms lower than the previous case which has object detection and display included. Thus the bottle neck is not the detection part but raw dta capture itself.