FINAL REPORT

Project H2

May 2021

Fly-By Image Processing for Real-Time Congestion Mitigation

Dr. Nasim Uddin | University of Alabama at Birmingham

Dr. Billy Williams | North Carolina State University

Abdel Aziz Abdel Latef | University of Alabama at Birmingham

STRIDE | Southeastern Transportation Research,
Innovation, Development and Education Center

UF | Transportation Institute
UNIVERSITY *of* FLORIDA

TECHNICAL REPORT DOCUMENTATION PAGE

1. Report No. Project H2	**2. Government Accession No.**	**3. Recipient's Catalog No.**

4. Title and Subtitle Fly-By Image Processing for Real-Time Congestion Mitigation	**5. Report Date** May 13, 2021 **6. Performing Organization Code**
7. Author(s) Dr. Nasim Uddin, University of Alabama at Birmingham Dr. Billy Williams, North Carolina State University Abdel Aziz Abdel Latef (Doctoral Candidate) University of Alabama at Birmingham	**8. Performing Organization Report No.** Project H2
9. Performing Organization Name and Address Department of Civil, Construction, & Environmental Engineering 1075 13th Street South Birmingham, AL 35294-4440 North Carolina State University 915 Partners Way Raleigh, NC 27695-7908	**10. Work Unit No.** **11. Contract or Grant No.** Funding Agreement Number 69A355174710
12. Sponsoring Agency Name and Address University of Florida Transportation Institute Southeastern Transportation Research, Innovation, Development & Education Center 365 Weil Hall, P.O. Box, Gainesville, FL 32611 U.S Department of Transportation/Office of Research, Development & Tech 1200 New Jersey Avenue, SE Washington, DC 20590 United States	**13. Type of Report and Period Covered** 8/1/2018 to 5/13/2021 **14. Sponsoring Agency Code**

15. Supplementary Notes

16. Abstract

Traffic monitoring is the centerpiece of congestion mitigation and traffic management. Whilst surveillance technologies have matured enough to provide informative depiction for the traffic, the current state-of-the-art systems cannot support immediate congestion problems. Proactive congestion mitigation requires a) real-time surveillance for traffic parameters, b) prediction for imminent congestion onset, to: c) inform responsible parties to take immediate actions to prevent congestion. The proposed congestion mitigation approach is based on the premise that a short time analysis (1-5 minutes) will be sufficient to manage the congestion. We foresee that using a "flock" of interconnected, self-managed drones, can establish a deployable system to perform immediate monitoring/assessment for traffic conditions to infer if congestion is approached. To detect vehicles, a faster technique of Convolutional Neural Network (CNN) called YOLOv3 is used. In this technique, a single neural network is used to the full image which divides the image into regions and predicts bounding boxes and probabilities for each region. Then these bounding boxes are weighted by the predicted probabilities. This technique requires huge computational power and therefore, GPUs are used to process the videos recorded by drones' cameras. By calibrating the camera using real values compared to their apparent values in images, the detected vehicles can be tracked. The targeted feature (herein, features correlated to traffic congestion) were reproduced utilizing a traffic simulation model. The proposed methodology was tested by collecting and investigating video images from drones. The project, if continued further, has the potential to advance the state of proactive traffic and congestion management by embedding a distributed, simulation-based traffic state prediction system within the integrated drone surveillance software to enable congestion mitigation actions to be undertaken before congestion happens rather than after traffic flow has already broken down.

17. Key Words Traffic Management, Camera Calibration, Object Detection, YOLO, GPU Computation, Drone, Real-time Surveillance, Congestion Mitigation	**18. Distribution Statement** No restrictions

19. Security Classif. (of this report) N/A	**20. Security Classif. (of this page)** N/A	**21. No. of Pages** 36 Pages	**22. Price** N/A

DISCLAIMER

The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the information presented herein. This document is disseminated in the interest of information exchange. The report is funded, partially or entirely, by a grant from the U.S. Department of Transportation's University Transportation Centers Program. However, the U.S. Government assumes no liability for the contents or use thereof.

ACKNOWLEDGEMENT OF SPONSORSHIP AND STAKEHOLDERS

This work was sponsored by a contract from the Southeastern Transportation Research, Innovation, Development and Education Center (STRIDE), a Regional University Transportation Center sponsored by a grant from the U.S. Department of Transportation's University Transportation Centers Program.

Funding Agreement Number - 69A3551747104

LIST OF AUTHORS

Lead PI:

Nasim Uddin, Ph.D., P.E., F. ASCE
University of Alabama at Birmingham
nuddin@uab.edu
0000-0001-6921-5577

Co-PI:

Billy Williams, Ph.D.
North Carolina State University
bmwilli2@ncsu.edu
ORCID 0000-0002-7599-1385

Additional Researchers:

Abdel Aziz Abdel Latef, Ph.D. Candidate
University of Alabama at Birmingham
abdlaziz@uab.edu
0000-0002-2005-794X

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Traffic monitoring is the centerpiece of congestion mitigation and traffic management. Whilst surveillance technologies have matured enough to provide informative depiction for the traffic, the current state-of-the-art systems cannot support immediate congestion problems. Proactive congestion mitigation requires a) real-time surveillance for traffic parameters, b) prediction for imminent congestion onset, in order to c) inform responsible parties to take immediate actions to prevent congestion. The proposed congestion mitigation approach is based on the premise that a short time analysis (1-5 minutes) will be sufficient to manage the congestion. We foresee that using a "flock" of interconnected, self-managed drones, can establish a deployable system to perform immediate monitoring/assessment for traffic conditions to infer if congestion is approached.

To detect vehicles, a faster technique of Convolutional Neural Network (CNN) called YOLOv3 is used. In this technique, a single neural network is used to the full image which divides the image into regions and predicts bounding boxes and probabilities for each region. Then these bounding boxes are weighted by the predicted probabilities. This technique requires huge computational power and therefore, GPUs are used to process the videos recorded by drones' cameras. By calibrating the camera using real values compared to their apparent values in images, the detected vehicles can be tracked. The targeted feature (herein, features correlated to traffic congestion) were reproduced utilizing a traffic simulation model. The proposed methodology was tested by collecting and investigating video images from drones.

The project, if continued further, has the potential to advance the state of proactive traffic and congestion management by embedding a distributed, simulation-based traffic state prediction system within the integrated drone surveillance software to enable congestion mitigation actions to be undertaken before congestion happens rather than after traffic flow has already broken down.

Keywords:
Traffic Management, Camera Calibration, Object Detection, YOLO, GPU Computation, Drone, Real-time Surveillance, Congestion Mitigation

EXECUTIVE SUMMARY

There are attempts being made at creating next generation advanced traffic management systems that provide for proactive system management based on remote surveillance. However, the proactive capabilities are yet to be fully realized. The current vision is focused on permanent surveillance systems, traffic management center-based management, and control software systems. The vision of this project differs in that it seeks to develop a dynamic system that uses a fleet of drones to create an on-demand, flexible surveillance network with distributed state prediction. Potential uses include special events, evacuation management, and even supplemental, re-deployable support in urban areas with some level of permanent surveillance infrastructure.

Remote imagery has recently been extensively used as a mean of characterizing traffic properties. Existing techniques relay only on forward image analysis by tracing the targeted feature in the image pixel by pixel. This framework requires immense computational capabilities that requires strong processors like GPUs. Therefore, a faster neural network method called YOLOv3 was used along with Nvidia GPU through CUDA C++ to detect vehicles. In other Convolutional Neural Network (CNN) methods, thousands of neural networks are required for a single image. However, YOLOv3 technique requires only a single network evaluation which makes it much faster. In this approach, drones record and transfer the current traffic flow to the computational station that determines its characteristics. This functionality is realized by incorporating short-term prediction algorithms into the system. Then it provides accurate predictions of the dynamic traffic conditions with available computational power. The real properties of the traffic are estimated by calibrating the used cameras via analogy between some lengths in both real scene and recorded video. The calibrated camera can estimate the location and velocity of detected vehicles. The traffic prediction elements include both simple time series-based forecast methods along with context-based models. The information in turn feeds into a traffic management and control simulation model that enables real-time, proactive management and control actions. Enabling an immediate mitigation sensing model utilizing the drone swarm implicitly requires: a) good modeling for the traffic bottlenecks to characterize features correlated with congestion onset, and b) an in-field intelligent reconnaissance device that can recognize these features when present.

This project aims to advance the field of real-time traffic condition monitoring by developing a rapidly deployable and dynamically flexible detection system using, as mentioned above, interconnected, self-managed swarm of drones, and by developing a robust, simulation-based traffic state analysis and prediction system. These contributions will enable effective real-time traffic management and congestion mitigation.

For future developments, the newly developed YOLOv4 will be used to achieve faster and more accurate detection. Also, more powerful GPU cards will be tested to approach real-time detection with the required efficiency.

1. PROPLEM DESCRIPTION

Traffic management is the primary key to handling and solving different traffic problems. It can also be used to protect transportation infrastructure from possible accidents. Remote surveillance can be used for this purpose using drones or fixed cameras. Images captured from these cameras are analyzed to get the properties of the observed traffic. An image of a camera represents a scaled flat view of the real street. Every point in the real scene is represented as a pixel on the image. However, getting detailed information and dimensions from this image about the real scene is not an easy task because of the image distortion that occurs due to many factors. The distortion factors include the height of the camera, the camera's resolution and focal properties, and the camera orientation. A simple mathematical model is used to estimate these image distortion factors and hence, construct the necessary formula to relate the coordinates of the real objects to their respective locations on the image. The model estimation of the mapping formula was experimentally verified. After estimating the real-world coordinates from the camera image, vehicle detection is conducted to determine the density of traffic in each lane. By using YOLOv3 technique developed by Redmon et al. (2018), vehicles can be detected with reasonable time and good accuracy.

2. LITERATURE REVIEW

For the purpose of observing and analyzing traffic there are two main problems: calibrating cameras and detecting vehicle. The used cameras are mainly fixed surveillance cameras. The calibration is usually done with the help of some features of the real scene and their corresponding in the video images. For example, the usage of vanishing point is commonly used (Schoepflin et al., 2003; Cathey et al., 2003; Grammatikopoulos et al., 2005; and Dubská et al., 2014). It is also possible to combine it with some other measurements as (Wang et al., 2007) used two parallel lane-marking lines with known lane width, and either the camera height or the length of a lane-marking line parallel to the road to calibrate cameras. These calibration methods were used to recover the focal length, the pan angle, the tilt angle, and the camera height. Detection of vehicles can be done by many methods. Grammatikopoulos et al. (2005) used blob detection while Dubská et al. (2014) used background modeling and foreground detection.

