Spring 2023

# A Camera-Only Based Approach to Traffic Parameter Estimation Using Mobile Observer Methods

Temitope D. Jegede

# A CAMERA-ONLY BASED APPROACH TO TRAFFIC PARAMETER ESTIMATION USING MOBILE OBSERVER METHODS

by

TEMITOPE JEGEDE

(Under the Direction of Ryan Florin)

## ABSTRACT

As vehicles become more modern, a large majority of vehicles on the road will have the required sensors to smoothly interact with other vehicles and infrastructure on the road. There will be many benefits of this new connectivity between vehicles on the road but one of the most profound improvements will be in the area of road accident prevention. Vehicles will be able to share information vital to road safety to oncoming vehicles and vehicles that are occluded so they do not have a direct line of sight to see a pedestrian or another vehicle on the road.

Another advantage of these modern connected vehicles is that different traffic parameters can be more easily estimated using the onboard sensors and technologies in the vehicles. For many decades traffic engineers have been able to estimate different traffic parameters like traffic flow, density, and velocity based on how many vehicles the primary vehicle passes and how many vehicles pass the primary vehicles. For much of the time that traffic engineers have been working on traffic estimation, it has been done using more manual and tedious methods. In this paper, a more novel approach of determining these traffic parameters is used.

Also, one of the problems with traffic parameter estimation is that sometimes the results are not accurate because of vehicles that might not have been counted because of occlusion. In this paper, a proposal is put forward on how this can be remedied utilizing the connected vehicle's framework.

INDEX WORDS: Mobile observer methods, Vehicle tracking

# A CAMERA-ONLY BASED APPROACH TO TRAFFIC PARAMETER ESTIMATION USING MOBILE OBSERVER METHODS

by

Temitope Jegede

B.S., Georgia Southern University, 2021

A Thesis Submitted to the Graduate Faculty of Georgia Southern University

in Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

# A CAMERA-ONLY BASED APPROACH TO TRAFFIC PARAMETER ESTIMATION USING MOBILE OBSERVER METHODS

by

TEMITOPE JEGEDE

| | |
|---|---|
| Major Professor: | Ryan Florin |
| Committee: | Andrew Allen |
| | Lixin Li |

Electronic Version Approved:
May 2023

# DEDICATION

I dedicate this to my family and friends who have encourage me throughout my master's program, I would not be able to do it without them.

# ACKNOWLEDGMENTS

I want acknowledge Dr. Florin for guiding, directing, encouraging, and assisting throughout writing this thesis. His advice kept me on track and steered me in the right direction from the beginning. I also want to acknowledge Dr. Allen and Dr. Li for being a part of my thesis committee.

TABLE OF CONTENTS

Page

ACKNOWLEDGMENTS…… 3
LIST OF TABLES…… 5
LIST OF FIGURES…….. 6
CHAPTER
1 INTRODUCTION…… 8
Motivation……..... 8
Technical Background ……...…… 9
Research Question ……..…… 18
2 RELATED WORKS……...……. 19
Distance Estimation…… 19
Mobile Observer Methods.…… 21
3 METHODOLOGY…….. 24
4 EXPERIMENTATION AND RESULTS……..…… 34
5 CONCLUSION…….. 51
6 FUTURE WORK……. 52
REFERENCES ..... 53

# LIST OF TABLES

Page

Table 1: Comparing the measured distance with an expected distance of 12 inches……..…...31
Table 2:. Statistical Metrics for results from Table 1……....31
Table 3: Comparing the measured distance with an expected distance of 24 inches……...…32
Table 4: Statistical Metrics for results from Table 3…….32
Table 5: Comparing the measured distance with an expected distance of 36 inches……....34
Table 6: Statistical Metrics for results from Table 5…….34
Table 7: Comparing the measured distance with an expected distance of 48 inches……35
Table 8: Statistical Metrics for results from Table 7…….36
Table 9: Comparing the percentage error with the distance from the camera…….…...37

# LIST OF FIGURES

Page

Figure 1: The fundamental diagram of traffic flow…… 13
Figure 2: Showing the surround view system of a vehicle…… 16
Figure 3: HSV value shown on the cylindrical plane…… 17
Figure 4: The first step of the centroid algorithm…….. 27
Figure 5: The algorithm calculates the Euclidian distance between every centroid……. 27
Figure 6: Pairs with Euclidian distances less than the threshold are assumed to be the same object……. 28
Figure 7: Object with identifier #3 is a newly assigned object……. 30
Figure 8: Line chart comparing the expected and measured distance across the tests……. 32
Figure 9: Line chart comparing the expected and measured distance across the tests……. 33
Figure 10: Line chart comparing the expected and measured distance across the tests…… 35
Figure 11: Line chart comparing the expected and measured distance across the tests…… 36
Figure 12: Bar chart showing the average percentage error for the tests at different distances…. 37
Figure 13: Green box on the left tracked as car 1 and green box on the right tracked as car 3…. 41
Figure 14: Green box on the right showing a tracking number of 1 and green box on the left shows a tracking number of 2 ……. 42
Figure 15: The camera is stationary, but the boxes are moving towards the camera……. 43
Figure 16: The green boxes were stationary and the camera moved towards them…….. 44
Figure 17: The green boxes were stationary and the camera moved towards them…….. 45
Figure 18: The camera and boxes are both moving in opposite directions…… 46
Figure 19: The camera and boxes are both moving in opposite directions…… 46

# CHAPTER 1

# INTRODUCTION

## 1.1 Motivation

According to the National Highway Traffic Safety Administration (NHTSA, 2017) there are 289.5 million registered cars on the road in the United States with that number going up about 1.5% every year (NHTSA, 2021). The increase of vehicles results in an increase of vehicle miles travelled by 11.2% in 2021 as compared to 2020, and results in an increase of traffic related fatalities by 10.5% in 2021 as compared to 2020. (NHTSA, 2021)

In recent years, auto manufacturers, under the premise of safer vehicles, have been outfitting vehicles with more and more sensors and systems designed to make the driver more aware of his or her surroundings and, as a result, safer. The first generation of systems would first detect a safety incident and notify the driver; however, later generations will actually act on behalf of the driver. An example of this is Forward Collision Warning, also Forward Collision Alert, that would notify drivers of an impending collision using an audible tone. As the technology has become perfected, the next generation now applies the brake automatically on behalf of the driver. This Automated Emergency Braking is now standard, in the United States, for most new vehicles and will be mandatory as of 2025 (NHTSA, 2016)

As vehicles are being outfitted with more and more sensors, vehicles also need more computing power to process data received from these sensors. Additionally, vehicles are being equipped with Connected Vehicle technology to enable vehicles to communicate automatically with other vehicles and infrastructure. In fact, a NHTSA study has found that Connected Vehicle technology can potentially reduce up to 80% of vehicle accidents that are not the result of impaired drivers (NHTSA, 2017). In addition to safety, with the increased awareness of its surroundings through sensors, the increased computing power, and the ability to communicate with other vehicles in infrastructure has led researchers to suggest several applications.

Of particular interest to the author is the ability to use the sensing, computing, and communication capabilities of vehicles to capture the flow or density of vehicles on the road. In fact, it is fundamentally important for traffic engineers to estimate different traffic parameters: namely, flow, density, and velocity. Historically, traffic engineers have favored, because of their simplicity, Stationary Observer (SO) methods for these estimations. The SO methods involve placing stationary observers by the roadside to count vehicles passing in a given time period. Using this information, flow, density, and average velocity can be inferred. The most user SO method uses inductive loop detectors; however, other methods include using pneumatic tubes, camera, radar, and other types of sensors.

Other methods include the Moving Observer (MO) method which employs vehicles on the roadway to count vehicles in traffic. After several readings, the counts of vehicles are aggregated to estimate traffic flow, density, and average velocity. The method was originally introduced in 1954 by Wardrop and Charlesworth (Wardrop and Charlesworth, 1954); however, has not been used much by traffic engineers because of its difficult to deploy. In modern times, the MO method has gained interest by Florin and Olariu (Florin and Olariu, 2017); they have proposed three variants of the MO method using the sensing, computing, and communication capabilities of today's vehicles. In their works Florin and Olariu propose and prove theoretical MO methods and they suggest how they can be implemented; however, they do not perform any applied studies of their methods. In this work, I carry out experiments to test the mobile observer methods with the end goal of being able to estimate fundamental traffic parameters. The different mobile observer methods will be explained in the next section.

## 1.2 Technical Background

### 1.2.1 Wireless Network Research

Starting in the 1970s, ALOHA net and PR Net (sponsored by DARPA) were the precursors to today's wireless networks. From PR Net came Mobile Ad Hoc Networks (MANETs) and Wireless Sensor Network Research. From MANETs came Vehicular Ad Hoc Networks (VANETs), which is just the

application of VANET research to vehicles. From VANETs came Vehicular Clouds (VC) which was first coined by Olariu in 2009 (Olariu and Weigle, 2009). Vehicle to Vehicle Communications (V2V) is an updated name for VANET which involves vehicles communication to other vehicles. Vehicle to Infrastructure (V2I) is the name for vehicles communicating to roadside base stations called Road Side Units (RSUs). RSUs are typically connected with fiber optic cables and also connected to the internet. RSUs may be used to provide vehicles with access to the internet. Vehicle to Everything (V2X) is the collective name for vehicles communicating to other vehicles, infrastructure, pedestrians, etc. Dedicated Short Range Communications (DSRC) was allocated part of the electromagnetic spectrum by the FCC in 1999. In 2020, due to failure to implement, the FCC reallocated the spectrum to the ISM band and to Cellular V2X (C-V2X) Federal Communications Commission, "In the matter use of the 5.850-5.925 GHz band.". C-V2X is a competitor to DSRC that uses LTE technologies to implement V2X.Edge Computing can be applied to Vehicle Networks by providing cloud resources at the RSU. This is called Edge Computing because the processing is brought to the "edge" of the network. One application of Vehicular Clouds is Vehicular Edge Computing where the vehicles themselves are the edge resources. The processing power and sensing capabilities of vehicles are used as a cloud resource.

### 1.2.2 V2X

Vehicle to Everything (V2X) is a communication system that allows a vehicle to communicate with other vehicles or infrastructure. Its main purpose is to share safety messages and warnings but can also be used for entertainment-based messages or just generally sharing information between vehicles. Currently, V2X technology is still in the early adoption phase; however, once fully embraced it will provide a connection to the ITS infrastructure, including traffic lights, parking spaces, etc. With this ubiquitous availability, the benefits would be numerous but most notably will be increased safety, fewer crashes, traffic efficiency, and less road congestion.

V2X has two main parts: Vehicle to Vehicle communication (V2V) and Vehicle to Infrastructure communication (V2I). As the names imply, V2V handles communication between vehicles while V2I

handles communication with outside entities such as RSUs, traffic lights, pedestrians, etc. The two main technologies used in V2X are DSRC which is based on radio waves and uses the same protocol as Wi-Fi, and C-V2X which uses cellular communications which is what telephones use to communicate with each other. With the introduction of 5G and even technologies stronger than 5G in the future, C-V2X will be more optimal because of its speed and reliability.

### 1.2.3 Vehicular Ad hoc Networks

An ad-hoc network is a network that is formed when two or more devices in the same geographical area connect to one another in an ad hoc manner. The term "Ad hoc" is of Latin origin and means spontaneous or impromptu. An ad hoc network does not need any infrastructure to form such as a base station or an access point. Ad hoc networks can access the internet as long as there is a node in the network that can access the internet. All the other nodes connect through this node.

A Vehicular Ad hoc Network (VANET) is formed when two or more vehicles connect, forming an ad-hoc network, enabling them to share important information like road safety, congestion, obstacles on the road, etc. In addition to communicating with other vehicles on the road, VANETs enable cars to communicate with smart infrastructure on the road such as Roadside Units. With VANETs, vehicles can access the internet and get important safety and weather information which is factored in by the car's processor in conjunction with onboard sensors in the car to make decisions about the driving path of the car. All enabled vehicles can join a localized VANET. This ad hoc network has been in development since the early 2000s and they have wide-ranging applications in today's intelligent and autonomous vehicles. With the presence and application of VANETs that is where the popular term "Connected Vehicles (CV)" comes from.

