Evan Gonzales, Jonas Shuai

EC463 Senior Design I

Hardware Mini Project Report

In EC463, we are presented with a wide array of engineering design projects that we must choose from. All of these projects have some basis in Electrical and/or Computer Engineering and will be tackled by a multidisciplinary team of ECE students. Being the electrical engineering students, we will be the hardware experts on our team. This will most likely require knowledge of embedded computing as well as video streaming. Many of us, however, have little to no experience working with micro-computers or video streaming. Therefore, this project gives us a chance to learn about and work with an embedded operating program before working on our more daunting senior project.

The objective of this mini project is to count vehicles passing by on the road with a Raspberry Pi. We were given a Raspberry Pi Zero, camera module, and many other accessories necessary for the project. We chose to use the Raspbian Stretch OS for the Pi because of its simple install process. From there, we edited the boot files on the OS in order to SSH into the Pi. We then installed opencv-python in order to capture and analyze video. We chose opencv-python because it's a basic version of OpenCv that takes up very little space, doesn't take long to download, and contains everything we needed. In addition, we searched online for helpful resources and concluded that opencv was one of the most widely used computer vision library. This made it easier to write programs in order to accomplish our goals. In addition, we found an xml file that detects cars on the internet that seems to work well enough. Since neither of us have

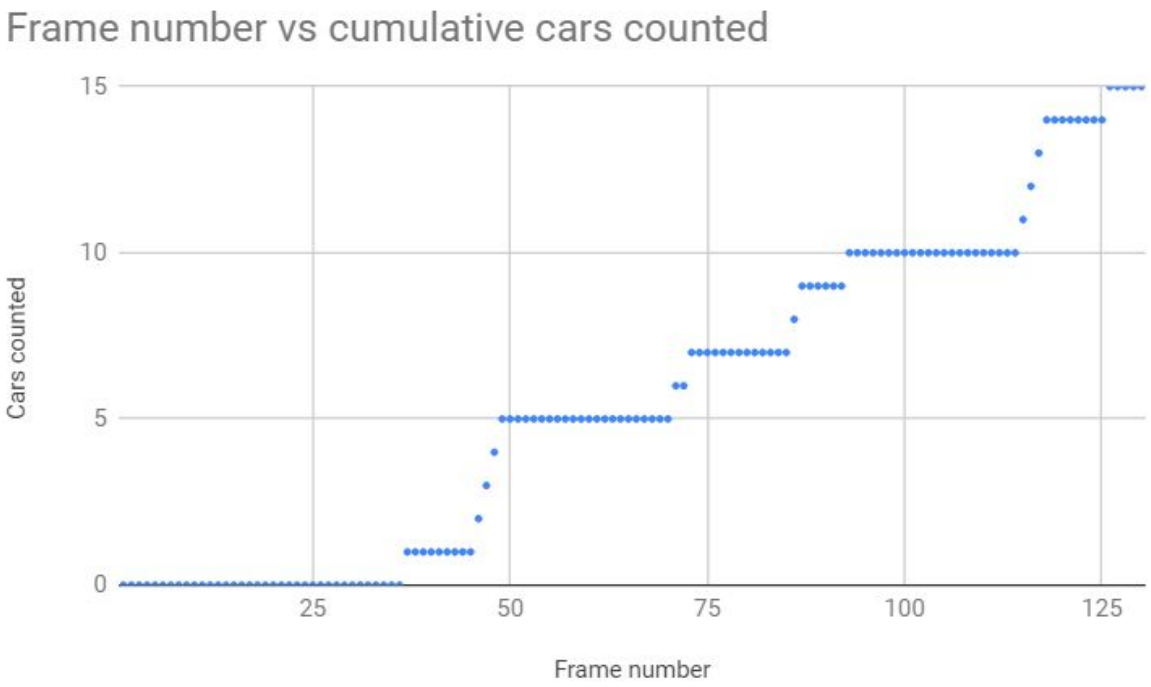
much experience in these topics, we decided to focus on functionality of the project as opposed to program optimization and therefore weren't as concerned with performance.

We count cars by first detecting a car in a frame of the video, then drawing a rectangle around it. When the bottom line of that rectangle is between a very narrow (3 pixel) section of the video, a car counter is incremented.

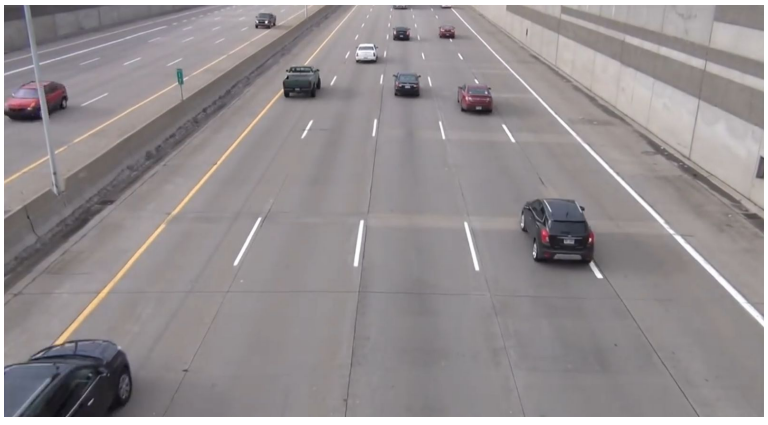
Our program currently works as follows:

1. Go to bridge perpendicularly overlooking a road
2. With a laptop, SSH into the Pi (no monitor required)
3. From above, aim camera down at road (~40 degrees downward).
4. Run python video_rec.py. This records a 20 second video of footage and saves it to "car_video.h264"
5. Run python car_detect.py. This takes "car_video.h264" and, for each frame, prints (# of cars counted, frame #) as well as writing (# of cars counted, frame #) to the text file data.txt. If a monitor is connected, it will display the frame-by-frame analysis.

After running our program, we plotted the data in data.txt onto the following graph:



There are several limitations of this code. First of all, it relies heavily on being able to detect a car in the frame. I'd roughly estimate that it was 95% successful at that. If it didn't detect a car in the frame, it couldn't be counted. Next, the program itself isn't fully accurate. For example, for 130 frames of video it counted 15 cars passing when there were only about 11 cars that passed. Clearly it has an over counting problem, most likely due to the width of the incrementing strip being too large. Finally, the counting is contingent on the way that the video is taken. The video must look similar to as follows:



This is because the narrow lines that count the cars are simply static horizontal lines across the frame. Our program wouldn't work for a video taken on the side of a street. A more modular program that can count cars in a video regardless of camera orientation or angle would be more ideal. We also were not able to test the program in the at night, but would assume that there would be accuracy issues due to low lighting.

While our program does function, there is still room for improvement. For instance, the current program only takes a 20 second video. If one wanted to take a longer video, you would have to change the sleep(20) function in video_rec.py. Ideally, this could be a user defined value. Moreover, this requires computer power and the ability to SSH into the Pi. Ideally, the Pi could be plugged into a 5V source and the programs can run on start-up. In addition, the data could be plotted and saved on the Pi instead of in a simple text file. Finally, there is most likely a way to combine both executable python files into one as opposed to two. Had we more time and/or experience, these are the additions we would make.

Very helpful sources:

https://raspberrypi4u.blogspot.com/2018/06/raspberry-pi-car-detection-with-opencv.html

https://desertbot.io/blog/headless-pi-zero-ssh-access-over-usb-windows

https://projects.raspberrypi.org/en/projects/getting-started-with-picamera

https://pypi.org/project/opencv-python/

https://github.com/BostonUniversitySeniorDesign/hardware-project-2018/wiki