3. DRONES AND PERCHING MECHANISM

The drones were equipped with a perching mechanism that allows the drone to land on vertical walls or upside down under horizontal surfaces. Perching enables the drones to shut down their power-consuming motors, thereby allowing the cameras to acquire video over an extended period. The perching mechanism consisted of a small lift sucking solenoid electromagnet that was able to hold 0.5 kg. The electromagnet weight was 15 g and the input volt was 5-12 DC. To resist the high temperature produced by the electromagnet wire, the framework was made from nylon plastic which had a high temperature resistance and was covered by a steel box. It should be noted that the electromagnet had to be light weight so the drone could easily fly. The main advantage of an electromagnet over a permanent magnet was that the magnetic field could be quickly changed by controlling the amount of electric current in the winding. However, unlike a permanent magnet that needs no power, an electromagnet required a continuous supply of current to maintain the magnetic field. The second part in the perching mechanism was the mini relay wireless switch with remote control to allow opening and closing the electromagnet field. The Working Voltage was DC3.5V-DC12V and Max Load was 1A. The RF remote control is adopted as RF technology which provides a very stable signal. Moreover, signals of the wireless RF switch could pass through walls, floors, and doors steadily. Drone flights can be controlled by a receiver from any place within a reliable distance with maximum range up to 164 ft. (50 m) with no obstacles. This proposed perching mechanism is very small in size, flexible in operation, and with high sensitivity and low power consumption. The electromagnet, mini relay, and drone used to make the sticky mechanism are shown in Figure 1.

To validate our perching mechanism, a field experiment was conducted at the University of Alabama at Birmingham (UAB) campus. The goal from this experiment was to make the drone stick to common steel structures found along streets like light columns, advertisement boards, traffic light columns, etc. Figure 2 shows a picture of the drone sticking to steel structures in a horizontal position.

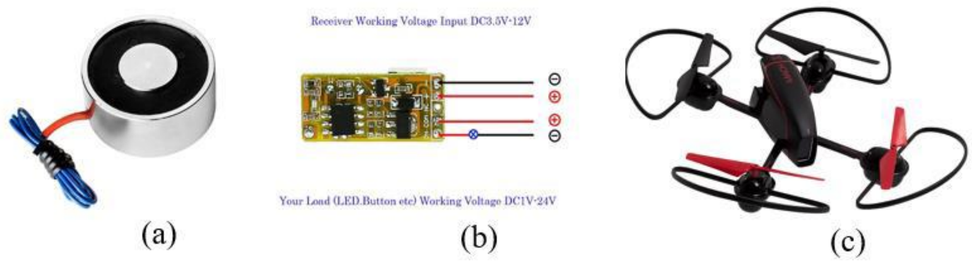

Receiver Working Voltage Input DC3.5V-12V
⊖
⊕
⊕
⊖
Your Load (LED.Button etc) Working Voltage DC1V-24V
(a)
(b)
(c)

Figure 1. (a) Electromagnet, (b) Mini relay wireless switch, and (c) Drone.

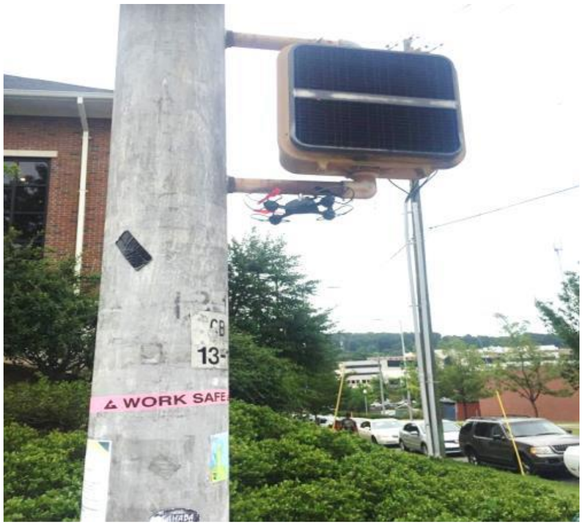


Figure 2. Drone sticking to a steel structure to monitor traffic.

4. ASSUMPTIONS AND FORMULA

The following assumptions underlie the image processing formula:

1. the camera is not moving,
2. the image is received on a flat surface, and
3. the real scene is flat ground.

As shown in Figure 3**Error! Reference source not found.**, the camera can detect a limited area depending on its properties, position, and orientation. The relations between the coordinates on the image and the real coordinates of the observed area can be inferred using simple geometry. There are seven parameters that control these relationships: the camera orientation (3 parameters), the size of the camera's image (2 parameters), the height of the camera's lens (1 parameter), and its focal length (1 parameter).

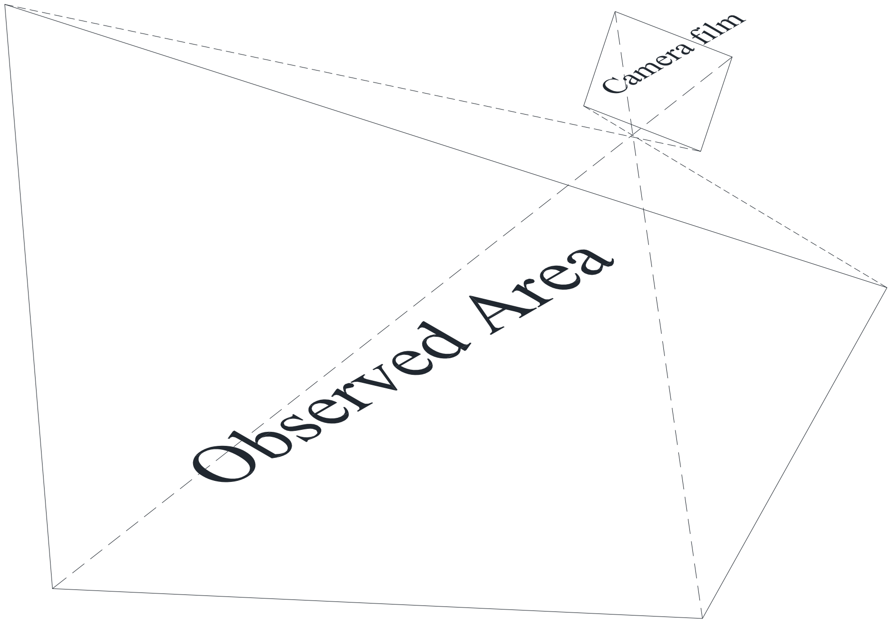


Figure 3. Geometry of camera observation.

According to these assumptions and by using analytical geometry, the real coordinate of any point related to an origin associated with the camera can be estimated. Considering a global coordinate system on the ground where the origin point is the intersection of the ground plane with a line perpendicular to it passing through the lens, the general equation of the image plane is defined as follows:

$$ax + by + z + d = 0 \qquad (1)$$

This equation contains three variables, *a*, *b*, and *d*, which define the external orientation of the camera and its focal length. Considering the coordinates of the origin point of the image x_0, y_0, which represent the size of camera's image, the third coordinate of that point can be estimated with the aid of equation 1, and is evaluated as:

$$z_0 = -(ax_0 + by_0 + d) \qquad (2)$$

By defining the angle of θ with x-axis that represents the projection of the lower edge of image on ground, this angle defines the internal rotation of the camera, the equation of that edge is represented in these equations:

$$\frac{x-x_0}{\cos\theta} = \frac{y-y_0}{\sin\theta} = \frac{z+ax_0+by_0+d}{-(a\cos\theta+b\sin\theta)} \qquad (3)$$

$$x = x_0 + \cos\theta\, t \qquad (4.a)$$

$$y = y_0 + \sin\theta\, t \qquad (4.b)$$

$$z = -(ax_0 + by_0 + d) - (a\cos\theta + b\sin\theta)t \quad (4.c)$$

The last parameter is the height of the camera's lens z_l.

Using these equations and definitions, the line passing through the lens perpendicular to the image plane is defined by these equations:

$$\frac{x}{a} = \frac{y}{b} = z - z_l \quad (5)$$

$$x = at \quad (6.a)$$

$$y = bt \quad (6.b)$$

$$z = z_l + t \quad (6.c)$$

Considering any point on the ground with coordinates x_g and y_g, the line passing through this point and lens point is defined by these equations:

$$\frac{x}{x_g} = \frac{y}{y_g} = \frac{z-z_l}{-z_l} \quad (7)$$

$$x = x_g t \quad (8.a)$$

$$y = y_g t \quad (8.b)$$

$$z = z_l - z_l t \quad (8.c)$$

By using these equations with the equation of the image plane, the coordinates of the projection of that ground point on image plane are defined as follows:

$$x_i = -\frac{z_l+d}{ax_g+by_g-z_l}x_g \quad (9.a)$$

$$y_i = -\frac{z_l+d}{ax_g+by_g-z_l}y_g \quad (9.b)$$

$$z_i = \frac{ax_g+by_g+d}{ax_g+by_g-z_l}z_l \quad (9.c)$$

The next step is to calculate the distances between that projection and the origin point of the image with respect to the lower edge of image defined by equations 3 and 4, these distances *h* and *v* can be estimated as follows:

$$h = \frac{C_{hx}x_g+C_{hy}y_g+C_h}{D_h(ax_g+by_g-z_l)} \quad (10.a)$$

$$v = \frac{C_{vx}x_g+C_{vy}y_g+C_v}{D_v(ax_g+by_g-z_l)} \quad (10.b)$$

where:

$$C_{hx} = [(a^2+1)\cos\theta + ab\sin\theta][ax_0 + z_l + d] + [(b^2+1)\sin\theta + ab\cos\theta]ay_0 \quad (11.a)$$

$$C_{hy} = [(a^2+1)\cos\theta + ab\sin\theta]bx_0 + [(b^2+1)\sin\theta + ab\cos\theta][by_0 + z_l + d] \tag{11.b}$$

$$C_h = -\{[(a^2+1)\cos\theta + ab\sin\theta]x_0 + [(b^2+1)\sin\theta + ab\cos\theta]y_0\}z_l \tag{11.c}$$

$$D_h = -\sqrt{(a^2+1)\cos^2\theta + 2\,ab\sin\theta\cos\theta + (b^2+1)\sin^2\theta} \tag{11.d}$$

$$C_{vx} = ay_0\cos\theta - (ax_0 + z_l + d)\sin\theta \tag{11.e}$$

$$C_{vy} = (by_0 + z_l + d)\cos\theta - bx_0\sin\theta \tag{11.f}$$

$$C_v = (x_0\sin\theta - y_0\cos\theta)z_l \tag{11.g}$$

$$D_v = -\sqrt{\frac{(a^2+1)\cos^2\theta + 2\,ab\sin\theta\cos\theta + (b^2+1)\sin^2\theta}{a^2+b^2+1}} \tag{11.h}$$