### 1.2.4 Connected Vehicles

A connected vehicle is a vehicle that is able to communicate with other vehicles, phones, laptops, RSUs, and other wireless devices around it. It is an Internet of Things (IoT) component that could have vast

implications in the way we drive, especially concerning safety. It is an initiative that was started by the US Department of Transportation (USDOT). It was initially called Vehicle Infrastructure Integration, then Intellidrive, before its present name Connected Vehicles. According to the National Highway Traffic Safety Administration, there are 5 million car crashes in the United States every year and there are about 30,000 deaths (NHTSA, 2021). Connected Vehicles technology could significantly reduce this number by allowing drivers to receive alerts of an obstacle or accident on the road.

Using onboard sensors and incident detection systems, vehicles can alert the driver or act on behalf of the driver. With CV technology, the vehicle then also alerts other vehicles of the same incident giving vehicles and drivers more time to react.

### 1.2.5 Traffic Parameters

Traffic engineering is a branch of engineering that uses various techniques to plan and manage the safe, economical, and efficient movement of vehicles on the road. To understand traffic behavior, it is important to understand the various traffic parameters. For the purpose of this paper, we are going to focus on the main three: velocity, flow, and density.

Velocity, *v*, is the rate at which a car travels over a distance, *d*, per unit time, *t*. When looking at a roadway, the velocity of the cars is going to be different. To get a measurement of the velocity of a section of the road where we are calculating the traffic parameters, we use the space mean speed also known as the average speed, which can be calculated by averaging all the velocities of the vehicles in our section of the road

$$v = \frac{d}{t}$$

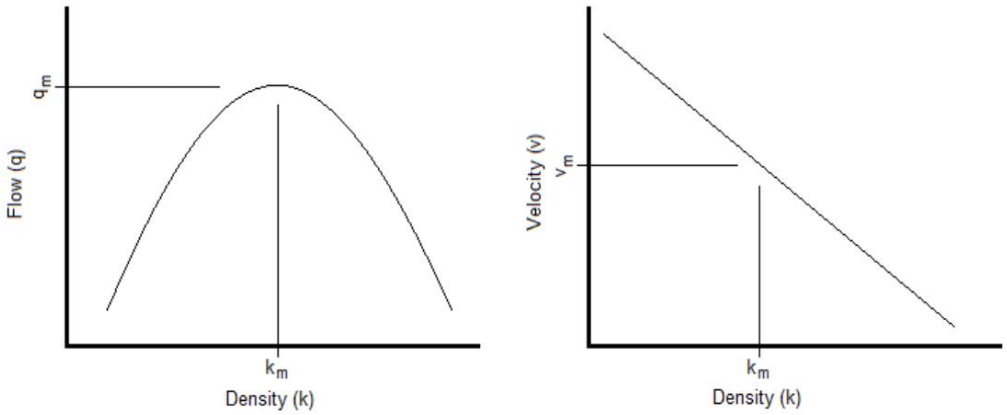
Flow is one of the most significant traffic parameters. Flow, *q*, is the rate at which vehicles pass a particular point on the road, that is, the number of vehicles, *n*, that pass this point per unit time, *t*. This point could be another vehicle, an RSU, or any other thing. The flow is calculated over an interval of time.

$$q = \frac{n}{t}$$

Density refers to the number of vehicles on a given roadway or section of the roadway. The formula for calculating the density, *k*, of a roadway is the number of vehicles, *n*, occupying that roadway divided by the length of the roadway, *x*. A smaller density would indicate that there is more space between cars and a larger density would indicate the cars are close together.

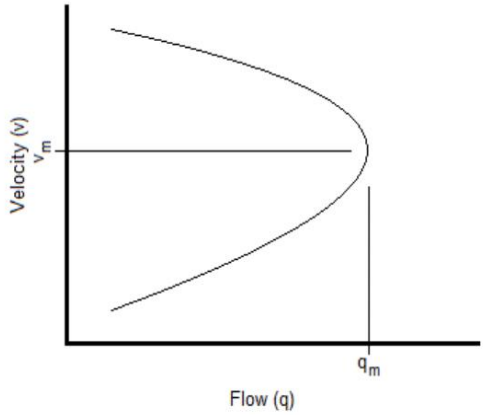
$$k = \frac{n}{x}$$

$$flow, q = \frac{number\ of\ vehciles}{time\ taken}$$




(a) The flow - density relationship

(b) The velocity - density relationship




(c) The velocity - flow relationship

Figure 1: The fundamental diagram of traffic flow

The equations for flow, density and velocity are called the fundamental equations of traffic estimation. These equations are very important in the traffic engineering field, the relationships between them is described in Figure 1 above. Figure 1(a) shows the relationship between flow and density. The relationship is similar to the inverse of a parabolic function. As flow increases to its maximum – the max flow – then density also increase to its maximum. Once the maximum density is reached, flow will start to decrease. This self-evident by looking at a roadway, as the number of cars on the road increases, the density, by definition, increases. When there are many cars on the road, cars will slow down and might even come to a complete stop as the density continues to increase. Figure 1(b) describes the relationship between

velocity and density of cars on the road. The relationship between velocity and density is linear. As density increases, velocity decreases. This is because the as the number of cars on the road increase, the road will be congested thus cars will have to slow down. The extreme of this will be when density is at its peak and cars will be at a standstill. Figure 1(c) shows the relationship between velocity and the flow of traffic on the road. This relationship is also an inverse parabolic relationship. As the velocity of vehicles on the road increase towards the maximum, the flow of vehicles on the road increases. But, as the vehicles increase past the maximum velocity, the flow would start to decrease because the road will start becoming congested and cars will have to slow down.

### 1.2.6 DSRC

DSRC stands for Short Range Communications for Wireless Access in Vehicular Environments. This can also be called WAVE and is based on IEEE 802.11p. DSRC is a wireless communication technology that allows cars and other intelligent vehicles on the road to communicate with one another and with other infrastructure on the road. DSRC operates on a 5.9 GHz band on the radio frequency spectrum and is most effective over shorter distances. (FCC, 2022)

### 1.2.7 SURROUND VIEW SYSTEM

A surround view system is an Advanced Driver Assistance System (ADAS) that shows the driver of the vehicle a 360-degree view of its surroundings so they driver can make better and more informed decisions when driving on the road, parking a vehicle, or reversing the vehicle. The surround view system is also sometimes called the bird's eye view. This system is made of four main components: 4 – 6 wide lens cameras, proximity sensors, image processing software, and the hardware to display the 360-degree image. The wide lens cameras are placed on all sides of the vehicle so that they cover 360 degrees and then the images captured by the cameras processed and put all into one plane. The image stitching process has three steps: image registration, warping, and blending. In the image registration phase, key points on the images

are identified and assigned to a custom ground plane. Custom pattern registration is the image process commonly used in this step. In the warping phase, the processor warps the images together to form a 360-degree image and cuts out all distortions on the image. Then in the blending phase, similar parts of different images are blended, and all images are merged to make one surrounding view image.

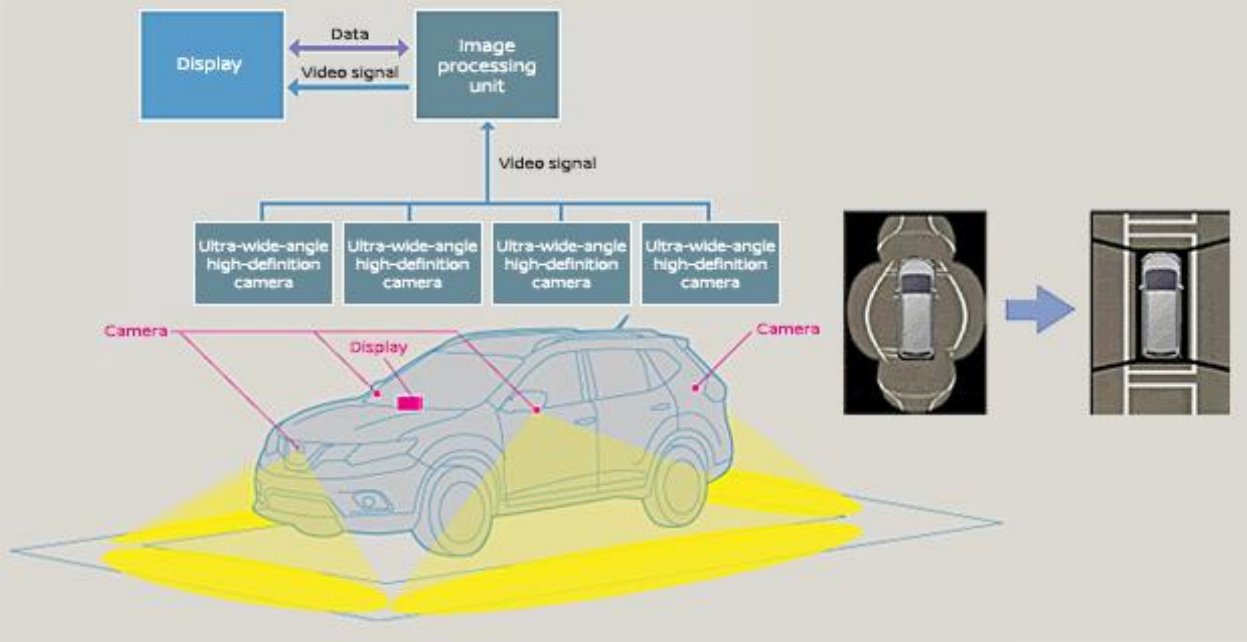



Figure 11: Showing the surround view system of a vehicle. (Photo source: *Intelligent around view monitor: Innovation*)

1.2.8 HSV

HSV stands for Hue-Saturation-Value. HSV is a color scale that was developed by computer graphics researchers in the 1970s as an alternative to the RGB model to try and represent colors on a computer closer to the way humans perceive color. HSV is generally preferred in image processing applications over the RGB model. When writing the programs to conducts experiments as part of this paper,

the HSV model was used. The HSV model is on a cylindrical coordinate plane. Hue represents the primary colors (red, blue, and green) and the secondary colors (orange, green, and violet). These are all pure colors which can be seen on a rainbow. Hue is a range that goes from 0 to 360 as follows: 0 = red, 60 = yellow, 120 = green, 180 = cyan, 40 = blue, 300 = magenta. Saturation in HSV means how intense an image is. The more vivid an image is the higher the saturation number will be and the less vivid an image is the lower the saturation number. The lower the saturation the more the image is to pure gray on the grayscale. The saturation range goes from 0 to 100 percent. The value in HSV indicates the brightness of the image. Human beings perceive color value based on how much light falls on the image, the more amount of light on the image, the higher its value. The value range goes from 0 to 100 with a 0 being pure black.

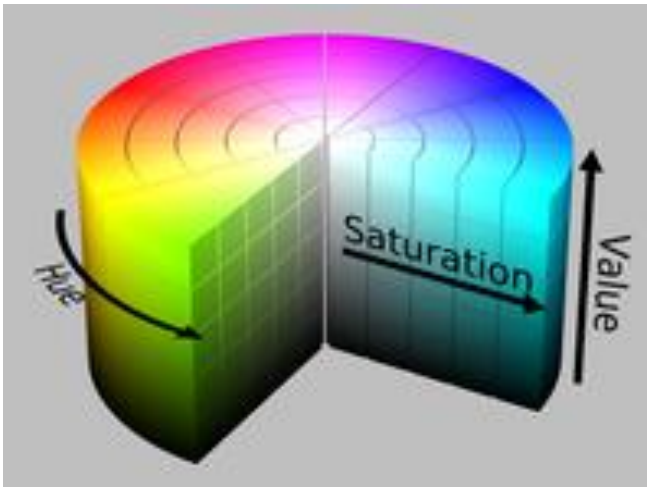



Figure 12: HSV value shown on the cylindrical plane. (*File:HSV cylinder.png - wikimedia commons*)

### 1.2.9 MOBILE OBSERVER METHODS

The mobile observer method is a technique used in traffic engineering to gather data on traffic flow and patterns by observing vehicles from a mobile platform, such as a car or a bicycle. This method involves a trained observer who records various parameters, such as traffic volume, vehicle speeds, lane occupancy, and other characteristics, while traveling along a designated route. In 1954 Wardrop and Charlesworth proposed the moving observer method to measure fundamental traffic parameters based on the number of

vehicles observed passing. A modernized version of Wardrop and Charlesworth moving observer method is Mobile observer method 1 (MO1) which utilizes the capabilities of modern vehicles to address some of the limitations associated with the moving observer method. Mobile Observer method 2 (MO2) utilizes two vehicles travelling in the same direction to determine the density between the vehicles. Mobile Observer method 3 (MO3) is similar to MO2 except that here, the vehicles are travelling in opposite directions, so their relative velocity is high. This difference makes MO3 quite suitable for high traffic urban areas.

