Senior Design EC463
Hardware Mini-Project Report

This project required us to configure a raspberry pi, install the OpenCV library and write an algorithm that could take video recordings of roads and count the amount of cars passing by.

The original algorithm we used tracks cars at approximately 80% accuracy. There are a few notable errors of the system that we implemented. First of all, the way we count the number of cars is flawed. We get a count frame by frame and then compare it to the previous frame's count. If the new count is larger, then we add the difference between the frames to the count. Otherwise, the counter stays steady. The problem arises when cars leave and enter in the same frame, making the counts equal; it entirely neglects this new car. Other algorithms address this by tracking each car it identifies and only counting it once it crosses a threshold line.

There are other fundamental flaws in the original algorithm: it double counts cars and it will sometimes identify phantom cars. The double counting is a phenomenon that occurs when there is a secondary color on or nearby the car. The algorithm draws a second circle, sometimes around a large distinctly colored bumper. Or it may interpret a similarly colored rear light of a nearby car as part of the car. It will also identify parts of road as a car. This could be due to improper video stability (something we see a remarkable amount of in our homemade video). Either way, with a clean video from an overpass, our algorithm detects 8 out of 10 cars. This is something we can account for and predict reasonably well.

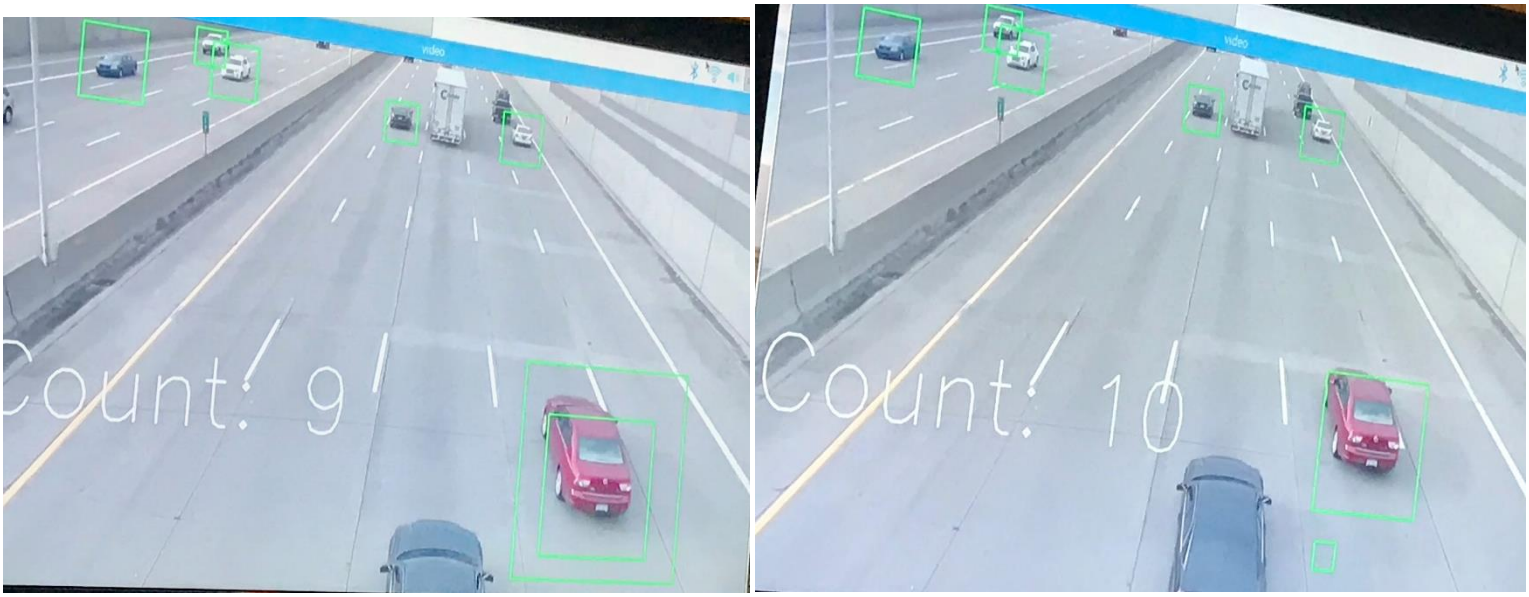

Figure: Shows live counter of the cars passing by and being detected by the algorithm running on the raspberry pi.

The improved algorithm we used for the car detection was taken from a public repository. Specifically, this improved algorithm creates a video capture object that reads from the camera recorded video file. The video then segments the video frame by frame. Each frame is converted to grayscale. By converting the RGB image to grayscale, we reduce all the noise that is introduced by car colors that may impact the image processing. Furthermore, by converting all the frames to grayscale we are able to find edges simply based on luminance whereas with an RGB frame image we would need to rely on luminance and chrominance increasing the complexity of the algorithm. After detecting the car, a simple rectangle is drawn to the perimeter of the car to allow us to figure out how many cars the algorithm is counting.

That being said, the homemade videos do not yield as clean of a result. Taking videos of Mass Pike from Photonics, the video detects a great number of false positives, not only of the

road, but also from the trees above. It fails to detect certain cars in the background of the frames possible due to the lack of edge detection.

In order to improve the overall system, stable video footage is essential. Potentially a board stabilizer or a better camera could record a cleaner video allowing for more accurate car image detection.

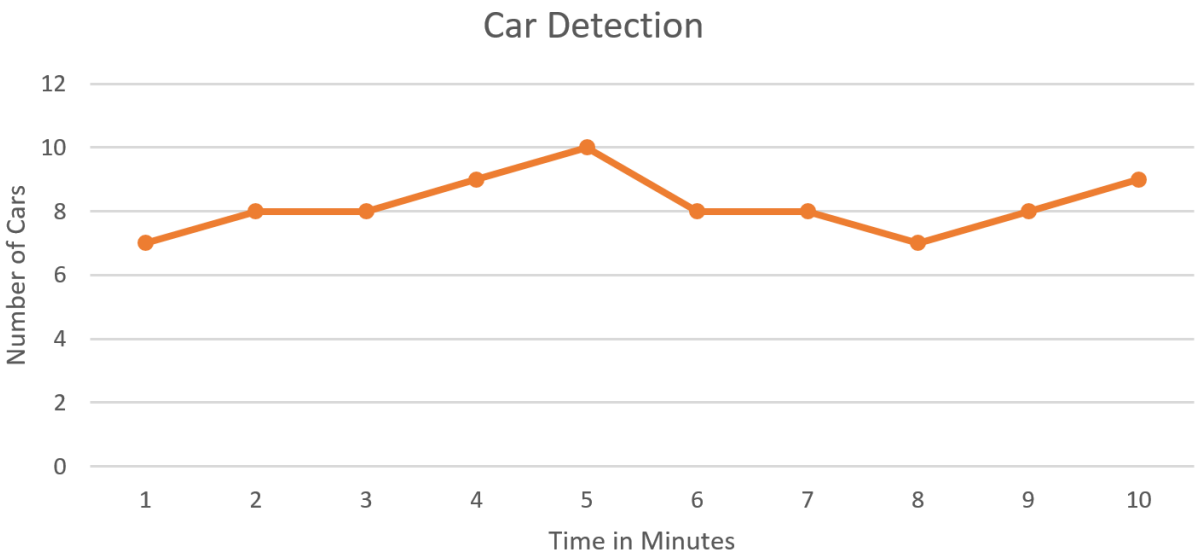

Figure 2: Car Detection Data captured from the pi