The final step is estimating the real coordinates using the distances from the image using these equations:

$$x_g = \frac{A + C_1 h + C_2 v}{T + C_5\,h + R\,C_6 v} z_l \tag{12.a}$$

$$y_g = \frac{B + C_3 h + C_4 v}{T + C_5\,h + R\,C_6 v} z_l \tag{12.b}$$

where:

$$A = \sqrt{1 + E^2\cos^2(\varphi - \theta)}\,x_0 \tag{13.a}$$

$$B = \sqrt{1 + E^2\cos^2(\varphi - \theta)}\,y_0 \tag{13.b}$$

$$T = \sqrt{1 + E^2\cos^2(\varphi - \theta)}\,(ax_0 + by_0 + z_l + d) = \sqrt{1 + E^2\cos^2(\varphi - \theta)}\,(z_l - z_0) \tag{13.c}$$

$$C_1 = \cos\theta \tag{13.d}$$

$$C_2 = \frac{1}{\sqrt{1+E^2}}\sin(\varphi - \theta)\cos\varphi - \sqrt{1+E^2}\cos(\varphi - \theta)\sin\varphi \tag{13.e}$$

$$C_3 = \sin\theta \tag{13.f}$$

$$C_4 = \frac{1}{\sqrt{1+E^2}}\sin(\varphi - \theta)\sin\varphi + \sqrt{1+E^2}\cos(\varphi - \theta)\cos\varphi \tag{13.g}$$

$$C_5 = E\cos(\varphi - \theta) \tag{13.h}$$

$$C_6 = \frac{E}{\sqrt{1+E^2}}\sin(\varphi - \theta) \tag{13.i}$$

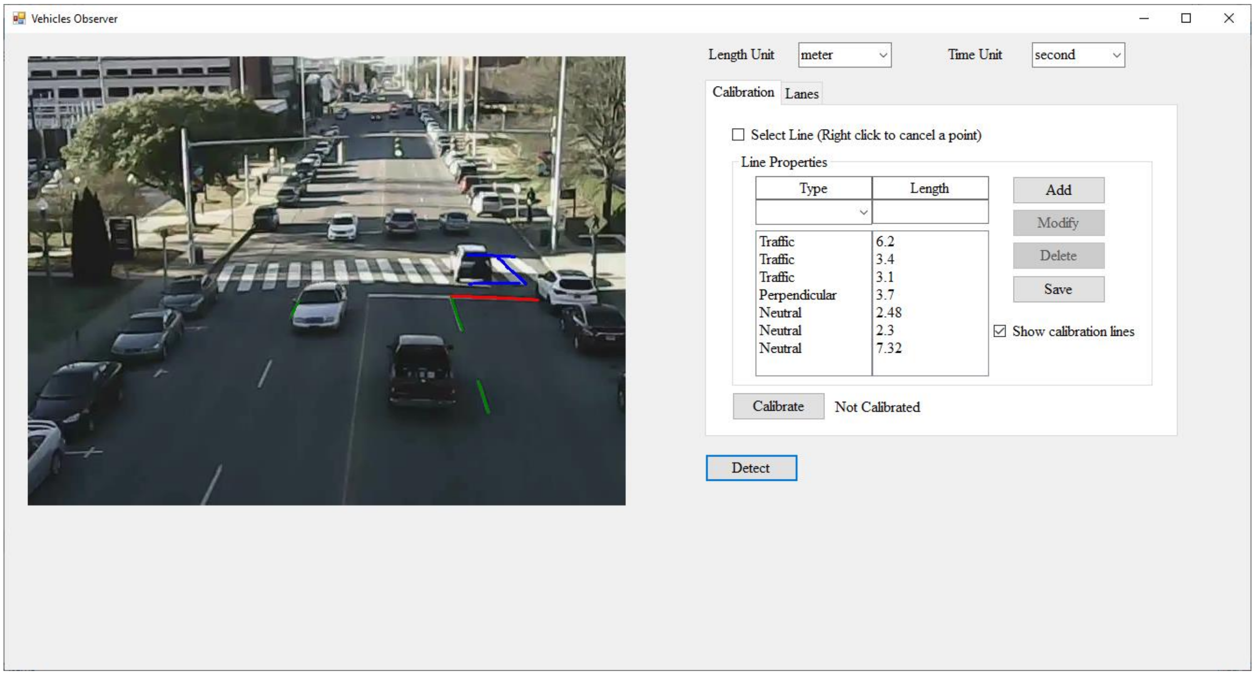
$$E = \sqrt{a^2 + b^2} \tag{13.j}$$

$$\tan\varphi = \frac{b}{a} \tag{13.k}$$

A, *B*, *T* and z_l are four independent coefficients while C_1, C_2, C_3, C_4, C_5, and C_6 are six coefficients that depend on the three angles that define the orientation of the camera.

5. CALIBRATION

These coefficients can be evaluated using a trial-and-error method with seven real inputs and their corresponding measurements on the image. Different length units can be used. The minimum inputs are six real lengths and one of them must have a known direction: traffic direction or perpendicular to it, or five real lengths with two of them in two perpendicular directions. It is recommended to have a traffic distance, y-direction, a perpendicular distance, x-direction, and three or more distances with no direction. Using C#, a program was made to do this calibration. In Figure 4, three lines in the traffic direction (green lines), a line perpendicular to the traffic direction (red line), and three non-directional lines (blue lines), are used for calibration. In Figure 5, two lines in the traffic direction and five non-directional lines are used. In Figure 6, a line in the traffic direction and six non-directional lines are used. However, there is a condition that must be considered when using two vertical or horizontal distances: they must not be on the same line. When calibration is done, it is displayed as “Calibrated,” the calibration lines can be hidden, and detection can start as illustrated in Figure 7.



Length Unit: meter

Time Unit: second

☐ Select Line (Right click to cancel a point)

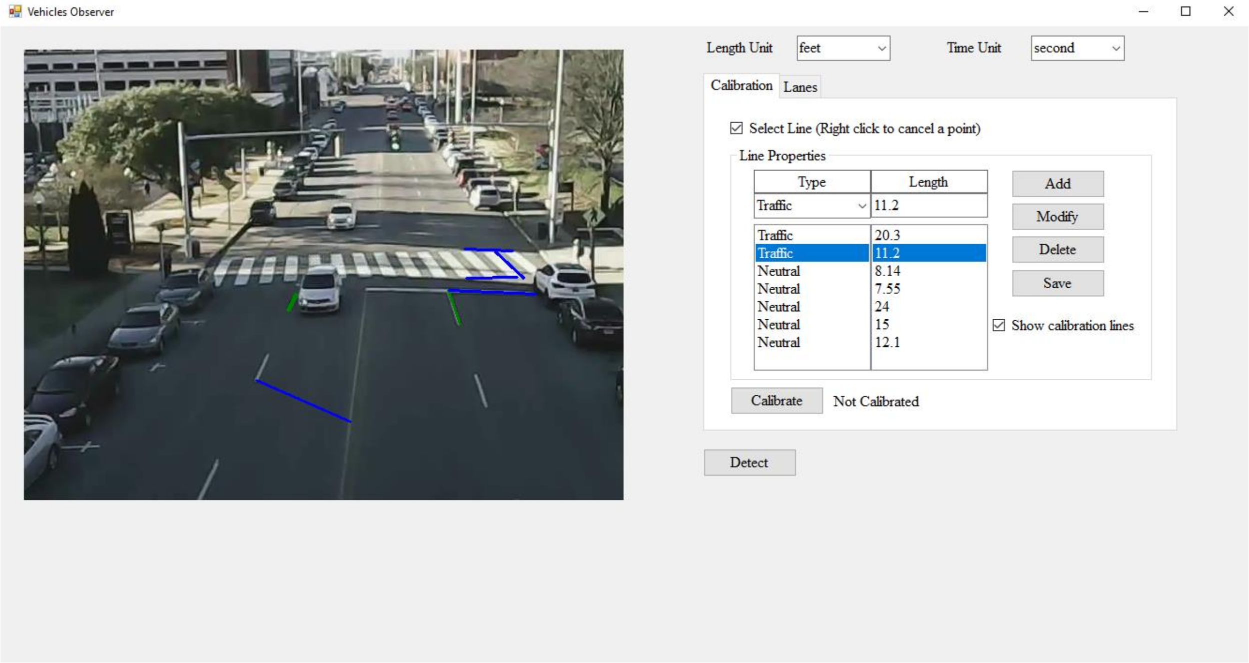
Line Properties

Type	Length
Traffic	6.2
Traffic	3.4
Traffic	3.1
Perpendicular	3.7
Neutral	2.48
Neutral	2.3
Neutral	7.32

☑ Show calibration lines

Not Calibrated

Figure 4. First set of lines used for calibration.



Vehicles Observer

Length Unit: feet

Time Unit: second

Calibration | Lanes

☑ Select Line (Right click to cancel a point)

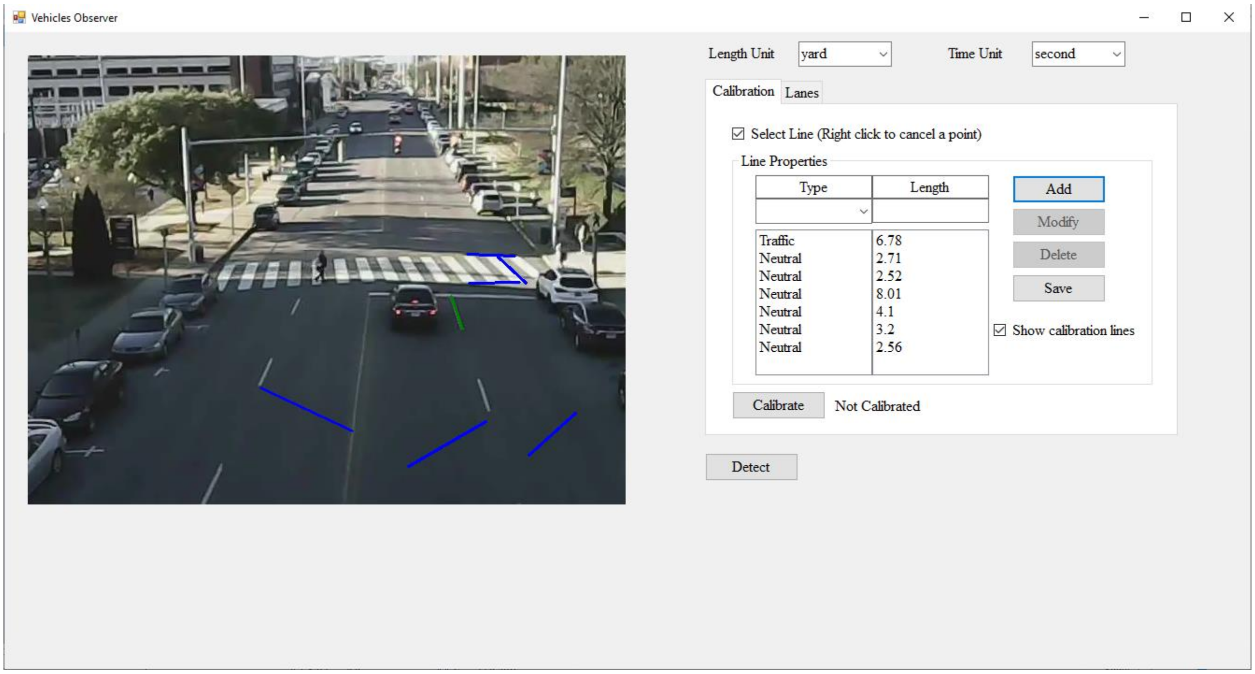
Line Properties

Type	Length
Traffic	11.2
Traffic	20.3
Traffic	11.2
Neutral	8.14
Neutral	7.55
Neutral	24
Neutral	15
Neutral	12.1

☑ Show calibration lines

Not Calibrated

Figure 5. Second set of lines used for calibration.