## 1.3 Research Question

In this introduction, we have discussed connected vehicles, traffic parameters namely flow, density and velocity, how vehicles communicate with each other – C-V2X/DSRC, machine learning models used for tracking cars' speed and distances on the road and estimating the distance and angle between vehicles using mathematical formulas. These topics are all pertinent for the purpose of this paper, which is to answer this question: Can a group of vehicles use their shared resources to accurately estimate the number of cars in their vicinity even the ones that are not visible to some of the cars in the group through occlusion. Furthermore, can this group of vehicles share their views and create a surround-view-like image of their surroundings and the cars around them? With having this information accurately computed will this make estimating traffic parameters more accurate to the vehicles in the group? Can this system be used as a practical sensor for the Mobile Observer Method? The goal of this work is to answer the research questions.

# CHAPTER 2

# RELATED WORKS

## 2.1 DISTANCE ESTIMATION

Rahul and Nair (Rahul and Nair, 2018) proposed a camera-based distance estimation system that uses a Convolutional Neural Network (CNN) and geometry to determine the distance an object is from the camera. The CNN is used to detect and identify the objects. When the object is identified, their distance is estimated by constructing a 3D point cloud using a triangulation method. Rahul and Nair (Rahul and Nair, 2018) method turned out to be 84% effective at estimating the distance of objects up to five meters (16.4 feet) away. Rahul and Nair's system used an Nvidia Jetson TX1 and a Zed video camera. In this paper, the authors tested different neural networks to ascertain which was the best for the goal they were trying to accomplish. Traditional CNNs are too large for the smaller processing devices so it was necessary to use something more compact. MobileNet SSD was deployed for the detection and identification of objects in this study. SSD (Single Shot Detection) performs the region proposition and object classification into one stage and this reduces the computing power and time needed. The neural network was trained with preprocessed data and had a high accuracy of detecting objects from the live video feed. To calculate the distance of an object from the Zed camera, Rahul and Nair used the follow below which divides the product of the focal length, f, the base line, e, with the product of width of Zed camera pixel, $P_x$, and the disparity for point P, ð.

$$Z_p = \frac{f \times e}{p_x \times \delta}$$

To test the accuracy of their system, the authors measured the distance of objects with increasing lengths from the camera of 1000 centimeters. The highest percentage error that occurred was at 2000 centimeters away with a percentage error of 4.64%. The overall percentage error of the experiment was 2.65%.

Rahul and Nair's methodology produced a high level of accuracy. In the system proposed in this paper, we differ from Rahul and Nair by using a tracking algorithm instead of a CNN to be able to identify and track objects and we base our formula for tracking the distance of objects on the triangle similarity theorem. This is where we look at the ratio of the number of pixels to how far or near an object is. Rahul and Nair's method is similar to this, but the method used in this paper has different parameters and calibrations which will be discussed more in chapter 3.

Liang *et al* proposed a system for estimating the distance of objects from a camera by using a network-based on ShuffleNet and YOLO-v5 to detect the object and using a supervised learning network to estimate the distance (Liang et al., 2022). This is a different type of approach because most researchers have used a neural network for detecting the object and mostly used geometry to determine the distance of the object from the camera. This method proposed by Liang *et al.* a variant of YOLO called Light-Fats YOLO (Liang et al., 2022). They trained the neural network with a premade COCO dataset to be able to detect distances of people from the camera. It is important to note that their detection and distance estimation was not happening in real time from a live video feed, but it was taking pictures trying to detect how far objects are after. Liang *et al.* model achieved very high accuracy when tested over other datasets.

The methodology used in Liang et al's paper differs from the methodology used in this paper because they did not use a live video feed. They first took the picture and performed the distance estimation using their trained deep learning model. In this paper, since we were performing the estimation in real time, we employed mathematical theorem's (discussed in Chapter 3) to be able to estimate the distance from the camera. The results from our experiments will be discussed in Chapter 4 of this paper.

## 2.2 MOBILE OBSERVER METHODS

After the stationary observer method then came the moving observer method which is when an observer in a moving vehicle counts the number of oncoming vehicles passed over a specified length of road to determine the flow rate. This method was developed by Wardrop and Charlesworth (Wardrop and Charlesworth, 1954). In this method, the observers in the test vehicles record: the number of opposing vehicles met, the number of vehicles overtaking the test vehicles while it's travelling, the number of vehicles the test vehicles overtook, and the average speed of the test vehicle and the distance covered. Mulligan and Nicholson tested the moving observer method by simulating rural roads in New Zealand, many of which had low flow rates (Mulligan and Nicholson, 2002). Mulligan and Nicholson simulated these roads by using a Risk analysis method. The Risk analysis method works in a spreadsheet in excel and it has the following steps: a model is developed by defining a problem and its associated variables, the uncertainty in the variables identified and a suitable probability distribution is assigned, then the model is analyzed by the simulation to determine the range and probability of all outcomes. In their simulation, the three control variables are actual flow rate, the section length and the observer speed. There were a total of 36 combinations or scenarios analyzed in their simulation (Mulligan and Nicholson, 2002). From analyzing the results obtained Mulligan and Nicholson determined that the moving observer method was a sound method for estimating traffic flow rates. Furthermore, they observed that the method is better suited for high volume urban roads and less suited low flow rural roads to achieve a high level of accuracy.

Langer et al. simulated a variant of the moving observer method on bidirectional one-lane roads, but they believe that their method can be replicated on multi lane roads (Langer et al., 2020). Their method focused on observing vehicles stopped by traffic lights at intersections. The data they tracked are the observer vehicle's speeds, travelling times, waiting times at intersections and the number of required stops. To calculate the flow rate of the intersection, the division of the number of vehicles passing the intersection during the green phase of the traffic light and the cycle time of the traffic light is calculated. To test this

method, Langer et al. simulated their data on SUMO. The simulation utilized one urban road with one lane in each direction and one signalized intersection at the middle. Their simulations used varying number of observers: 2,3 and 4. Their average relative error was 22.4% which was better than Mulligan and Nicholson's average relative error of 33.8% (Mulligan and Nicholson, 2002).

Redmill et al. proposed a system and prototype for detecting vehicles over multiple lenses and providing velocity and shape-based vehicle classification for each vehicle (Redmill et al., 2011). This system will be done on the moving observer method platform and will use a range of sensors such as GPS, IMU, and OEM. In this system, there will not be a few cars on the road as data collection vehicles but the sensors and will be placed on transit and municipal vehicles such as buses, police cars, taxis, etc. These are cars that are taking multiple routes and are on the road for long periods of time, so they make suitable vehicles to gather large amounts of data from. To run this experiment, Redmill et al. attached sensors and data collection equipment to a 2000 Hyundai Odyssey minivan. The vehicle was equipped with a GPS, three lidar sensors, a long-range radar sensor, a gyroscope and three cameras. The data collected was stored on a Linux system connected to the sensors inside the car. The vehicle was able to drive on a pre-planned route and gather information about the type of vehicles on the road and the distance of the vehicles from the Hyundai Odyssey.

Lin et al. proposed a system where data is gathered from the cameras in vehicles to calculate traffic parameters on the road (Lin et al., 2019). Lin et al. mentioned how most of the traffic estimation is done with cameras placed on Roadside Units (RSU) or street cameras, but this leads to inefficiency, inaccuracy and limited data gathering because RSUs are not widely available on the road. But with the cameras in vehicles and harnessing the power of edge computing and with vehicles communicating with each other using V2X (vehicles to everything) technology, traffic parameters can be calculated more efficiently and with greater accuracy. The system proposed is called In-vehicle Camera based Traffic Estimation scheme (ICTE) and it also uses a YOLO-based vehicle detection and a Hough-based Lane detection system (Lin et al., 2019).

Florin and Olariu (Florin and Olariu, 2023) proposed a way to implement a real-time traffic density estimation method that uses vehicle-to-vehicle communication and the sensors in the vehicles. In their method, vehicles will keep tally of the difference between the number of cars it passes and the number of cars that pass it. One of the unique things about this method is that the vehicles do not have to be at constant velocity therefore it simulates real conditions on the road and allows for a vehicle to be passed multiple times. Florin and Olariu also proposed a new methodology in which vehicles were travelling in the opposite direction. What this methodology provides is that even with very little to no passes, other vehicles in on-coming traffic will still be able to observe traffic. In addition to that, since the vehicles are travelling towards each other, it will take less time and less distance for the vehicles to reach other. This greatly reduces the area needed for the conservation of vehicles and thereby makes this methodology a lot more efficient. The efficiency and accuracy of the proposed methodology was tested extensively using SUMO. The simulation and sensitivity analyses showed this methodology to be robust and highly accurate.

Florin and Olariu (Florin and Olariu, 2020) proposed a variant of the Mobile Observer method that utilizes the communication capabilities present in modern vehicles to mitigate issues that were problematic in the original Mobile Observer methods. One of the major contributions of this paper was proposing a protocol for traffic density estimation which uses exclusively V2V communications coupled with the on-board sensing and computing capabilities of modern vehicles. One thing that sets this paper apart is that most methods proposed for estimating traffic density require the aggregation of probe vehicle data, this method is self-sufficient and does not require any such aggregations that are not pertinent. Furthermore, as opposed to older methodologies which require constant vehicle flow at uniform velocities, this method simulates real traffic flow with velocities and densities that are not constant and uniform.

# CHAPTER 3

# METHODOLOGY

## 3.1 DETERMING THE ANGLE AND DISTANCE OF CARS

In the experiment to test the MO methods, it was necessary to be able to track the distance and angle of the surrounding cars to know their positions at every instance so it would be easier to tell if a car had passed the camera or the camera has passed a car. It was necessary to track the distance because that would indicate how far or close cars are and this could help with predicting when the camera would pass a car or a car would be passed by the camera. It was necessary to track the angle to determine on which side of the road a car was going – either to the left of the camera or to the right of the camera or in the opposite direction of which the camera was travelling. There are many different methods to track the distance of an object from a camera because this is an important problem in the computer vision/ image processing space. There are very complex and intricate (and more accurate) methods but there are more straightforward methods. One of which is the triangle similarity method which was employed in this experiment.

The triangle similarity method for determining the distance of an object from the camera is based on a geometric theorem that two triangles are similar if the only difference between them is size. For example, if the length of the sides of tringle A are 5, 5, and 7 and the length of the sides of triangle B are 15, 15, and 21, these are similar triangles because the only difference between them is that triangle B is three times larger but all ratios still remain the same.

Applying this concept to problem of finding the distance of an object from the camera, if an object with a known width, W is placed at a known distance, D, a picture can be taken to determine the number of pixels occupied by this object at that certain distance, D. This gives the perceived of focal length at distance D when applying the formula:

$$F = \frac{P \times D}{W}$$

Where F is the focal length, D is the distance, P is the number of pixels taken up by the image, and W is the width of the object. Now once this calibration step is done, the triangle similarity is applied because if the object was a distance D when the number of pixels was P, then the distance would become D* following the ratio of the known focal length, F and the new number of pixels P1. So, the formula to continuously track the distance becomes:

$$D^* = \frac{F \times W}{P^*}$$

For example, to measure an object with a known width of six inches the following steps will be taken. The object will be placed at known distance, in this case twelve inches. When the picture is taken, the number of pixels the object occupies on the picture is noted and, in this case, it is 470 pixels. So, the focal length at twelve inches is 235. Now this will run on a loop and continuously measuring the number of pixels occupied by the object on the image and using equation 2 to calculate the distance.

In the experiment, the distance of objects was tracked in real time from a video feed from a computer's webcam. The python library OpenCV was used to create a bounding box around the object and continuously calculate the number of pixels occupied by the objects at any given time.

There are many ways to calculate the angle of an object relative to ninety degrees from the camera angle. For this experiment, a simple but effective method was employed which is to use ratios. To do this, the first step is finding out how wide of angle the camera can capture. This was done manually by placing the camera and drawing lines to both ends of the frame of the image captured by the camera and measuring the angle at where these lines meet. For the camera used in this experiment, the angle was 110 degrees. So there was a range from -55 degrees to 55 degrees, any object tracked with an angle above 0 degrees was to the right of the camera and vice versa. The next step is finding out the furthest positions the camera can

capture. That is, what is the furthest position that we can move the object to the left or right of the camera and it would still be in the picture frame. This was done by using the python OpenCV library. The OpenCV library provides a function that places a bounding box around the object when its color is specified. This function returns the dimensions of the bounding box and the 'x' and 'y' position of the object. Using the 'x' position, the midpoint of the object was tracked by continuously calculating the sum of the x-position and the width of the bounding box divided by two. The midpoint of the bounding box was tracked to the furthest points on the left and right side of the frame, and it was determined that the distance between them is 1736. Therefore, the ratio to obtain the angle is 110 / 1736. To make sure the angle falls in the correct range 55 is subtracted from the product of the position and the ratio. The formula for the angle calculation is shown below.

$$angle = \left(x \times \frac{110}{1736}\right) - 55$$

It is important to note that the numbers 110 and 1736 are dependent on the camera. So, calibration would need to be done on each new camera to the program is running on to make sure the values are accurate.

