HW Mini Project Report

By: Marta Taulet & Leonard Gunawan

1. Introduction

Climate change is a crucial problem in our generation and one factor that has a significant impact on it is air pollution. Air pollution is caused by factories, automobiles and aerosols. To prevent further damage due to air pollution, we must understand its causes, specifically, the effect of car emissions. Hence, we will be looking at the relationship between the number of cars in a certain area within a specific time frame.

For this project, we decided to use a microcontroller (in this case, a Raspberry Pi Zero W) to detect and count the number of automobiles heading west on the Massachusetts turnpike at around 2pm in a sunny afternoon. This can be done by using the camera capabilities of the Raspberry Pi Zero W and a video processing algorithm.

2. Algorithm

Our logic for detecting motion used the motion data vectors from the frames in the video. It compares the average of the x and y motion data values from a predetermined amount of frames against a predetermined threshold value. If the motion data change was large enough (the variation was larger that the threshold) then we consider that a movement. If that movement is the same as the last movement (for instance, if the car was moving to the left and continues to move to the left two seconds later) then we consider that to be the same object as before and we do not count it as a new car.

We decided to take a 1 minute video that records our car count every 5 seconds. The reasoning for this is so that we can see any anomalies in the data in smaller intervals. In the beginning, we had a few anomalies such as no cars were observed in 5 seconds when there were clearly cars passing on the road. Fortunately, we were able to calibrate our code by changing the number of consecutive frames to analyze as well as the minimum threshold for a movement to be detected, which changed the sensitivity of the camera to detect motion.

The benefit of using a user defined function is the ability to customize a function as to our needs. What we did with our algorithm is essentially further customizing a user defined function (GestureDetector) to fulfil our needs. User defined functions allow us to be more flexible and generate an output suitable to our need. However, the downside of user defined functions is the time taken to come up with one. Sometimes, coming up with the logic behind a user defined function can take longer than expected due to issues with accuracy and syntax. Furthermore, there are libraries such as numpy which was very useful for image processing. With the time constraint of this project, it is reasonable for us to reuse code and come up with a user defined function. We might end up not completing the project if we were to come up with a user defined function from scratch.

3. Results

The output of running our program is a text file called ‘data.txt’ containing two arrays, both of length 12. The first array represents the amount of cars observed every interval of five seconds during one minute. The second array represents the elapsed time from the beginning of the video.

The figure below shows a plot of the cumulative sum of the number of cars observed against time. The data used for the plot is actually an average of five one-minute samples that were recorded. On

average, there were about 71 cars per minute that were observed. In addition, the data's R^2 value shows that the data does not really fluctuate, indicating a reasonable amount of cars every time interval.

Average Number of Cars / 5 Sec	11	6	4	5	11	6	7	3	3	1	6	7
Cumulative Sum of Number of Cars / 5 Sec	11	17	21	26	37	43	50	53	56	57	63	70
Time / Sec	5	10	15	20	25	30	35	40	45	50	55	60

Table 3.1 Average Number of Cars/ 5 Sec and Cumulative Sum of Number of Cats / 5 Sec