Vehicles Observer

Length Unit: yard

Time Unit: second

Calibration | Lanes

☑ Select Line (Right click to cancel a point)

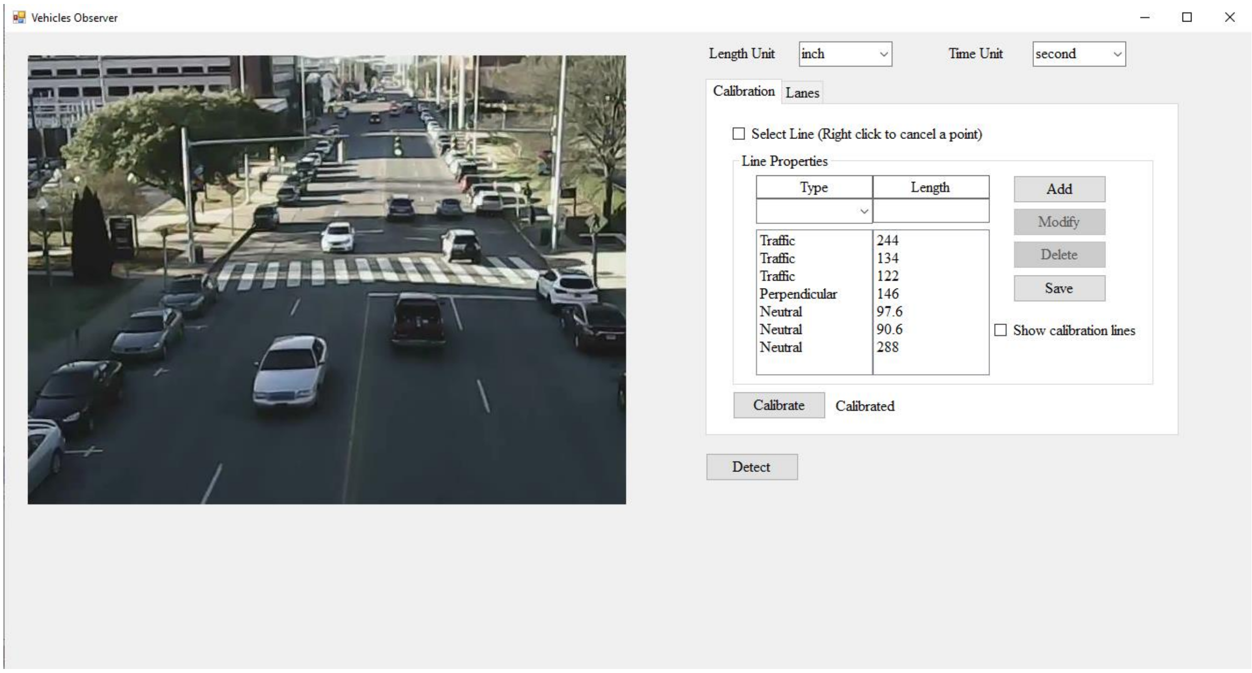
Line Properties

Type	Length
Traffic	6.78
Neutral	2.71
Neutral	2.52
Neutral	8.01
Neutral	4.1
Neutral	3.2
Neutral	2.56

☑ Show calibration lines

Not Calibrated

Figure 6. Third set of lines used for calibration.



Length Unit: inch

Time Unit: second

☐ Select Line (Right click to cancel a point)

Line Properties

Type	Length
Traffic	244
Traffic	134
Traffic	122
Perpendicular	146
Neutral	97.6
Neutral	90.6
Neutral	288

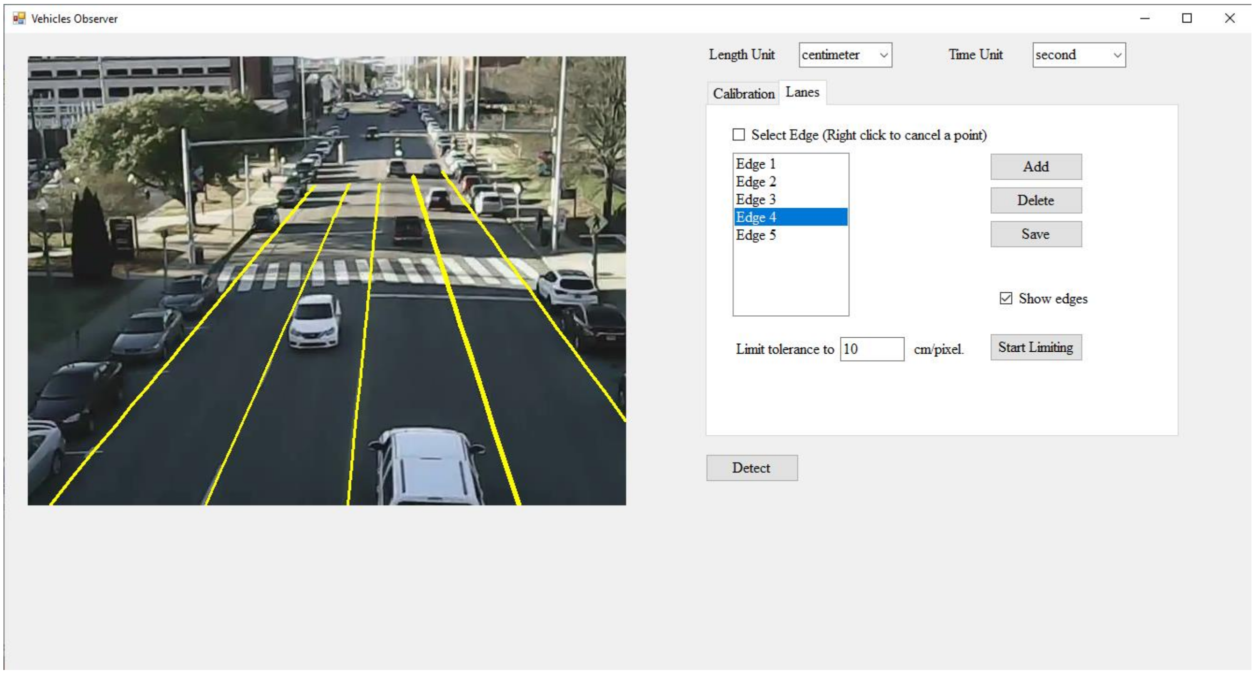
☐ Show calibration lines

Calibrated

Figure 7. Calibrated view.

6. DEFINING LANES

After calibration, lanes must be defined to separate vehicle flows by lane. The edge of the lane can be pinpointed by selecting two points on it as shown in Figure 8. To avoid wrong detections for far areas, a tolerance is used to limit the lanes and the detection to it. For example, if a tolerance of 10 cm/pixel is used, detection will be limited only to the image area where the change in one pixel causes a difference in real distance by 10 cm or less. The higher the tolerance is used, the more detection area is covered. However, this will lead to higher wrong measurements estimated at far areas as displayed in Figure 9.



Vehicles Observer

Length Unit centimeter

Time Unit second

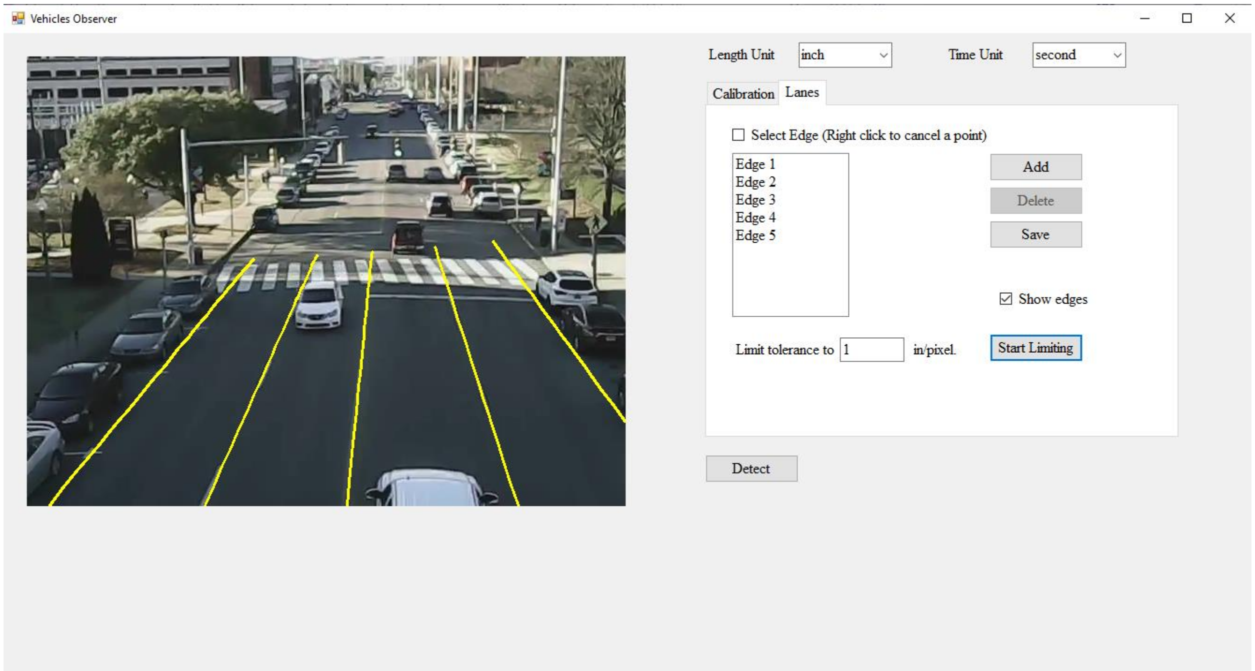
☐ Select Edge (Right click to cancel a point)

Edge 1
Edge 2
Edge 3
Edge 4
Edge 5

☑ Show edges

Limit tolerance to 10 cm/pixel.

Figure 8. Defining lanes.



Vehicles Observer

Length Unit inch

Time Unit second

☐ Select Edge (Right click to cancel a point)

Edge 1
Edge 2
Edge 3
Edge 4
Edge 5

☑ Show edges

Limit tolerance to 1 in/pixel.

Figure 9. Effect of changing tolerance.

7. VEHICLE DETECTION USING YOLOV3

Using YOLOv3, a good and accurate detection of vehicles can be achieved. This method applies a convolution neural network (CNN) feature recognition algorithm to the full image. The CNN divides the image into regions and predicts bounding boxes and probabilities for each region.

These bounding boxes are weighted by the predicted probabilities. These steps are displayed in Figure 10**Error! Reference source not found.**. This technique is a fast CNN technique as it analyzes the image only once and ignores small objects. It is more than 1000x faster than R-CNN and 100x faster than Fast R-CNN. YOLOv3 network has 53 convolutional layers, trained for 160 epochs, and its batch size is 64. When a vehicle is detected, the previously calibrated formula is used to estimate the location and speed of it. Then location and speed of the vehicle are displayed on it as shown in Figure 11. Location, speed, and acceleration of each vehicle are calculated every frame to keep tracking each one by predicting its next position. By dividing the scene to lanes, the vehicles can each be assigned to its moving lane. Hence back-to-back or front-to-front distances between every two successive vehicles in each lane can be obtained easily. Different speed signs indicate vehicles moving in different directions. Figure 12, Figure 13, and Figure 14 display some of the results of these detections. The results can be displayed in different distance and speed units and the lanes can be hidden. However, due to the moderate accuracy of detecting vehicles' bounding boxes, the accuracy of evaluating speed is not accurate although the accuracy of determining location is acceptable as will be shown later.