## 3. 2 TRACKING AND COUTING CARS

One of the most important parts of the experiment is being able to count the number of cars that the camera passes and the number of cars that pass the camera. In a real-world setting, this would be how many cars the main car overtook and how many cars overtook the main car. To simulate this, objects would need to be tracked to determine how many were passed and how many passed the camera. There are many ways to track objects, one of the most common is using a machine learning model. This was initially tried by training a model with pictures of a green square (representation of a car in the simulation) and using the You Only Look Once (YOLO) model to track the object. This worked at first but was inefficient because of the amount of time was spent training and would be needed to retrain the model in the case of using

another color or different object to represent the cars. Trying not to reinvent the wheel, some popular tracking algorithms that are also good for counting the number of objects were tested to see which gave the best results. The tracking algorithms used were centroid tracking, IOU tracking, and SORT tracking. Before looking how each of these tracking algorithms work, there are some important terms to understand. The first term is Intersection over Union (IOU). IOU is a very common evaluation metric used in machine learning, image processing and computer vision to determine how accurately placed the predicted bounding box is. This metric is one of the ways an object detector's accuracy is measured. IOU is also referred to as Jaccard Index or Jaccard similarity coefficient. To calculate the IOU, two parameters are needed: the ground-truth bounding box and the predicted bounding box. The ground-truth bounding box is the manually placed bounding box put on images in the testing set while the predicted bounding box is the bounding produced from object detector or ML model. The formula for calculating IOU is shown below:

$$IOU = \frac{Area\ of\ Intersection/Overlap}{Area\ of\ Union}$$

The area of intersection is area contained within the convergence of the ground truth bounding box and the predicted bounding box obtained from the output of the object detector. The area of union is the area contained in both the predicted bounding box and the ground-truth bounding box. Another way the formula can be written is using the mathematical formula of the Jaccard index as shown below:

$$J(A,B) = \frac{|A\ \cap B|}{|A\ \cup B|}$$

The highest possible value for the IOU is 1 which is obtained when the ground-truth bounding box and the predicted bounding box are exactly in the same place. However, an IOU of 1 is very difficult to achieve. There is not an established range of which IOU is excellent and which is poor because these ranges might vary depending on the project being worked one. For the experiment in this paper, an IOU greater than or equal 0.75 was considered good and an IOU greater than 0.90 was considered excellent.

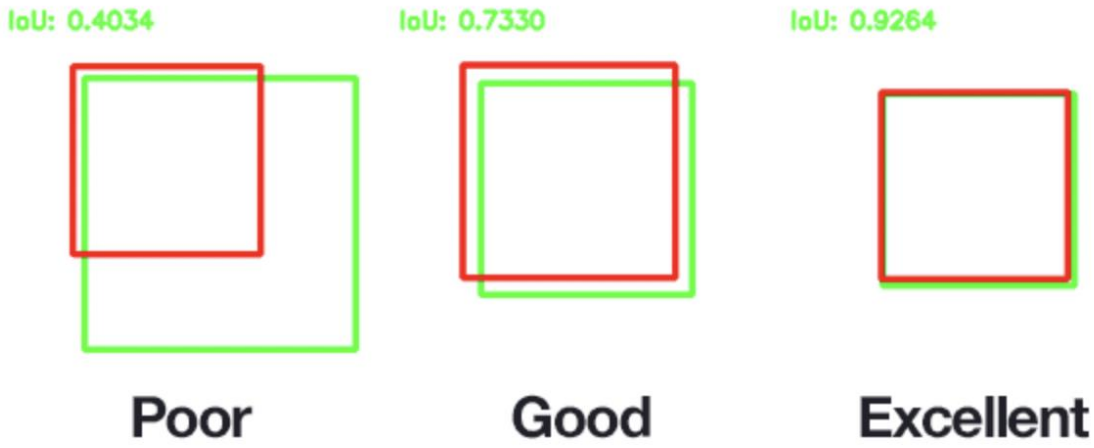



Figure 1: Showing different IOU scores and the visual representation of bounding box overlaps.

As a type of tracker, the IOU tracker only depends on the tracking information and the training image set so it does not use any image information and motion model. Because of this, the IOU tracker can operate at higher frame rates and can outperform, in terms of efficiency, much more complex algorithms. However, because the high frame rate this algorithm has a requirement for a high recall of the underlying detector. This is because for every frame, the detector has to be called to scan it and if the frame rate is very high the program will not be able to keep up. Because of the high frame rate, the gaps caused by missed frames can lead to wrong detections or false negatives. Missed frames are frames of the video which were unable to be scanned by the detector. The IOU threshold is an input parameter in the IOU tracker that determines whether a detection has occurred or not. For example, for an IOU threshold of 0.75, if the detector calculates an IOU score of 0.75 or above that would be counted as a detection and anything lower than 0.75 would not be counted.

Another one of the tracking algorithms tested was the SORT tracking. SORT stands for Simple Online Realtime Tracking. SORT tracking is an approach to object tracking that uses a Kalman filter motion model and handles the detection assignment problem using the Hungarian algorithm (Cohen, 2021). Kalman filtering is simply the method of providing the estimated location of some variable given the measurement observed over time. It consists of two stages: prediction and update also called propagation

and correction. The filter takes stored data and, based on the measurements available, it makes a prediction where an object will be at a specific time. The Hungarian algorithm is used for determining whether an object in the current frame is the same as one in the previous frame. The Kalman filtering method and Hungarian algorithm working together is how the SORT algorithm works and because tracking from frame to frame of the Hungarian algorithm and predictions from Kalman, the SORT algorithm is very good for problems involving occlusion – both partial and full occlusion. The SORT algorithm has four key components: detection, estimation, data association, and creation and deletion of track identities. Detection is the first step in tracking. This is when an object detector is employed to detect the required objects in the frame to be tracked. Any type of object detector can be used here, some of the most common are YOLO, SSD, and R-CNN object detectors. In the estimation step, the detection from the current frame is transferred to the next frame where it tries to estimate the position of the target in that frame. If the detection achieved is associated with the target then the bounding box is updated. In the data association step, there are now two bounding boxes, the target bounding box (also called ground-truth) and the detected bounding box. Using the positions of these two bounding boxes, the IOU is calculated and if the calculated IOU is less than the minimum IOU threshold then that detection is rejected. Unique identifiers are created and destroyed depending on the minimum IOU threshold. When an object has been tracked and given a unique identifier, if this object is not tracked in subsequent frames this track will be deleted. The number of frames where an object is not seen would be specified by the user of the algorithm. After the unique identifier for an object has been deleted, should the object appear again it will be assigned a new unique identifier.

The final tracking method used in this experiment was the centroid tracking method. The centroid tracking method is based on calculating the Euclidian distances between all objects in one frame and the next frame from a video feed. If this distance is more than a set threshold, then a new identifier is assigned to that object. That is, if the Euclidian distance is less than the threshold then it is the same object in motion. The premise of this being that is expected that even if the object is moving quickly, the distance moved

between one frame and the next would not be much. The formula for calculating Euclidian distance is shown below:

$$Distance = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Why the centroid tracking method was very good for the experiment done in this paper is because it had the highest level of accuracy between the different tracking algorithms that were tested. Detection is the ability to know that the object is in the frame but tracking the ability to monitor the same object from frame to frame by assigning a unique identifier and still maintaining this identifier as objects move around. This algorithm is very good tracking multiple objects that look identical to each other because of Euclidian distance calculation it employs.

The first step of the centroid tracking algorithm is to calculate the centroid of all the objects in the frame. It does this by using the coordinates of the bounding boxes obtained from which every object detection model the tracker is using. A centroid is the center of an object. The bounding boxes produced from the object detection model will be rectangles. To calculate the centroid (middle point) of a rectangle, the two diagonals of the rectangle are drawn and the point where they intersect is the centroid. The bounding box obtained from the object detector model returns four values, the *x* and *y* coordinates of the top left of the box and height and width of the box. To calculate the centroid the two formulas shown below are used:

$$centroid_x = \frac{x + width}{2}$$

$$centroid_y = \frac{y + length}{2}$$

Length and width are the dimensions of the bounding box placed around the target.

Once the centroids are calculated, they are also assigned a unique identifier

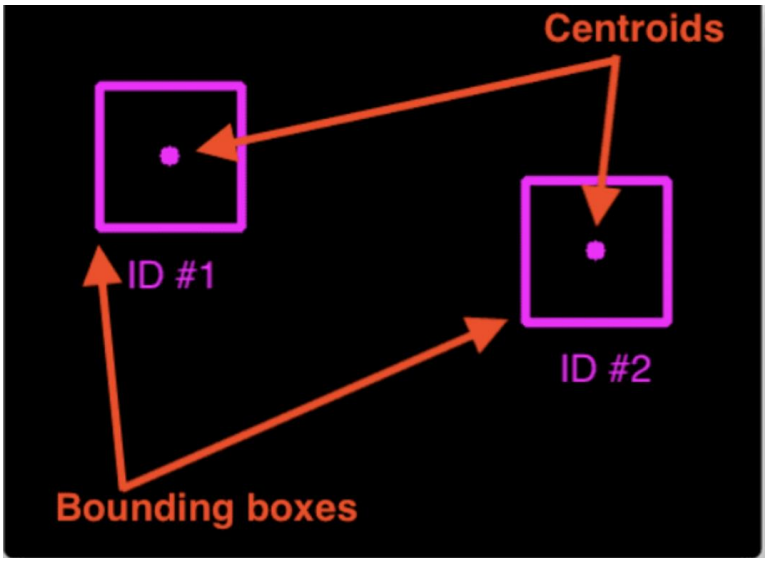



Figure 2: The first step of the centroid algorithm.

The next step of the centroid algorithm is to continue to calculate the centroid of each object on the screen from frame to frame and also calculate the Euclidian distance between each and every centroid from one frame to another.

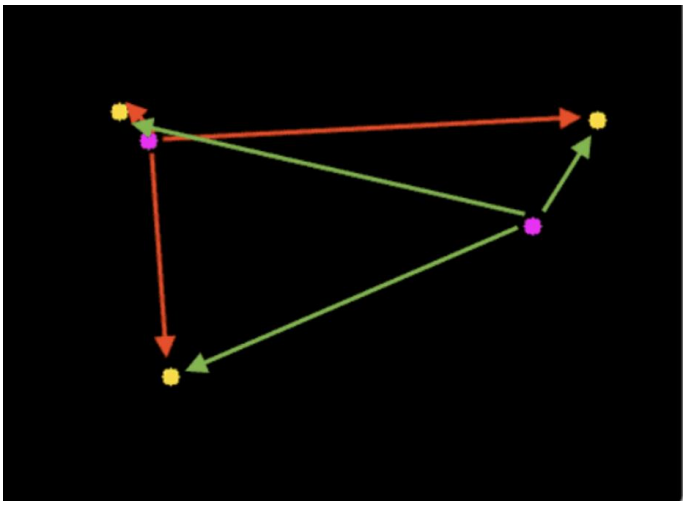


Figure 3: The algorithm calculates the Euclidian distance between every centroid.

The third step in the centroid algorithm is comparing the Euclidian distance to the threshold. It is assumed that objects will move from frame to frame but they will move less than the threshold. If the Euclidian distance between two pairs of centroids is less than the threshold then they are assumed to be the same object.