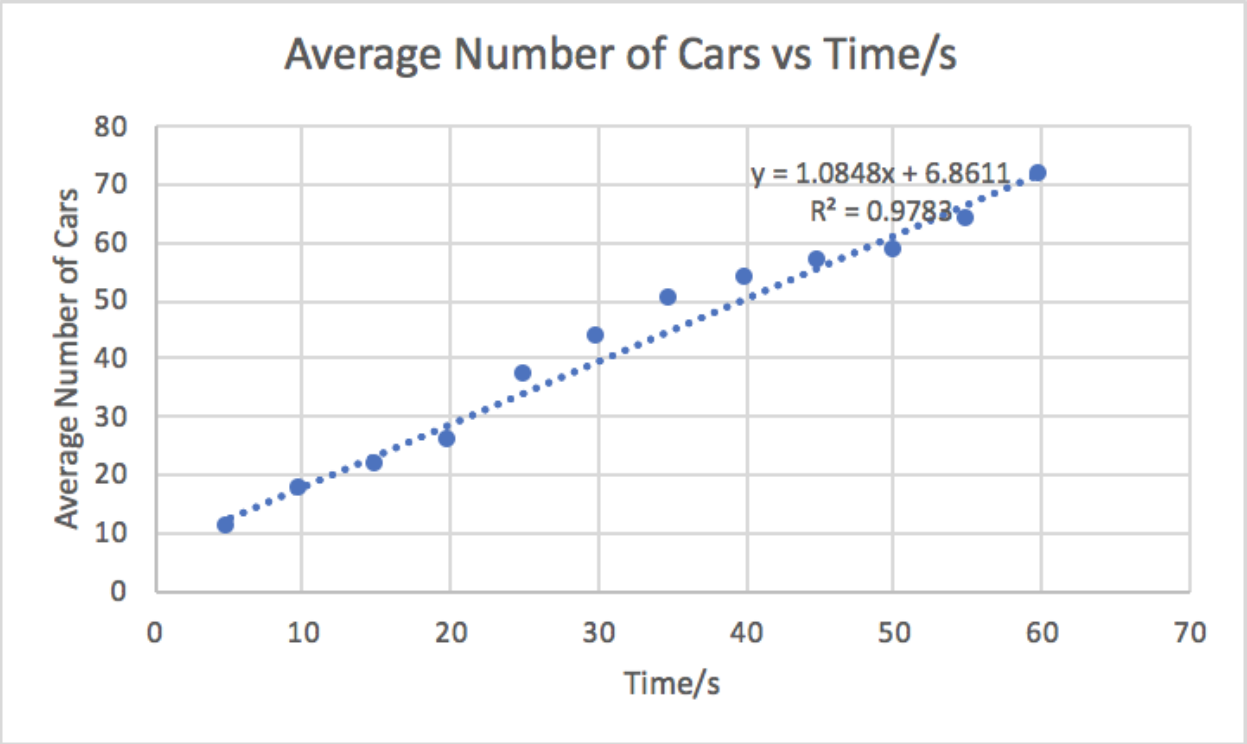

Fig 3.1 Plot of Average Number of Cars vs Time

4. Problems and Solutions

Some of the problems that may cause inaccuracies in the measurements are the shaking of the camera when placing it on an unstable support or holding it by hand. If the background is not stable, the camera detects motion when there are no cars passing so the results may be higher that the actual value. In order to maximize the accuracy, the camera should be mounted on a steady support.

Another modification that can be made is the addition of a portable power source to power the Pi (for example, a battery or solar cell). For the Pi to be able to function when mounted on a pole, it would need a portable power source since it can't be directly connected to a laptop or a socket.

The positioning of the Pi is also crucial for the counter to be accurate. The counter will not be able to detect cars covered by another car. If a car is in motion but is covered by the car in image Fig 4.1, the counter will not be able to detect it. To prevent this from happening the camera should be positioned so that it can capture an image like Fig 4.2.

Fig 4.1 Image of a car

Fig 4.2 Image of cars from the top

5. Individual Contribution

Leonard Gunawan:

1. Purchase of Micro SD card, OS installation & ssh.
2. Setting up the camera and case
3. Optimized threshold and frame queue parameters to count correctly
4. Sampling and plotting data through Excel spreadsheet

Marta Taulet:

1. Research on video processing and image recognition algorithms
2. Setting up github environment
3. Setting up file communication with the Pi through scp command
4. Python syntax

6. Important Links

Raspberry Pi Camera tutorial
https://picamera.readthedocs.io/en/latest/recipes1.html

Car Images, Fig 4.2
http://paulsaulnier.com/2010/07/12/empire-state-building-brooklyn-bridge-new-york-6-of-7/