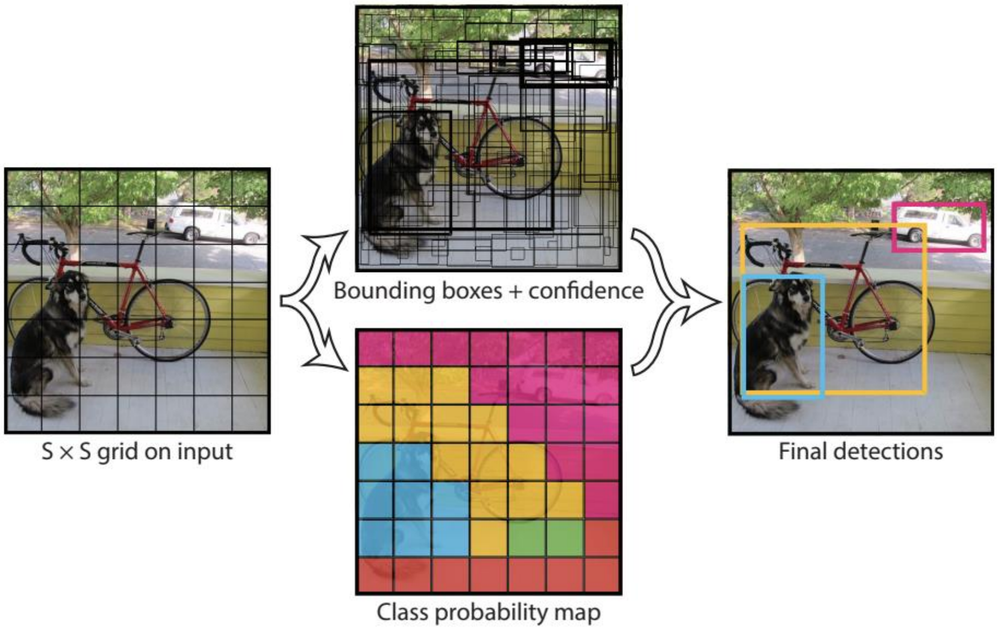
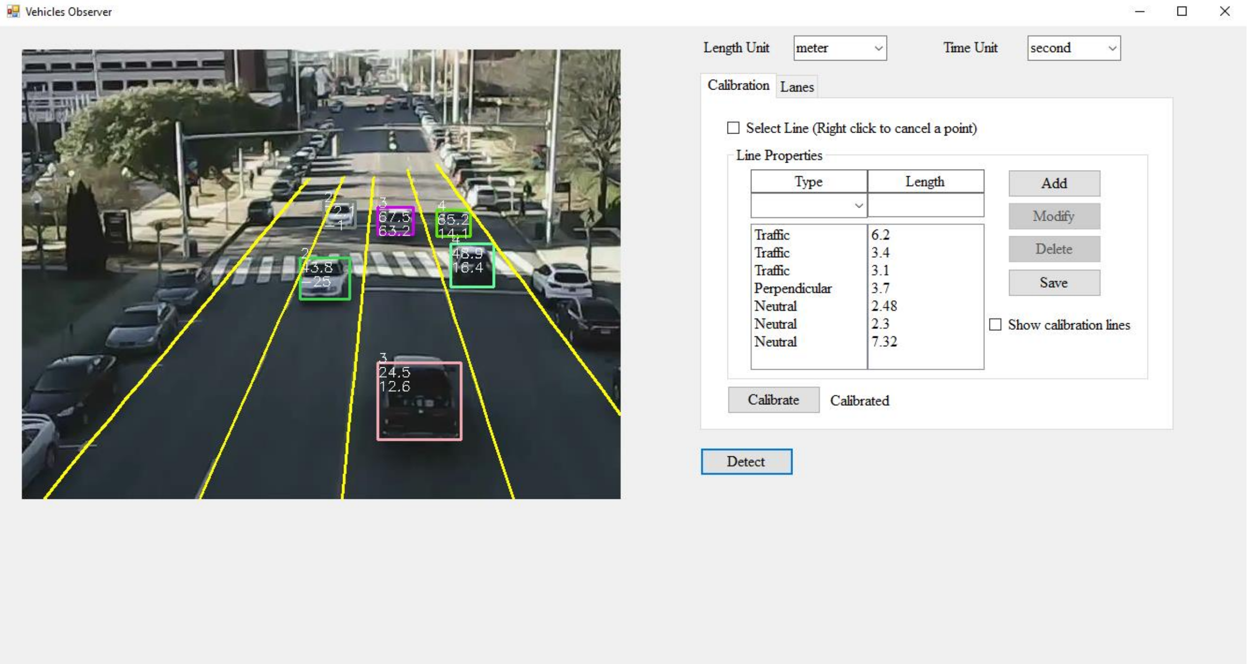


Figure 10. Detection using YOLOv3 (Redmon et al. 2016).



Length Unit meter Time Unit second

□ Select Line (Right click to cancel a point)

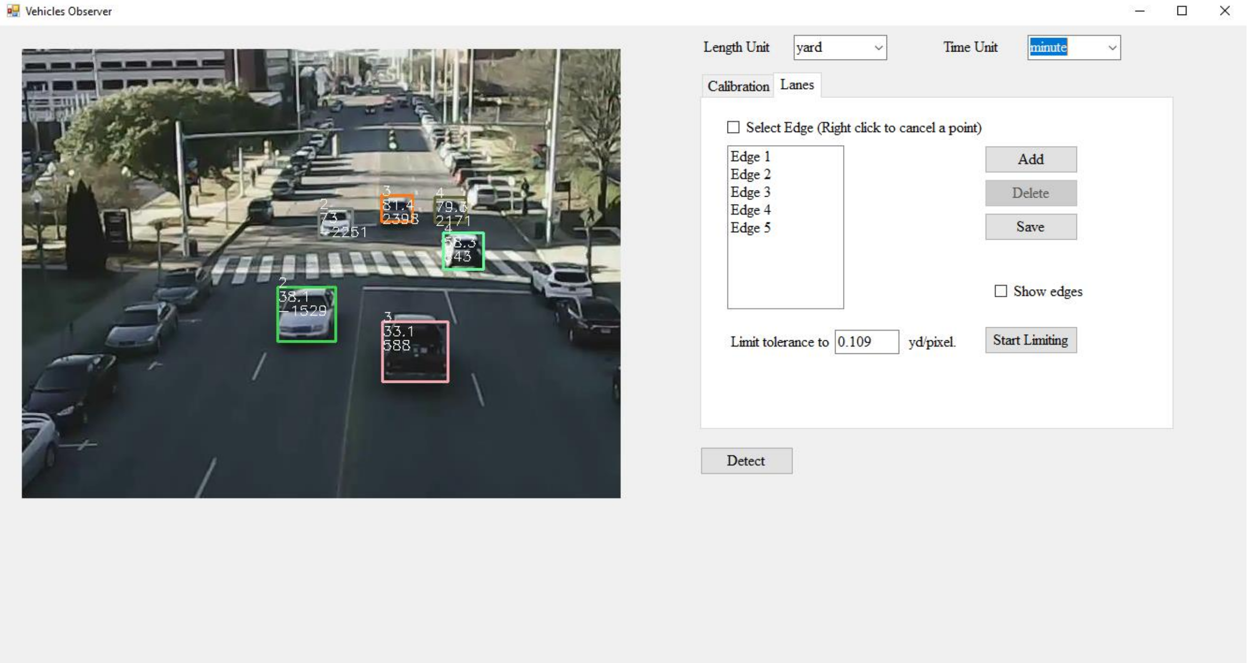
Line Properties

Type	Length
Traffic	6.2
Traffic	3.4
Traffic	3.1
Perpendicular	3.7
Neutral	2.48
Neutral	2.3
Neutral	7.32

□ Show calibration lines

Calibrated

Figure 11. Location and speed of vehicles in m/s.



Length Unit yard Time Unit minute

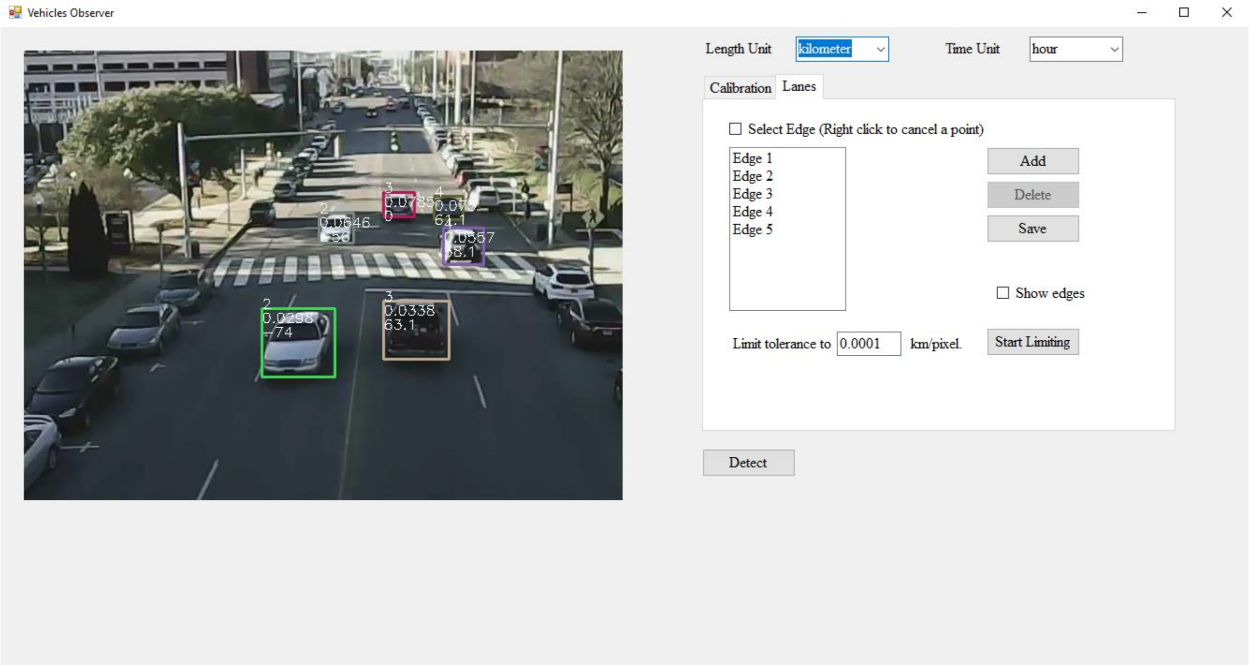
□ Select Edge (Right click to cancel a point)

Edge 1
Edge 2
Edge 3
Edge 4
Edge 5

□ Show edges

Limit tolerance to 0.109 yd/pixel.

Figure 12. Location and speed of vehicles in yd/min (Lanes hidden).