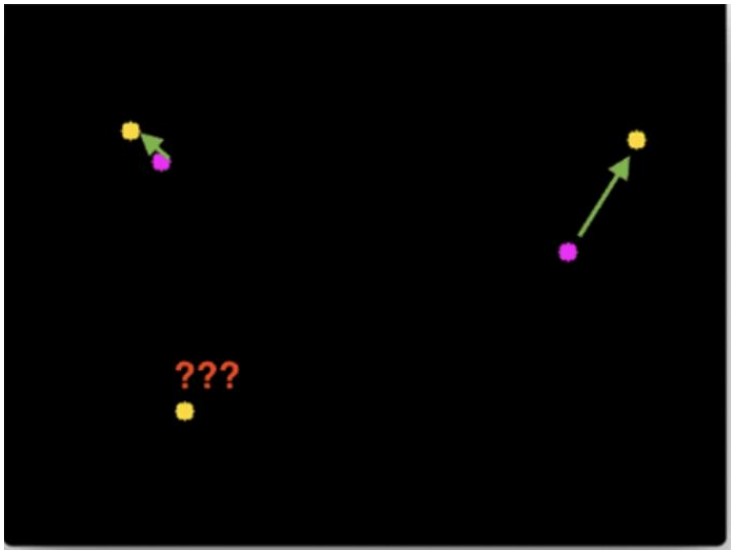



Figure 4: Pairs with Euclidian distances less than the threshold are assumed to be the same object.

Once an object with a Euclidian distance more than threshold is found then this object is assigned a new unique identifier.

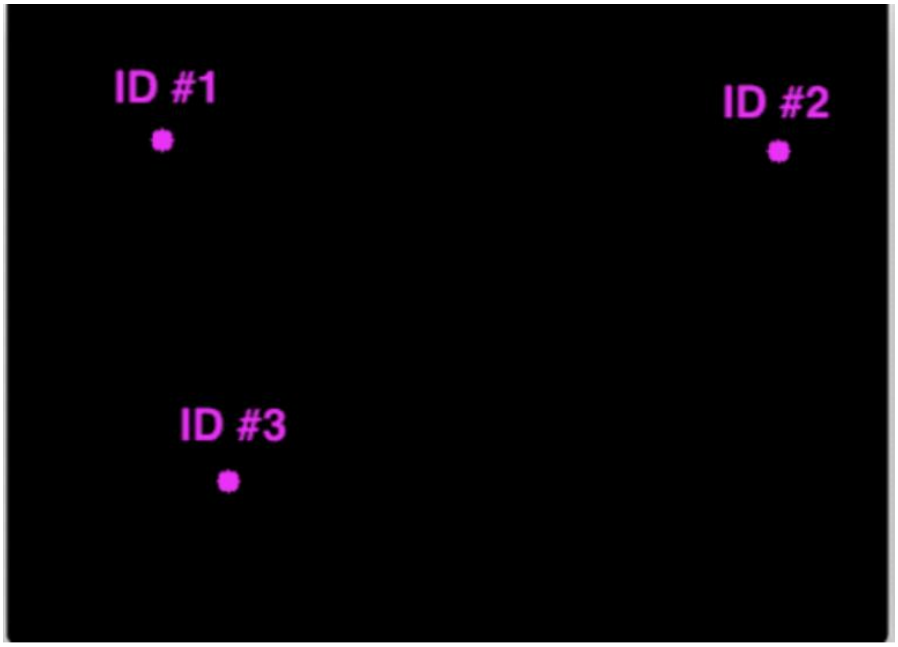



Figure 5: Object with identifier #3 is a newly assigned object.

One problem that can be encountered while using the centroid algorithm for tracking objects is that there can be identifier switching. If two objects that are identical in size pass each other or overlap, the algorithm may swap their identifiers because the Euclidean distance between them may be less than the required threshold.

# CHAPTER 4

# EXPERIMENTATION AND RESULTS

## 4.1 ANGLE AND DISTANCE MEASUREMENTS

### 4.1.1 EXPERIMENT

The first experiment tests the distance and angle measurements obtained from the program written and comparing it to the actual values. This experiment was conducted using the 1080p webcam. As mentioned before, the angle of view – the widest angle this camera can capture – is 110 degrees. So, it can see 55 degrees to the left of itself and the same amount to the right of itself. The materials used in conducting this experiment were measuring tape, 1080p camera, protractor, green flat boxes. In this experiment, we would be simulating the green flat boxes as cars. Why are we using green flat boxes as cars? We could use a ML model to find an actual, but this adds complexity. To remove variables from the experiment, thus removing complexity, we use the green boxes as an analog for a car. The goal is to be able to identify the cars, determine how far away and at what angle each car is from the camera. The camera in this case simulates a lead car which would be on the road tracking the other cars. The experiment was set up by placing the camera on a flat surface and using the measuring tape to mark a distance from the camera. Then, the target would be held at that distance that was just marked and the distance estimated from the program would be recorded. At each distance, the measurement was taken three times, once at positive fifty-five degrees – all the way to the right of the camera, once at negative fifty-five – all the way to the left of the camera, and once at zero degrees, that is, directly in front of the camera. To test the angle measurement, fifty-five degrees was measured on both sides of the camera and a line was drawn at positive and negative 55 degrees. The targets were held at these lines and the angle measured by the program was recorded.

The results of the experiment are shown below:

| Test No. | Expected | Measured | Percentage Error |
|---|---|---|---|
| 1 | 12 | 12.3 | 2.50 |
| 2 | 12 | 12.1 | 0.83 |
| 3 | 12 | 11.8 | 1.67 |

Table 1: Comparing the measured distance with an expected distance of 12 inches.

| Metric | Value |
|---|---|
| **Average Percentage Error** | 1.67 |
| **Mean** | 12.06 |
| **Standard Deviation** | 0.25 |
| **Max** | 12.30 |
| **Min** | 11.80 |

Table 2: Statistical Metrics for results from Table 1

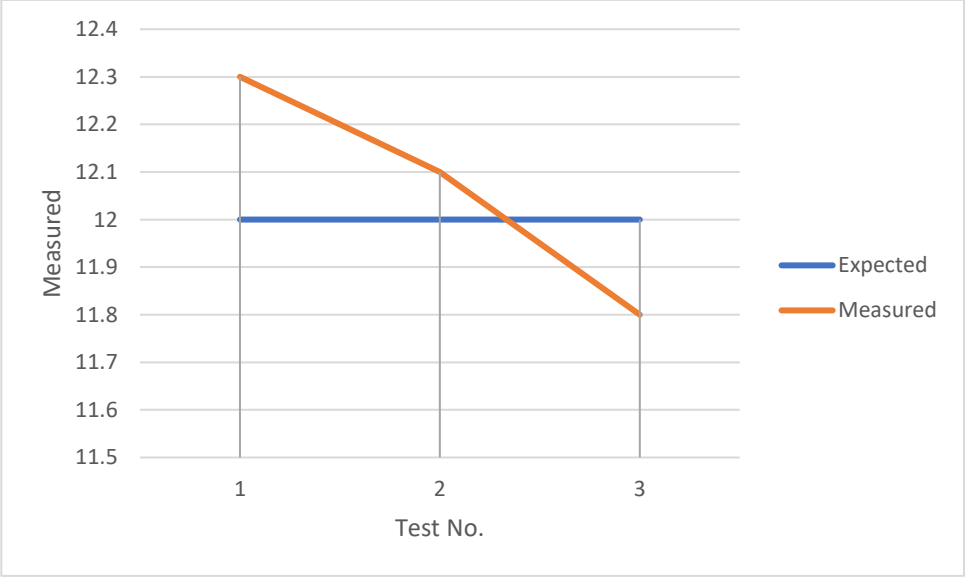



Figure 6: Line chart comparing the expected and measured distance across the tests.

For this test, the expected distance was 12 inches. The test was run three times, once from the leftmost point of the camera, once directly in front of the camera, and once at the rightmost point of the camera. The average percentage error was 1.67% and test 2 – object right in front the camera – had the least percentage error, the measured distance was only off by a tenth of an inch.

| Test No. | Expected | Measured | Percentage Error |
|---|---|---|---|
| **1** | 24 | 24.8 | 3.33 |
| **2** | 24 | 24.1 | 0.42 |
| **3** | 24 | 23.6 | 1.67 |

Table 3: Comparing the measured distance with an expected distance of 24 inches.

| Metric | Value |
|---|---|
| **Average Percentage Error** | 1.81 |
| **Mean** | 24.17 |
| **Standard Deviation** | 0.61 |
| **Max** | 24.8 |
| **Min** | 23.6 |

Table 4: Statistical Metrics for results from Table 3

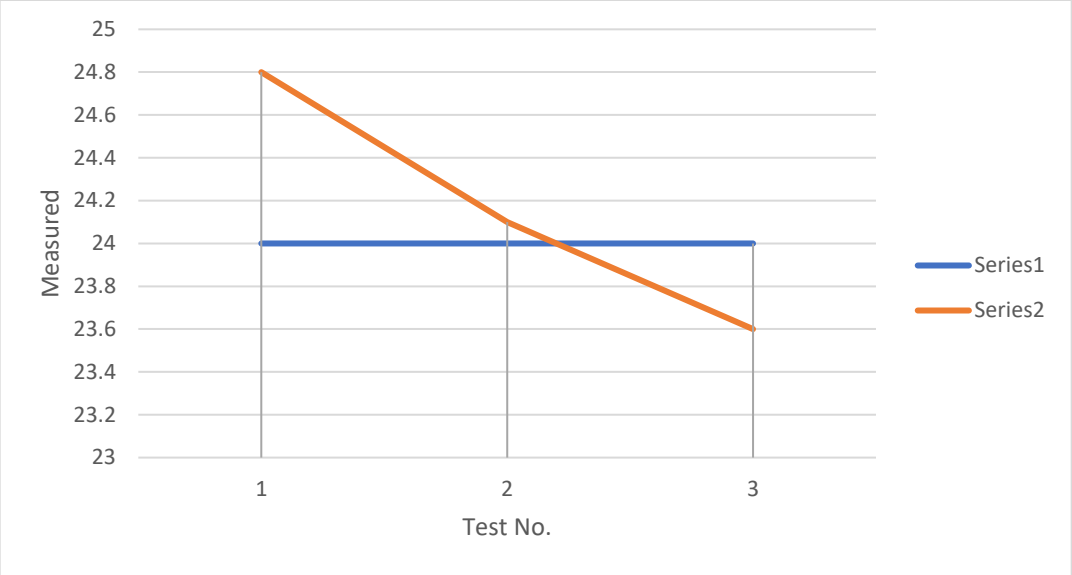



Figure 7: Line chart comparing the expected and measured distance across the tests.

For this test, the expected distance was 24 inches. The test was run three times, once from the leftmost point of the camera, once directly in front of the camera, and once at the rightmost point of the camera. The average percentage error was 2.03% and test 2 – object right in front the camera – had the least percentage error, the measured distance was off by a tenth of an inch.

| Test No. | Expected | Measured | Percentage Error |
|---|---|---|---|
| **1** | 36 | 36.8 | 2.22 |
| **2** | 36 | 35.8 | 0.56 |
| **3** | 36 | 37.2 | 3.33 |

Table 5: Comparing the measured distance with an expected distance of 36 inches.

| Metric | Value |
|---|---|
| **Average Percentage Error** | 2.04 |
| **Mean** | 36.60 |
| **Standard Deviation** | 0.72 |
| **Max** | 37.2 |

| | |
|---|---|
| **Min** | 35.8 |

Table 6: Statistical Metrics for results from Table 5

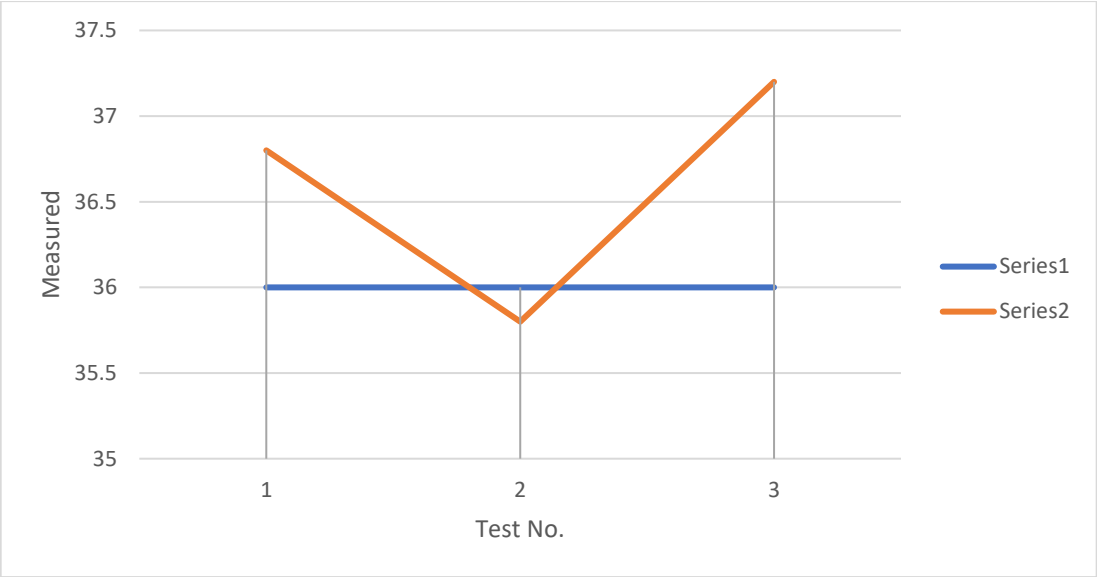



Figure 8: Line chart comparing the expected and measured distance across the tests.