Vehicles Observer

Length Unit: kilometer Time Unit: hour

Calibration | Lanes

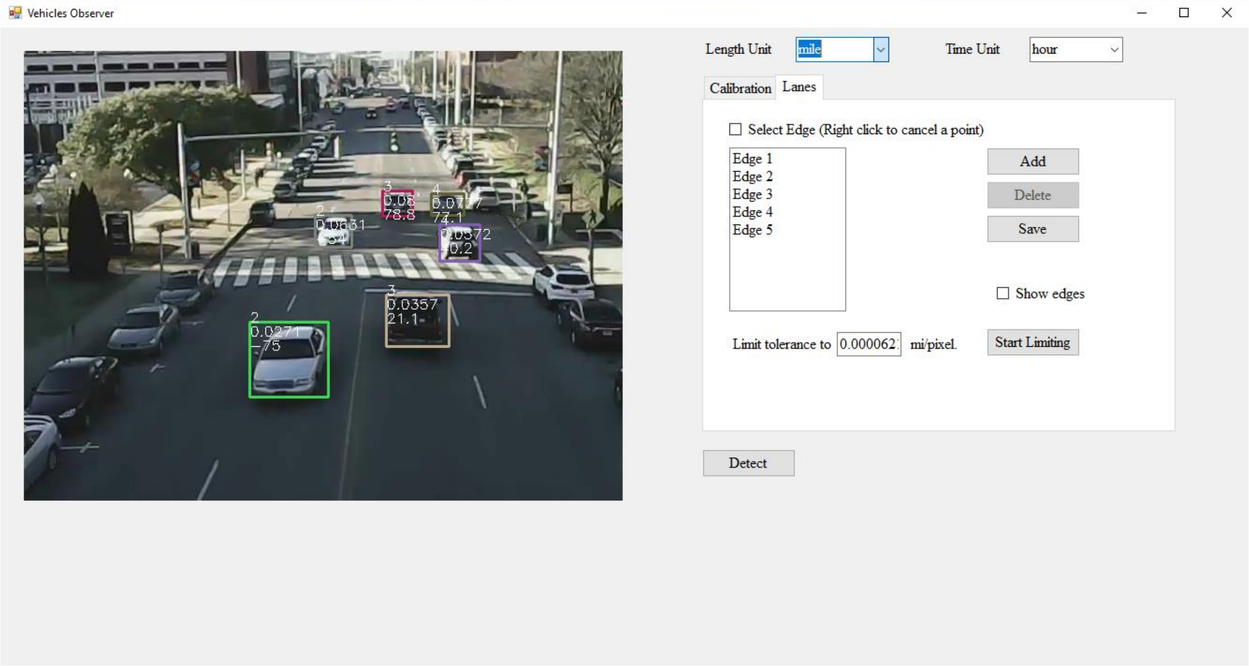
☐ Select Edge (Right click to cancel a point)

Edge 1
Edge 2
Edge 3
Edge 4
Edge 5

☐ Show edges

Limit tolerance to 0.0001 km/pixel.

Figure 13. Location and speed of vehicles in km/hr.



Vehicles Observer

Length Unit: mile Time Unit: hour

Calibration | Lanes

☐ Select Edge (Right click to cancel a point)

Edge 1
Edge 2
Edge 3
Edge 4
Edge 5

☐ Show edges

Limit tolerance to 0.000062 mi/pixel.

Figure 14. Location and speed of vehicles in mph.

To achieve faster detection, two neural networks were trained using only the five desired classes to be detected: car, bus, truck, motorbike, and bicycle, rather than the eighty classes of the original network provided by YOLO developers. Two different models were trained: a full model was 5 YOLO layers and a simple model with 3 YOLO layers only. Both models were trained using images containing objects of the desired classes. The two models were faster than the original network, however, the detection efficiency was less. While the original network almost detects every vehicle, the trained full network detects about 50 to 70% of them and the simple network about 30 to 40% only. Two videos were tested using the three networks. The first video was recorded using a drone camera for 14th St. S. between 9th Ave. S. and 10th Ave. S., Birmingham, AL, had a resolution of 640*480 and the second one, used by many developers of vehicle detection, was 1280*720. Both videos were 30 fps. Table 1 displays the efficiency of the networks and ratio of detected vehicles with both videos. It is obvious that the detection is better for higher resolution video, and it is poor for the tiny network. To inspect the average required detection time for each frame to evaluate the benefits gained in exchange of sacrificing efficiency, Intel® Core™ i5-6300U CPU @ 2.40 GHz with four cores was used. As shown in Table 2 the full network is two times faster than the original one and the tiny network is about 20 times faster than the original one. For both videos, the required time is very close. By using a more powerful processor, Intel® Xeon® CPU E5-2680 v3 @ 2.50 GHz with twelve cores the required time reduced drastically for all networks as shown in Table 3, 25% for original network and 50% for the two other networks. The interesting thing is that the required time for both the original and full networks are the same.

Table 1. Detection efficiency of YOLO networks used

Network Used	**640*480 video**	**1280*720 video**
Original YOLO	100%	100%
Trained full network	50.5%	68.4%
Trained tiny network	27.6%	39.2%

Table 2. Detection time for a frame of YOLO networks used (Intel® Core™)

Network Used	**640*480 video**	**1280*720 video**
Original YOLO	5.2 s	5.1 s
Trained full network	2.5 s	2.6 s
Trained tiny network	0.26 s	0.27 s

Table 3. Detection time for a frame of used YOLO networks used (Intel® Xeon®)

Network Used	640*480 video	1280*720 video
Original YOLO	1.3 s	1.3 s
Trained full network	1.3 s	1.3 s
Trained tiny network	0.13 s	0.13 s

It is better to use the original network when higher efficiency is required, low resolution video is used, or a strong processor is available. For other cases, trained networks will be more suitable. Unfortunately, even the fastest method cannot achieve real-time detection (33 ms for a frame) so some improved methods will be required for applications requiring near-real time detection.

8. USING GPU TO ACCELERATE DETECTION

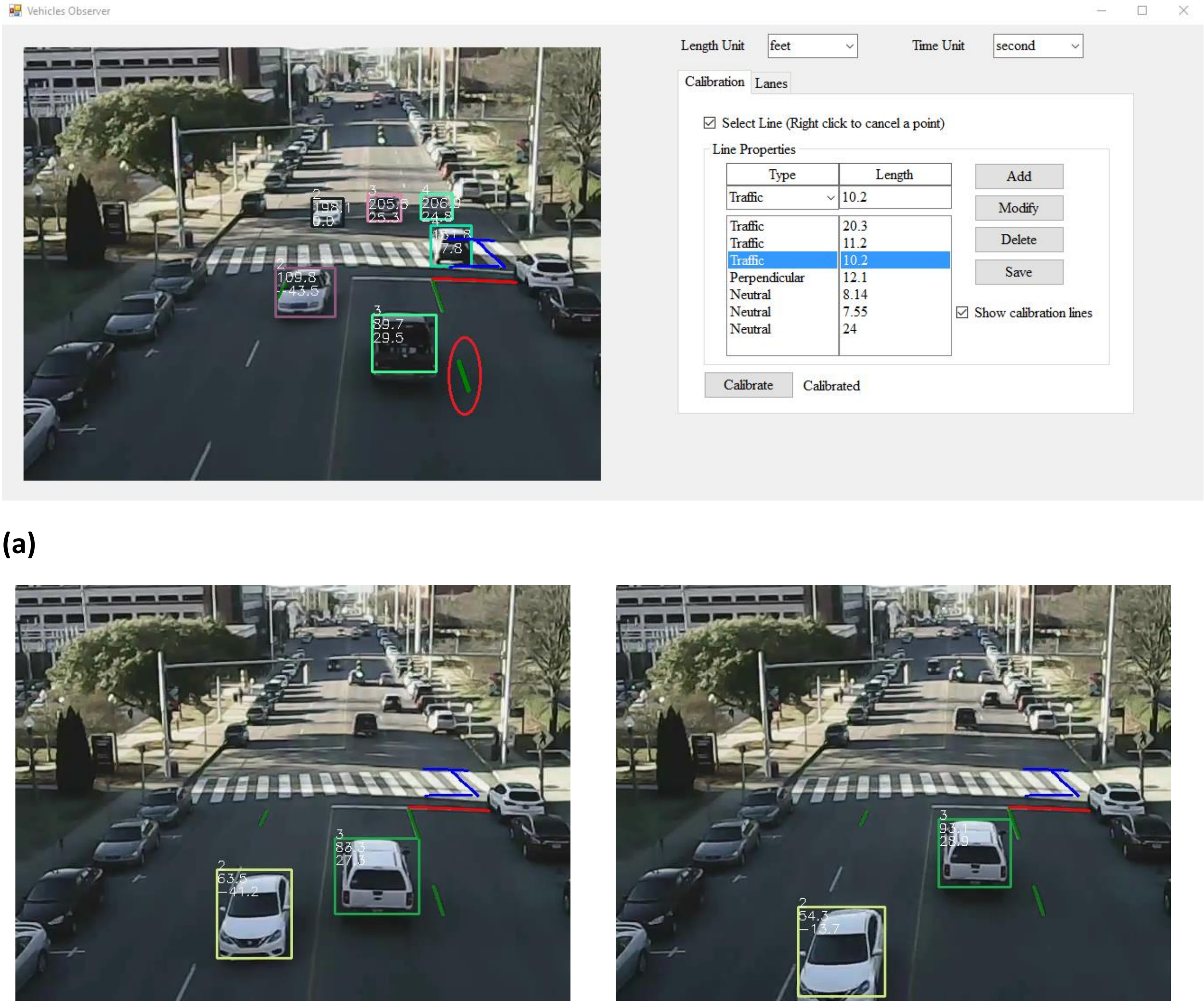
As seen in the previous section, increasing the number of processing cores led to less detection time. Therefore, GPU is used to accelerate detection. GPU has a higher number of smaller cores so it can carry out parallel processes, like our case, more efficiently. CUDA C++ on Nvidia GPU cards are used to enhance the performance of detection. Dynamic Link Library (DLL) was made using C++ then it was used inside the original C# program. Nvidia® GeForce™ 930M with 384 CUDA Cores was used to test the same networks and videos. Table 4 shows good improvement in detection speed, especially for the original and trained full network. Approaching real-time detection is possible with higher GPU cards like Nvidia® Tesla® P100 with 3584 CUDA Cores, which is under testing now.

Table 4. Detection time for a frame of YOLO networks used (Nvidia® GeForce™ 930M)

Network Used	640*480 video	1280*720 video
Original YOLO	0.46 s	0.48 s
Trained full network	0.36 s	0.40 s
Trained tiny network	0.10 s	0.12 s

9. VALIDATION OF ESTIMATED POSITIONS OF VEHICLES

To validate the used program, some results were inspected to confirm the accuracy of the evaluation of vehicles' positions. With the aid of some of the calibration lines whom lengths are known, it is possible to compare the position of a vehicle when it is at the start of the line and when it is at its end, then we can get the difference between these two positions and compare it with the length of the calibration line. By using the first calibration line shown in Figure 15 with length of 10.2 ft (a), a vehicle was at its start at location 83.3 ft (b) and at its end at location 93.1 feet (c), so the difference is 9.8 ft, another vehicle was at its start at location 82.4 feet (d) and at its end at location 93.2 ft (e), so the difference is 10.8 ft. By using the second calibration line shown in Figure 16. with length of 11.2 ft (a), a vehicle was at its start at location 133.2 ft (b) and at its end at location 121.4 ft (c), so the difference is 11.8 ft, another vehicle was at its start at location 133.7 ft (d) and at its end at location 123.1 ft (e), so the difference is 10.6 ft. From these results, it is obvious that the difference between the estimated length and the actual length does not exceed 6% which is an acceptable accuracy. More validation was intended using a vehicle of known speed, but this was postponed due to the pandemic COVID-19.



Length Unit: feet   Time Unit: second

☑ Select Line (Right click to cancel a point)

Line Properties

Type	Length
Traffic	10.2
Traffic	20.3
Traffic	11.2
Traffic	10.2
Perpendicular	12.1
Neutral	8.14
Neutral	7.55
Neutral	24

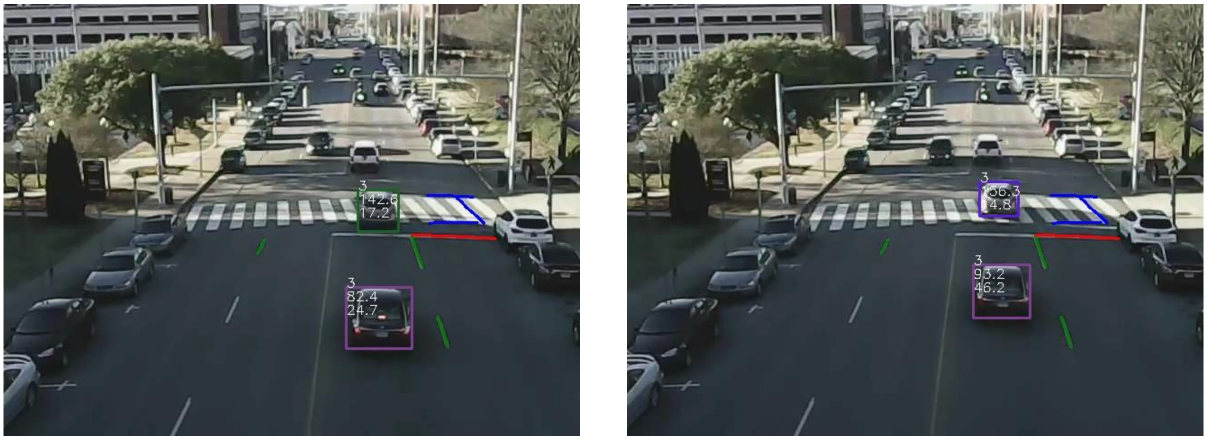
☑ Show calibration lines

Calibrated

(a)

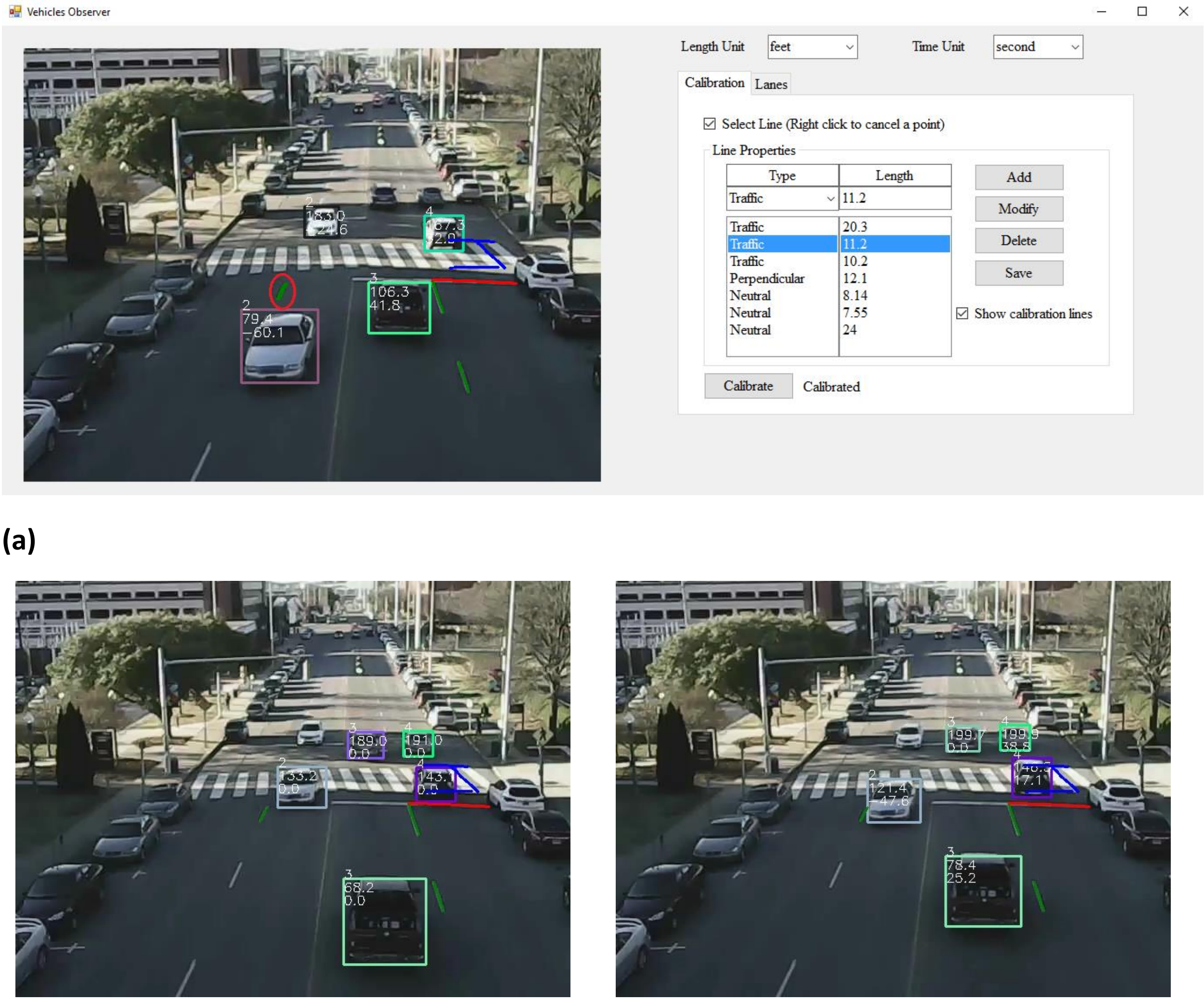
(b)

(c)



(d) **(e)**

Figure 15. Validation using the first calibration line.



Vehicles Observer

Length Unit: feet

Time Unit: second

Calibration | Lanes

☑ Select Line (Right click to cancel a point)

Line Properties

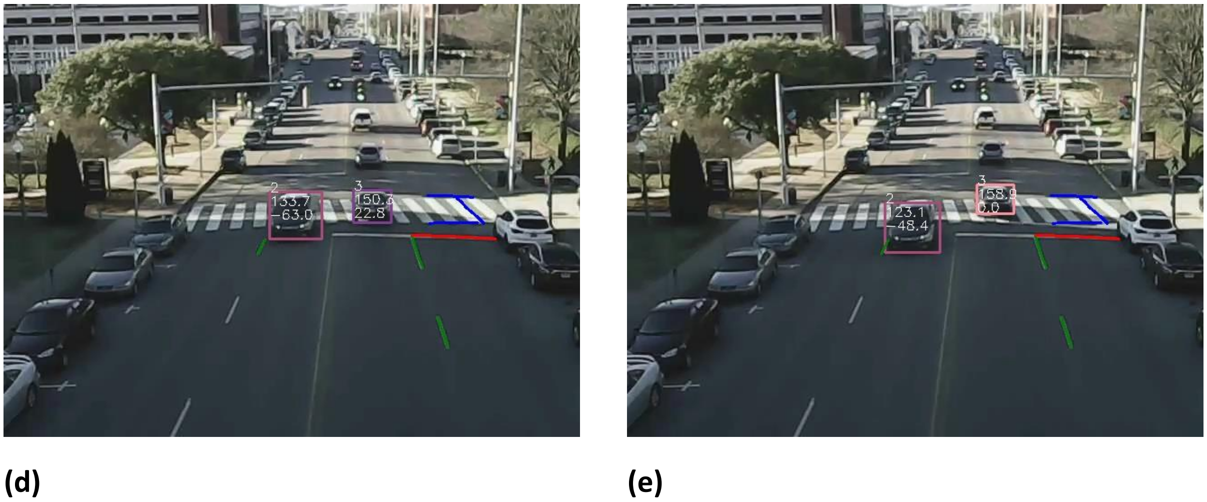
Type	Length
Traffic	11.2
Traffic	20.3
Traffic	11.2
Traffic	10.2
Perpendicular	12.1
Neutral	8.14
Neutral	7.55
Neutral	24

☑ Show calibration lines

Calibrate — Calibrated

(a)

(b) **(c)**



(d) **(e)**

Figure 16. Validation using the second calibration line.

10. DETECTION LIMITATIONS

The detection is limited to vehicles of considerable size inside the image. Therefore, multiple videos were tested from the City of Raleigh, North Carolina to estimate the minimum size of detection. Starting with the video of resolution of 640*480, the smallest detection bounding box was 12*8 pixels as shown in Figure 17. In the second video of resolution of 720*480 (captured for I-285 from GDOT camera 238 located near the midpoint between the Midvale Road underpass to the south and the Henderson Road overpass to the north) the smallest detection bounding box was 19*11 pixels as shown in Figure 18. In the third video of resolution of 1280*720, the smallest detection bounding box was 20*21 pixels as shown in Figure 19. In the last video of resolution of 1920*1080 using a drone flying over I-40 and I-440, the smallest detection bounding box was 13*14 pixels as shown in Figure 20. From these results, it was found that the bounding box of the vehicle must have the size of 100 pixels at least, and its smallest dimension must not be less than 8 pixels. Using the previous detection formula, in addition to the known dimensions of the smallest targeted vehicle, the properties of the camera, and the dimensions of the targeted area, the height and orientation of the camera can be estimated.

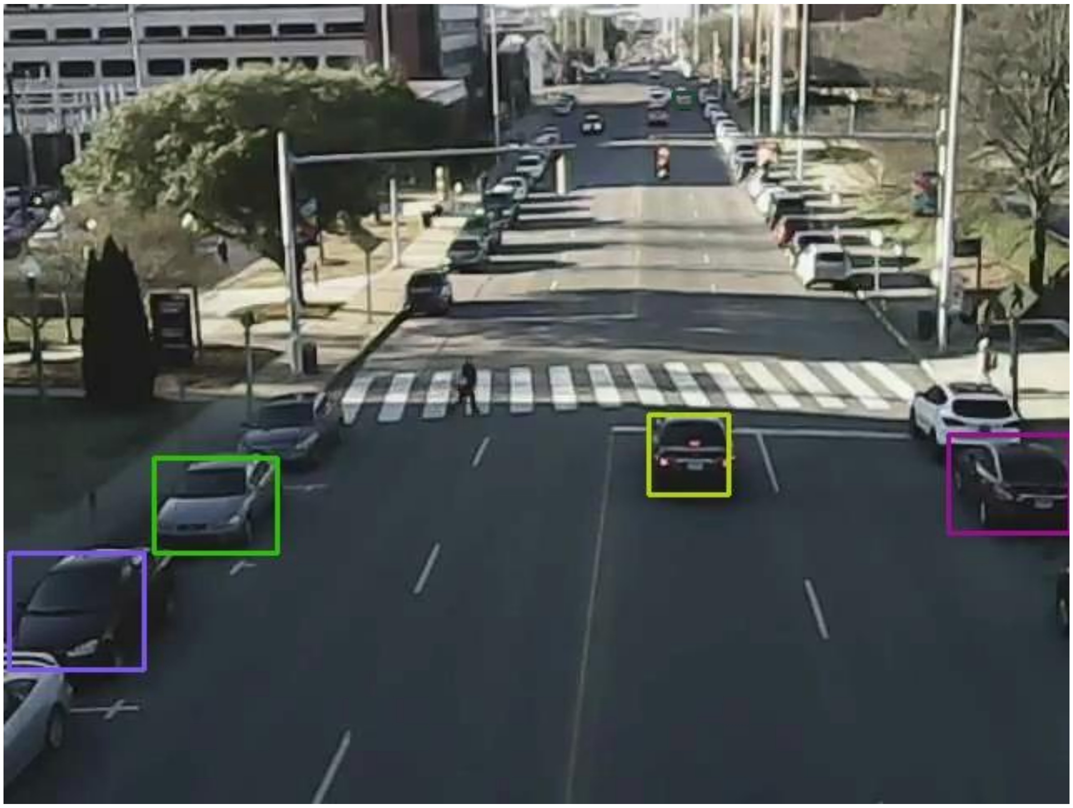


Figure 17. Smallest detection bounding box of 640*480 video.