For this test, the expected distance was 36 inches. The test was run three times, once from the leftmost point of the camera, once directly in front of the camera, and once at the rightmost point of the camera. The average percentage error was 2.84% and test 2 – object right in front the camera – had the least percentage error, the measured distance was of by two tenths of an inch.

| Test No. | Expected | Measured | Percentage Error |
|---|---|---|---|
| **1** | 48 | 50.1 | 4.38 |
| **2** | 48 | 47.2 | 1.67 |
| **3** | 48 | 46.8 | 2.50 |

Table 7: Comparing the measured distance with an expected distance of 48 inches

| Metric | Value |
|---|---|

| | |
|---|---|
| **Average Percentage Error** | 2.85 |
| **Mean** | 48.03 |
| **Standard Deviation** | 1.80 |
| **Max** | 50.1 |
| **Min** | 46.8 |

Table 8: Statistical Metrics for results from Table 7

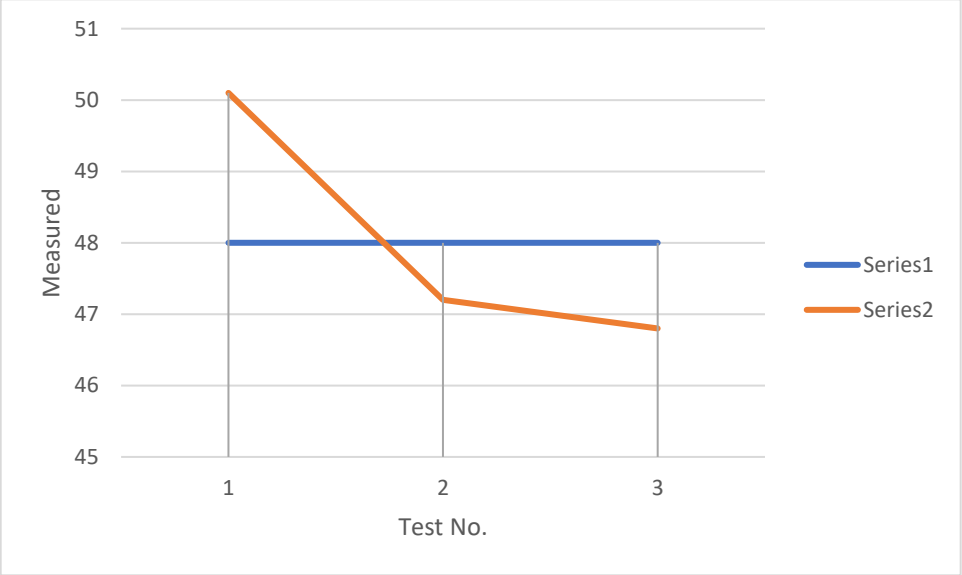



Figure 9: Line chart comparing the expected and measured distance across the tests.

For this test, the expected distance was 48 inches. The test was run three times, once from the leftmost point of the camera, once directly in front of the camera, and once at the rightmost point of the camera. The average percentage error was 2.03% and test 2 – object right in front the camera – had the least percentage error. Albeit the tests at this distance has higher percentage errors than the tests at closer distances.

| Distance | Average Percentage Error |
|---|---|
| 12 | 1.67 |
| 24 | 1.81 |

| | |
|---|---|
| 36 | 2.04 |
| 48 | 2.85 |

Table 9: Comparing the percentage error with the distance from the camera.

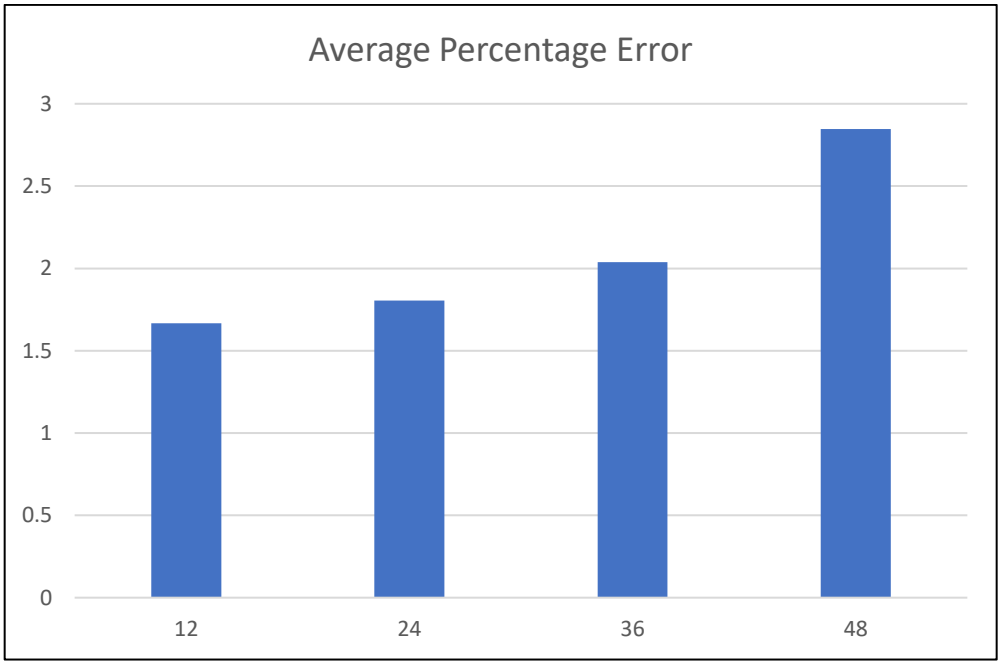



Figure 10: Bar chart showing the average percentage error for the tests at different distances.

From the figure shown above, as the distance increased, the percentage error increased. Although percentage error remained relatively small for all tests. This correlation between the increase in distance and the percentage error was expected because as the object goes farther and farther away from the camera, it would be harder to track.

TESTING ANGLE MEASUREMENTS:

| Test No. | Expected | Measured | Percentage Error |
|---|---|---|---|

| | | | |
|---|---|---|---|
| **1** | 0 | 1.52 | 1.52 |
| **2** | 0 | 1.73 | 1.73 |
| **3** | 0 | 0.97 | 0.97 |

Table 10: Comparing the measured angle with an expected angle of 0 degrees.

Average percentage error with an angle of 0 degrees: 1.41 %

| Test No. | Expected | Measured | Percentage Error |
|---|---|---|---|
| **1** | 55 | 58.26 | 5.92 |
| **2** | 55 | 57.49 | 4.53 |
| **3** | 55 | 57.21 | 4.02 |

Table 11: Comparing the measured angle with an expected angle of 55 degrees.

Average percentage error with an angle of 55 degrees: 4.82%

| Test No. | Expected | Measured | Percentage Error |
|---|---|---|---|
| **1** | -55 | -53.72 | 2.33 |
| **2** | -55 | -52.98 | 3.67 |
| **3** | -55 | -53.89 | 2.02 |

Table 12: Comparing the measured angle with an expected angle of -55 degrees.

Average percentage error with an angle of -55 degrees: 2.67%

The accuracy of the angle measurement was tested by measuring the reading at the leftmost point of the field of view of the camera, at the rightmost point, and right in front of the camera. The percentage error was the lowest when the object was right in front of the camera and this was expected. Also, the percentage error remained relatively small throughout all the tests with a total average percentage error of 2.97%.

### 4.1.2 DISCUSSION

Although the average percentage errors across all tests were relatively low, it is necessary to try and examine what was causing these errors. This test is simulating cars being tracked on the road so the distance would be scaled up. This means that a difference of a couple of inches in the simulation could be a big deal when it is scaled up to tracking moving cars on the road. The first and most obvious reason for error is the camera being used. The camera being used for this experiment is a MacBook pro webcam. This is a limiting factor on the tests because of the limited resolution of this web camera relative to more powerful cameras. A better camera with a higher resolution and a higher frame rate would be better for tracking objects and would be likely to reduce the percentage error. Throughout these tests, the program is placing a bounding box around the tracked objects and is using the coordinates of these bounding boxes coupled with other variables to compute the angle and distance of the object. These bounding boxes are placed by using a python OpenCV model, but if a better trained model was being used, these bounding boxes could be placed much tighter and more accurately around the objects and this would lead to more accurate measurements. Furthermore, the program running to calculate these metrics relies on the color of the object being tracked. In the experiments, it is tracking the targets so even if there were identical targets in size but the color was different, these targets would not be tracked. The drawback from this is if there is any other object with the same color this could confuse the program and produce incorrect results. When this experiment is done on the road with cars, the program would not be working based on colors but would be working on a model that is trained to recognize and count cars. Lastly, this experiment was performed when the objects being tracked were stationary but, on the road, cars would be moving and moving at high speeds.

A pattern noticed when testing this program is that the higher the speed of object being tracked the lower the accuracy to measuring the angle and distance. This could be caused by the resolution and frame rate of the camera and also the processing power of the computer the program is being ran on.

## 4.2 COUNTING OF CARS

### 4.2.1 EXPERIMENT

One of the integral components of this paper is being able to count the cars on the road using the simulated mobile observer methods and to use this information to calculate fundamental traffic parameters such as flow, density, and velocity of vehicles on the road. To test this at a much smaller scale, a green square measuring 2.5 inches by 2.5 inches is used as an analog for a car; it is called throughout this section the target. This experiment was setup to analyze whether the program can identify the target, show that is has detected it by placing a bounding box around it and assigning it a number and continue to track it as long as it is in the camera's field of view. The centroid tracking algorithm is being used here. The program should be able to do this with any number of green squares available. The number of green squares is not a limiting factor. This simulates that the camera is a lead car driving on the road and it encounters other cars and tracks how many cars it passes and how many cars it has passed. If a car is seen it is assigned a number and when that car leaves the field of view that means the lead car has overtaken, and it is counted as a car that was passed. When a car is in view and goes so far out of view through the top of the screen such that the lead car can no longer track it, this car is counted as car that has passed the lead car. There were three scenarios in this experiment: the camera is stationary, and the green boxes pass the sides, the green boxes are stationary, and the camera passes them, and the last scenario is that the camera and green boxes are both moving but are moving in opposite directions. This experiment was performed by taping the green squares to the top of skateboards so the skateboards could be given a push and the boxes would glide along. In some scenarios, the skateboards were not pushed but held and moved steadily to try and control for the speed at which the green boxes were moved across the screen. The camera used was a MacBook pro webcam and the experiment was performed in a bright room that had no other green color that could tamper with the readings obtained from the program.

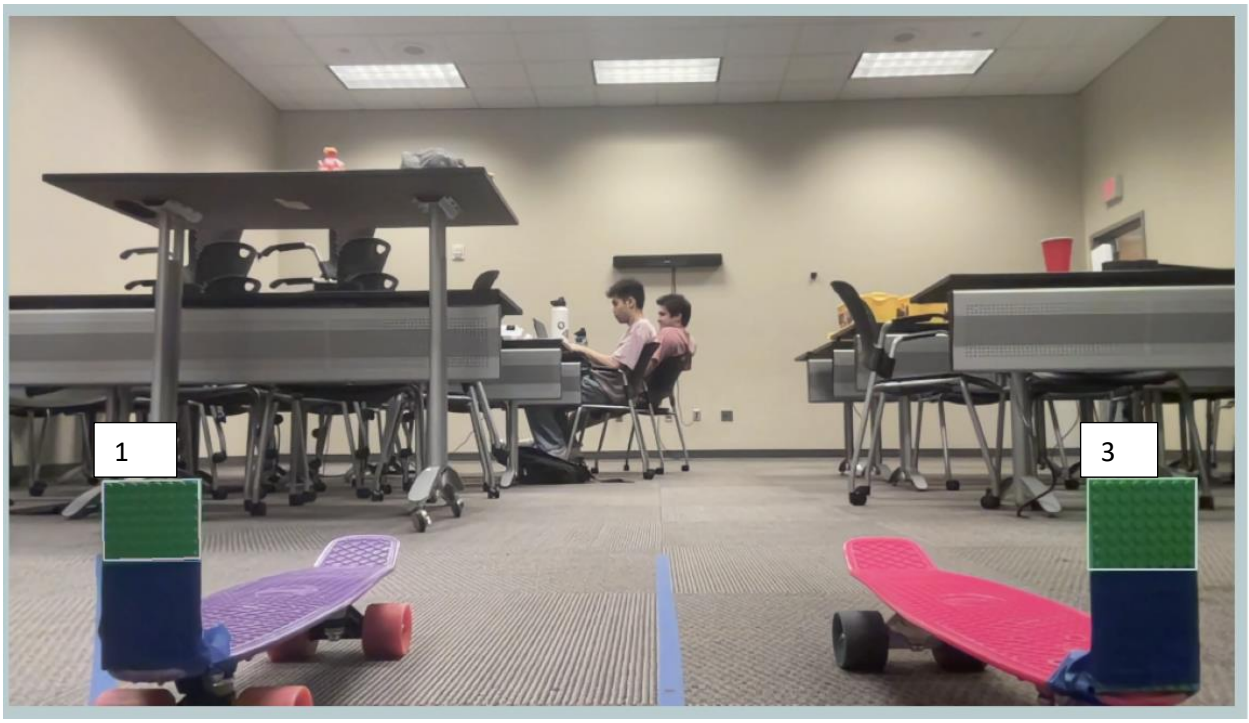



Figure 13: Green box on the left tracked as car 1 and green box on the right tracked as car 3.

The figure above is a screenshot from a video of the green boxes moving past the camera farther and farther away until they could not be tracked anymore. The green box on the left came on the screen first hence it took the tracking number 1 and the green box on the right came second and it showed a tracking number of 2 at first then changed to three indicating that the program thought it was a third green box on the screen.

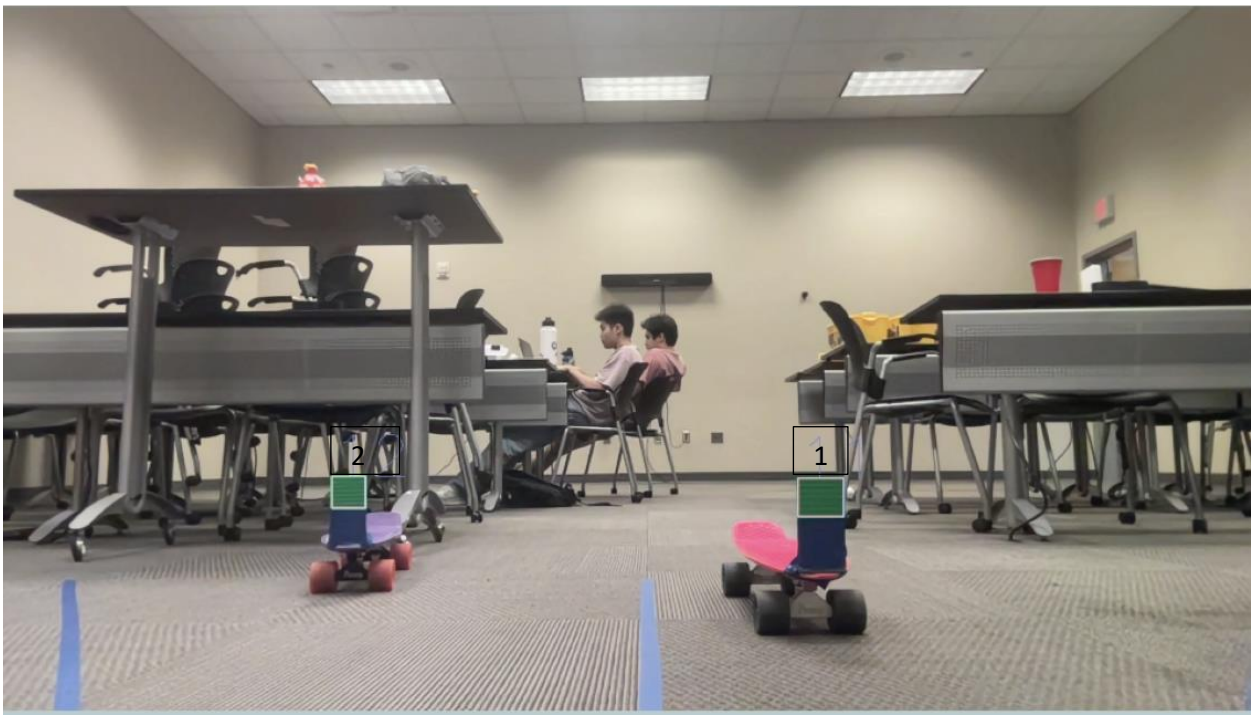



Figure 14: Green box on the right showing a tracking number of 1 and green box on the left shows a tracking number of 2.

The figure above is a screenshot of the second test where the camera is stationary, and the green boxes move across the screen. The results in this test were correct. The green box on the right entered the camera's field of view first so it was assigned a tracking number of 1 and the green box on the left entered the field of view second so it was assigned a tracking number of 2. In the screenshot, box 2 is ahead of box 1. This is because the skateboard holding box 2 was pushed with more force because it was necessary to try and simulate varying speeds of boxes moving across the screen. The boxes continued to move until they were far enough that the camera could no longer track them on the screen.


Figure 15: The camera is stationary, but the boxes are moving towards the camera.

In this test, the camera is stationary, and the green boxes are moving towards it from the opposite direction from the prior tests. This is supposed to simulate the lead car tracking cars on the other side of the road going an opposite direction to itself. The green boxes were placed far away from the camera so it could not track it from the starting distance but as the boxes started to come closer, it could track it. The results from this experiment were not as expected because the green box on the left came on the screen first and has a tracking number of 8. The initial tracking number was 1 and the tracking number 2 was assigned to the green box on the right, but as the boxes continued to come closer to the camera, the tracking number of the box on the left continued to go up.

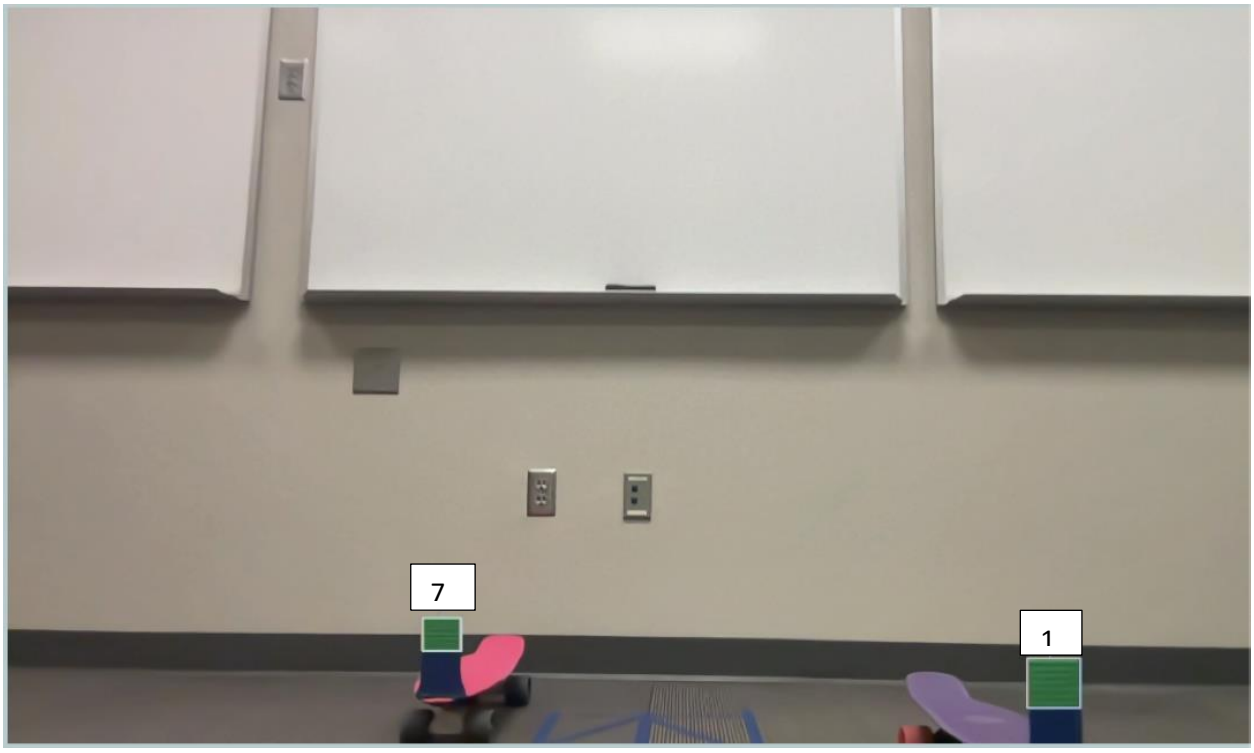



Figure 16: The green boxes were stationary and the camera moved towards them.

In this test, the boxes were stationary but the box on the right was closer to the camera. Initially, the camera was far enough away that it could not track the boxes. This is because so it doesn't look like the boxes just appeared on the screen because that would be an incorrect simulation. A car would be driving towards another car from a distance and as it gets closer its tracking of that car would get better. In this test, the camera was placed on a skateboard and pushed hard to simulate that the car was moving fast. As expected, since the box on the right was closer, it was tracked first and giving the tracking number 1. The box on the left was given the tracking number 2 initially, but as the camera kept getting closer the tracking number kept increasing.

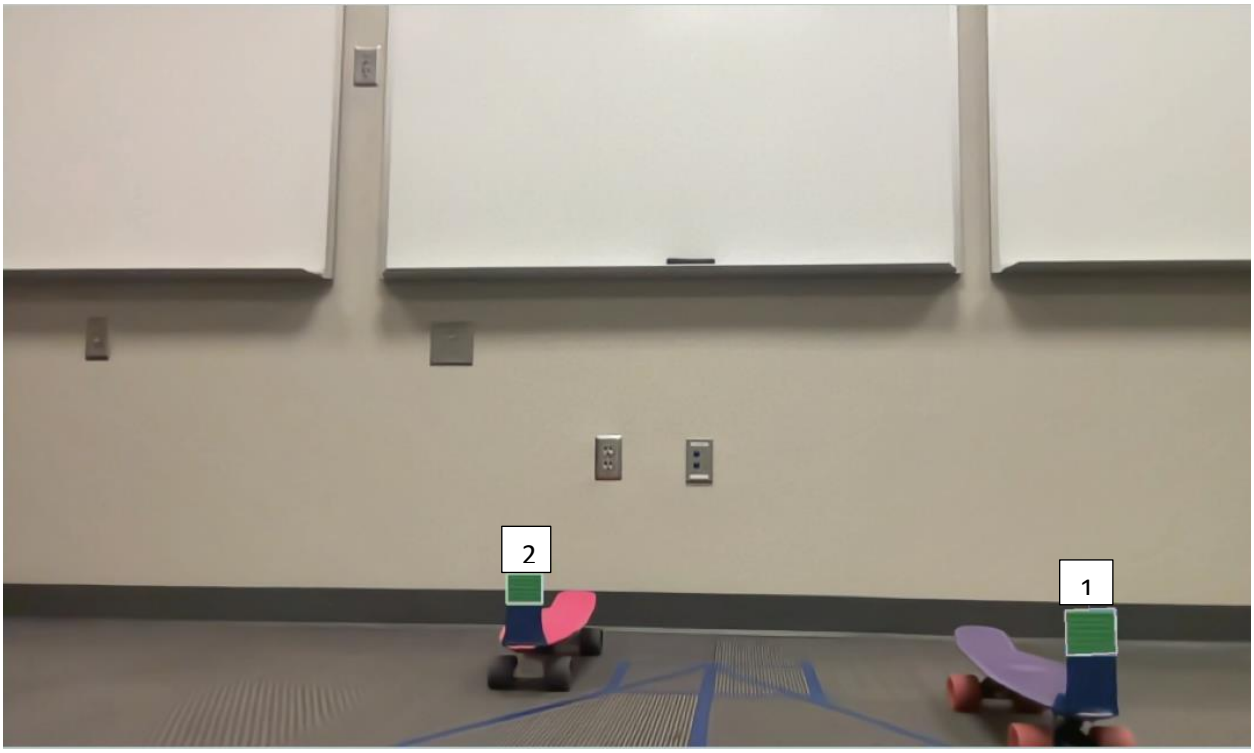



Figure 17: The green boxes were stationary and the camera moved towards them.

In this test, the camera was initially at a far enough distance that it could not track the boxes but as it moved closer to them it could track them. In this experiment, the skateboard with the camera on top of it was guided slowly to pass the boxes and get to the wall. The results from this test were very accurate because the camera saw the box on the right first and assigned it a tracking number of 1 and saw the box on the left second and assigned it a tracking number of 2. It maintained these numbers as it passed the boxes. After the camera had completely passed the first box and it couldn't see it anymore, it continued to track the second box on the left and it maintained its tracking number of 2.

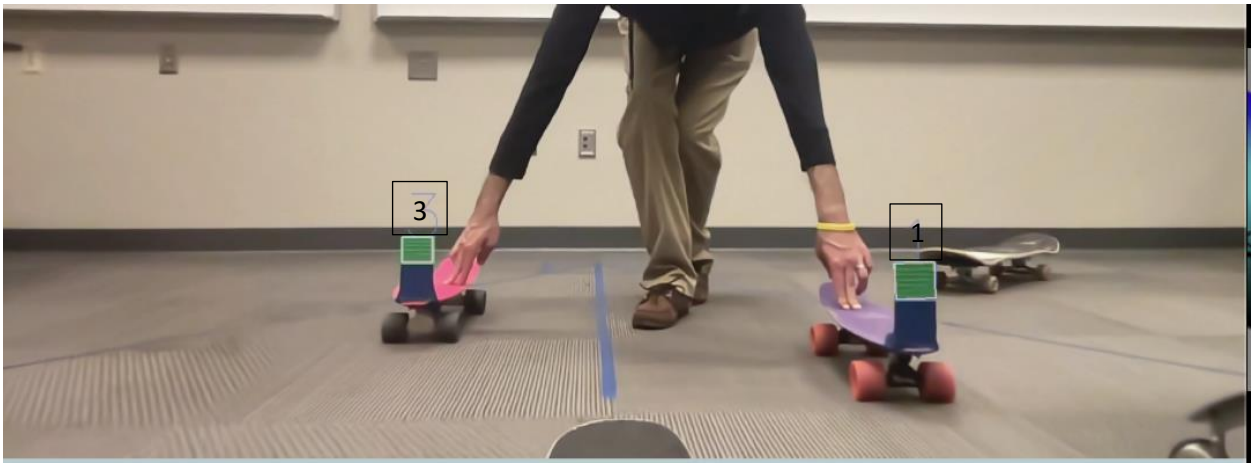



Figure 18: The camera and boxes are both moving in opposite directions.

In this test, the camera started far away enough so that it could not track the boxes from its starting point. As the distance between the camera and the boxes reduced it could track them. In this test, the skateboard with the camera on top of it was pushed to simulate speed and the green boxes were guided slowly. As expected, the box on the right was assigned a tracking number of 1 because it is closer, and the camera would have seen it first. Then, the box on the left was initially assigned a tracking number of 2 but it was reassigned to 3.

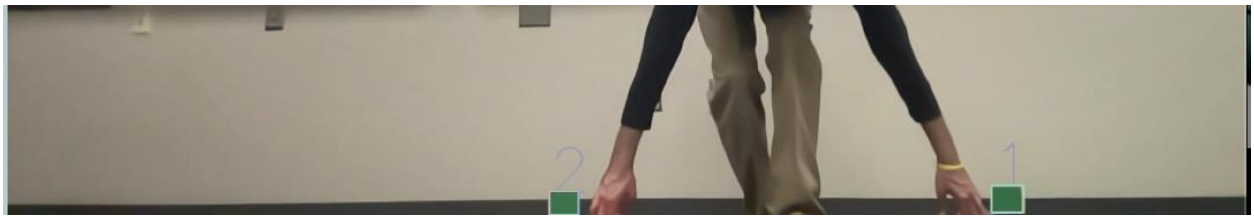



Figure 19: The camera and boxes are both moving in opposite directions.

In this test, the camera was initially far enough away so it could not initially track the boxes. As the boxes and the camera started to move to one another, the camera could track them. In this test, the camera was pushed to simulate speed, it was gently guided until it past the green boxes. The results from

this test were accurate. The program assigned the box on the right a tracking number of 1 and the box on the left a tracking number of 2.

### 4.2.2 DISCUSSION

From the tests conducted, it was evident that when the camera was moving quickly or the boxes were moving quickly, then the accuracy of counting the cars reduced. This could be because of the way the cars are being tracked. Using centroid tracking, the computer would compare the centroid of the bounding box from frame to frame. If the distance between them is greater than a specific threshold, then a new number is assigned to that box. When the green boxes are moving through the screen quickly, the distance between the centroids is likely greater than the threshold from frame to frame. There are a couple of ways to fix this: one way is to increase the distance threshold, but this also comes with its problems. Increasing the distance threshold can cause confusion in the program if two boxes are very close to one another because the program will think these two different boxes are the same and count them as one. Another way to mitigate this problem is to use a more powerful camera that can handle higher frame rates. If the frame rate was higher, even if the box is moving quickly across the screen, when examined frame by frame there would be not much difference in the position. One other way is to utilize a tracker that does not track the centroid of the bounding box. This was tested while working on this paper and the centroid tracker still had the highest accuracy over other trackers which we tried.

With the conjunction of the data collected from distance and angle measurements and the tracking of cars, can traffic parameters be estimated? The answer is yes. The fundamental traffic parameters are flow, density, and velocity. Flow is calculated by dividing the number of cars on a section of the road by the time it takes them to pass through that section. Using the program used in conducting the prior experiments, the number of cars can be obtained tracking the cars that the lead car sees. This number would then be divided by how long it took the lead car to pass through that section of the road and this would give the first traffic parameter – flow. Density is calculated by dividing the number of cars in a section of the road by the length of that section of road. To calculate density, the length of road is a known variable. The

number of cars in that section of road would be computed by tracking program used in the experiments in this paper. Once the lead car drives through that section of road and counts the number of cars there, this number would be divided by the length of the road and the road density would be calculated. Calculating the average velocity of the cars on the road can be calculated by using a relationship between the fundamental traffic parameters, average velocity is computed by dividing the flow by the density. Since these two parameters can be computed as explained earlier, their division would give the third fundamental traffic parameter. The formula used is show below:

$$V_{avg} = \frac{Flow, q}{Density, k}$$

# CHAPTER 5

# CONCLUSION

The goal of this thesis was to try and explore a way to be able to use a camera-only approach to estimate traffic parameters using mobile observer methods. Though this was being done on a smaller scale, the goal was to simulate a real-world situation where a lead car (connected vehicle) will be used to track other cars on the road and use the information gathered to help traffic engineers estimate fundamental traffic parameters and also have the ability to share and sync this information with other cars on the road. To simulate this, the first step was implementing a program that could accurately determine how far other cars are on the road and estimating the angle connecting the lead car to other cars on the road. This was achieved and the program is able to determine the distance of cars around it and the angle with about a three percent margin of error.

The second step was to be able to count the cars on the road. The scenarios were tested: when the lead car is stationary and other cars are driving by, when the lead car is in motion and other cars are stationary, and when the both the lead car and other cars on the road are in motion and travelling in opposite directions. The accuracy from performing this experiment was fair but could have been better. Also, it was determined that the accuracy dwindles as the speed of car increases but this could be handled by employing superior cameras to the ones used in the experiment.

This thesis adds to bodies of works pertaining to image processing, traffic parameter estimation, and most importantly, mobile observer methods.

# CHAPTER 6

# FUTURE WORK

The end goal of this thesis is to be able to test the experiments performed on real cars on a highway. But before getting there, there are other things that can be improved to make the testing better. One would be to get a higher resolution camera that can handle higher frame rates which would be much better for tracking and produce more accurate results. Another would be using smaller cars instead of boxes. This would make the simulation closer to a real-world setting. In this case, a model will need to be trained to track and count these cars. Also, having more cameras so that a 360 view of the surroundings of the lead car is possible. A system similar to the surround view system can be created by using the distance and angle measurements obtained from the experiments discussed earlier. This system will not work by creating a 360-degree image of the surrounding of the object being tracked but it will be to produce an image showing the target and all other objects around it placed accurately by scaling the distance and the angle measured to fit this new image. The limiting factor here is that in the experiment done, there was only camera so the image produced will only be able to show the objects in front of the camera and fifty-five degrees on each side. But if, for example, there were three more cameras feeding data into the program, it would be able to get the distance and angle of all other objects to the right, left and behind it to form a true surround-view-like system. Another interesting area of research here is in predictive analysis. After tracking a car's movement for some time can this information be used to predict where the car would move next?

# REFERENCES


Rahul, & Nair, B. B. (2018). Camera-based object detection, identification and distance estimation. *2018 2nd International Conference on Micro-Electronics and Telecommunication Engineering (ICMETE)*. https://doi.org/10.1109/icmete.2018.00052

Liang, H., Ma, Z., & Zhang, Q. (2022). Self-supervised object distance estimation using a monocular camera. *Sensors, 22*(8), 2936. https://doi.org/10.3390/s22082936

Wardrop, J. G., & Charlesworth, G. (1954). A method of estimating speed and flow of traffic from a moving vehicle. *Proceedings of the Institution of Civil Engineers*, *3*(1), 158–171. https://doi.org/10.1680/ipeds.1954.11628

Florin, R., & Olariu, S. (2017). On a variant of the Mobile Observer Method. *IEEE Transactions on Intelligent Transportation Systems, 18*(2), 441–449. https://doi.org/10.1109/tits.2016.2582511

Florin, R., & Olariu, S. (2020). Towards real-time density estimation using vehicle-to-vehicle communications. *Transportation Research Part B: Methodological, 138,* 435–456. https://doi.org/10.1016/j.trb.2020.06.001

Florin, R., & Olariu, S. (2023). Real-time traffic density estimation: Putting on-coming traffic to work. *IEEE Transactions on Intelligent Transportation Systems, 24*(1), 1374–1383. https://doi.org/10.1109/tits.2022.3184843

*Photo source:HSV cylinder.png - wikimedia commons*. (n.d.). Retrieved March 26, 2023, from https://commons.wikimedia.org/wiki/File:HSV_cylinder.png

*Intelligent around view monitor: Innovation*. Innovation |. (n.d.). Retrieved March 25, 2023, from https://www.nissan-global.com/EN/INNOVATION/TECHNOLOGY/ARCHIVE/IAVM/

*Early estimates of motor Photo source: USDOT Connected Vehicle Safety Pilot Program - its.dot.gov*. (n.d.). Retrieved March 26, 2023, from https://www.its.dot.gov/factsheets/pdf/JPO_SafetyPilot.pdf

*USDOT Connected Vehicle Safety Pilot Program - its.dot.gov*. (n.d.). Retrieved March 26, 2023, from https://www.its.dot.gov/factsheets/pdf/JPO_SafetyPilot.pdf

Cohen, J. (2021, August 3). *Computer Vision for tracking*. Medium. Retrieved March 26, 2023, from https://thinkautonomous.medium.com/computer-vision-for-tracking-8220759eee85

*Dedicated short range communications (DSRC) service*. Federal Communications Commission. (n.d.). Retrieved April 17, 2023, from https://www.fcc.gov/wireless/bureau-divisions/mobility-division/dedicated-short-range-communications-dsrc-service

Mulligan, A.-M., & Nicholson, A. (2002). Uncertainty in traffic flow estimation using the moving observer method. *Institute of Professional Engineers New Zealand*.

Langer, M., Schien, T., Harth, M., Kates, R., & Bogenberger, K. (2020). An improved moving observer method for traffic flow estimation at signalized intersections. *2020 IEEE Intelligent Vehicles Symposium (IV)*. https://doi.org/10.1109/iv47402.2020.9304762

Redmill, K. A., Coifman, B., McCord, M., & Mishalani, R. G. (2011). Using transit or municipal vehicles as moving observer platforms for large scale collection of traffic and Transportation System Information. *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. https://doi.org/10.1109/itsc.2011.6083132

Lin, J., Huang, Y., Su, X., Su, Z., & Zhao, P. (2019). An in-vehicle camera based traffic estimation in Smart Transportation. *2019 IEEE 5th International Conference on Computer and Communications (ICCC)*. https://doi.org/10.1109/iccc47050.2019.9064182

*Early estimates of motor vehicle traffic fatalities and fatality rate ...* (n.d.). Retrieved April 17, 2023, from https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/813298

Media, N. H. T. S. A. (2016, March 17). *U.S. dot and IIHS announce historic commitment of 20 automakers to make automatic emergency braking standard on new vehicles*. NHTSA.