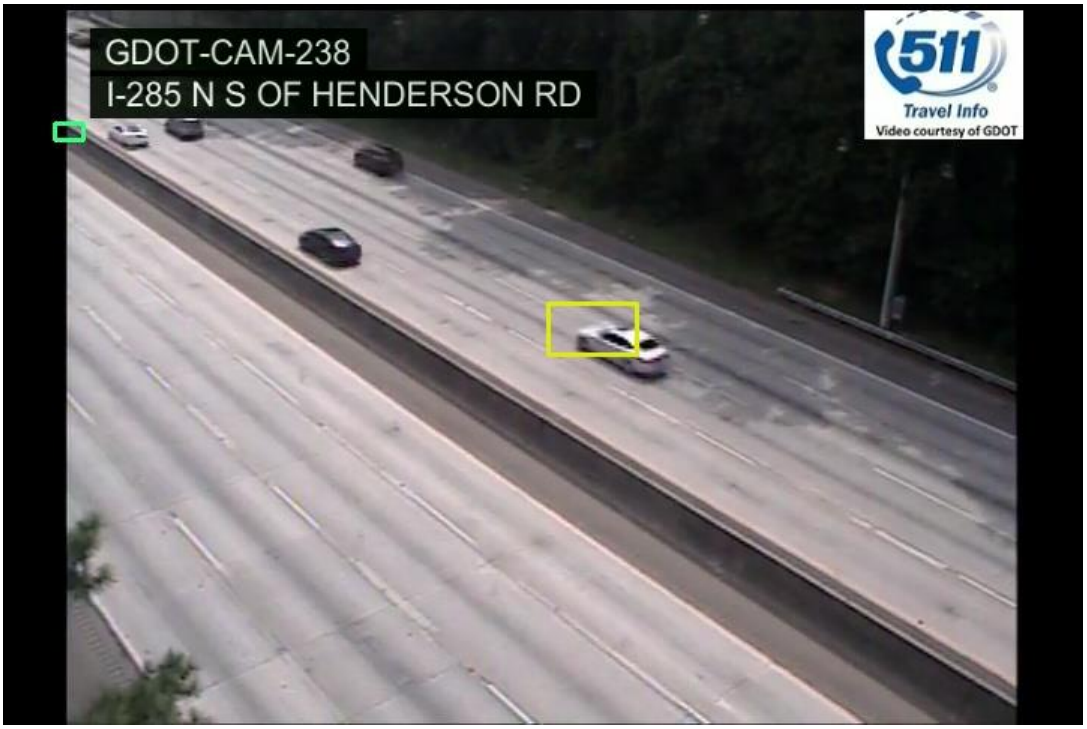


Figure 18. Smallest detection bounding box of 720*480 video.

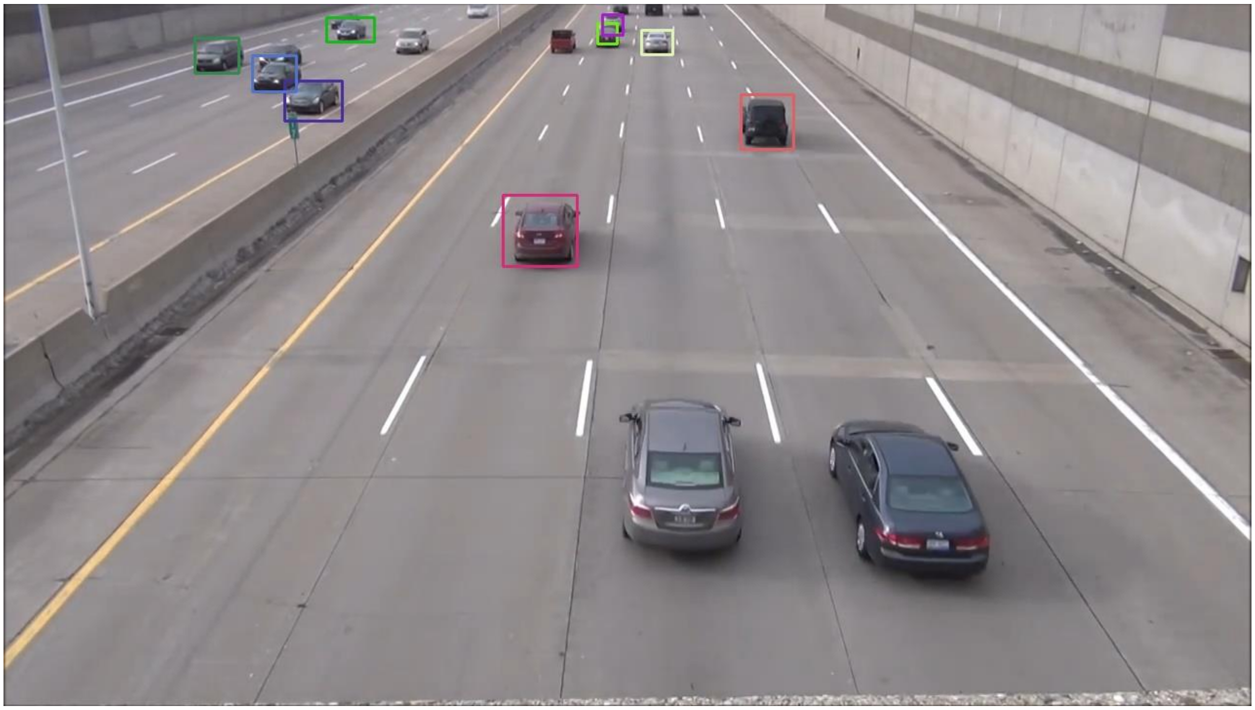


Figure 19. Smallest detection bounding box of 1280*720 video.

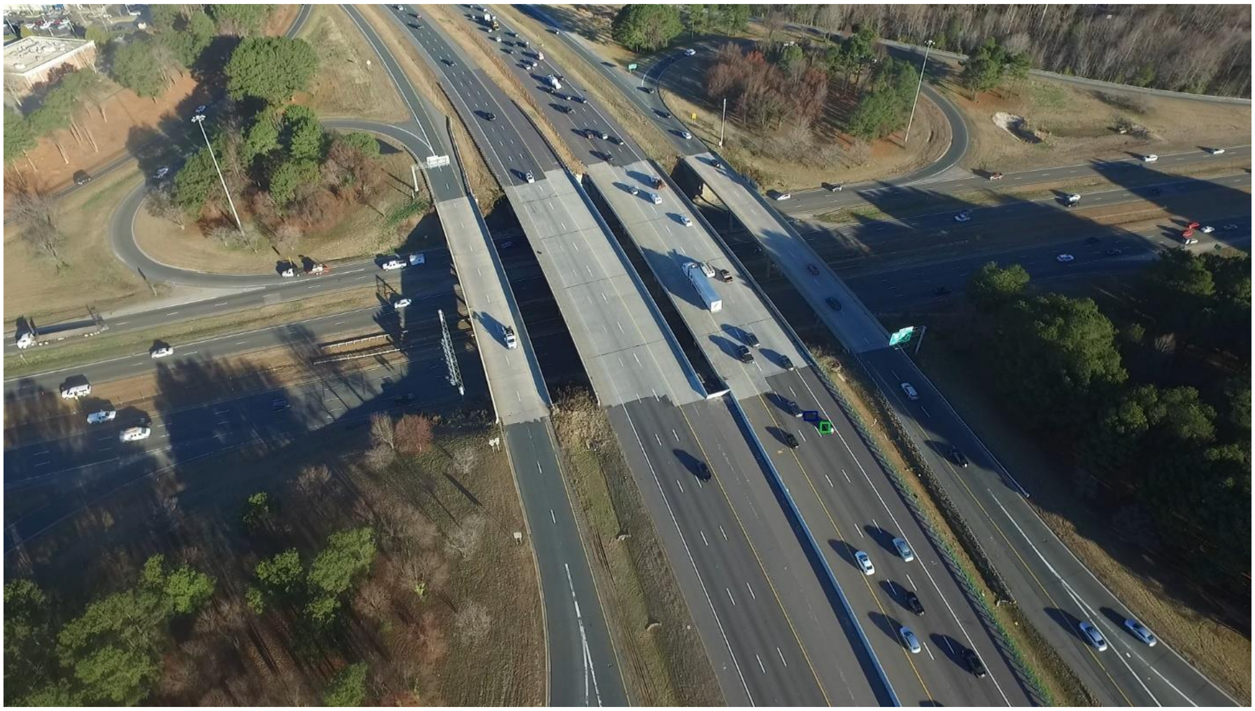


Figure 20. Smallest detection bounding box of 1920*1080 video.

11. CONGESTION DETECTION

The final goal was to estimate the congestion on the road from the positions of observed vehicles. Congestion is defined by exceeding the road capacity and is characterized by high vehicular density and vehicle speeds that are lower than the desired speeds. When a certain limit of vehicle/mile/lane is reached, then the road is considered congested. The limit value of 45 vehicle/mile/lane (28 vehicle/km/lane) which is typical for a regular roadway is considered as congestion in this case as shown in Figure 21.. To clarify this concept, an initial speed of 65 mph is considered for the vehicles to develop the congestion development graphs in Figure 22. In the beginning, the traffic keeps its maximum speed as long as vehicular density is low. After a while, this density increases, and when it reaches a certain limit, the traffic starts to slow down gradually to keep a safe distance between vehicles. Then, the speed continues to decrease as the vehicular density increases until it reaches 45 vehicle/mile. This distance, with the reduced speed and enough minimum safe trailing distance between vehicles (2-second rule) is considered the congestion limit.

Determining vehicular density from the observed segment of the road is as follows: each lane is considered individually, then the number of vehicles ‘*n’* is counted and the distance between the first and last car *‘d’* is calculated, and finally vehicular density *‘VD’* is estimated using Formula (14).

$$VD = \frac{n-1}{d} \tag{14}$$

In the case of only one car in a lane, vehicular density is considered to be equal 1. Figure 21 displays some of these results.

Figure 21 and Figure 22 use the *Highway Capacity Manual, Sixth Edition* traffic models with remotely sensed measurements superimposed. The data plotted on these figures are from permanently mounted side-fire radar stations. However, these data serve to illustrate how macroscopic traffic stream models can be used to evaluate traffic stream flow and detect the onset of congestion for the location monitored by the temporary drone-based array envisioned by this research. Traffic state evaluation would be further enhanced by using field calibrated traffic stream models. Therefore, for locations that are pre-selected for drone array deployment, it would be advisable to deploy the drones prior to the actual event requiring traffic management to validate the detection and calibrate the traffic stream models that will be used for traffic state assessment.

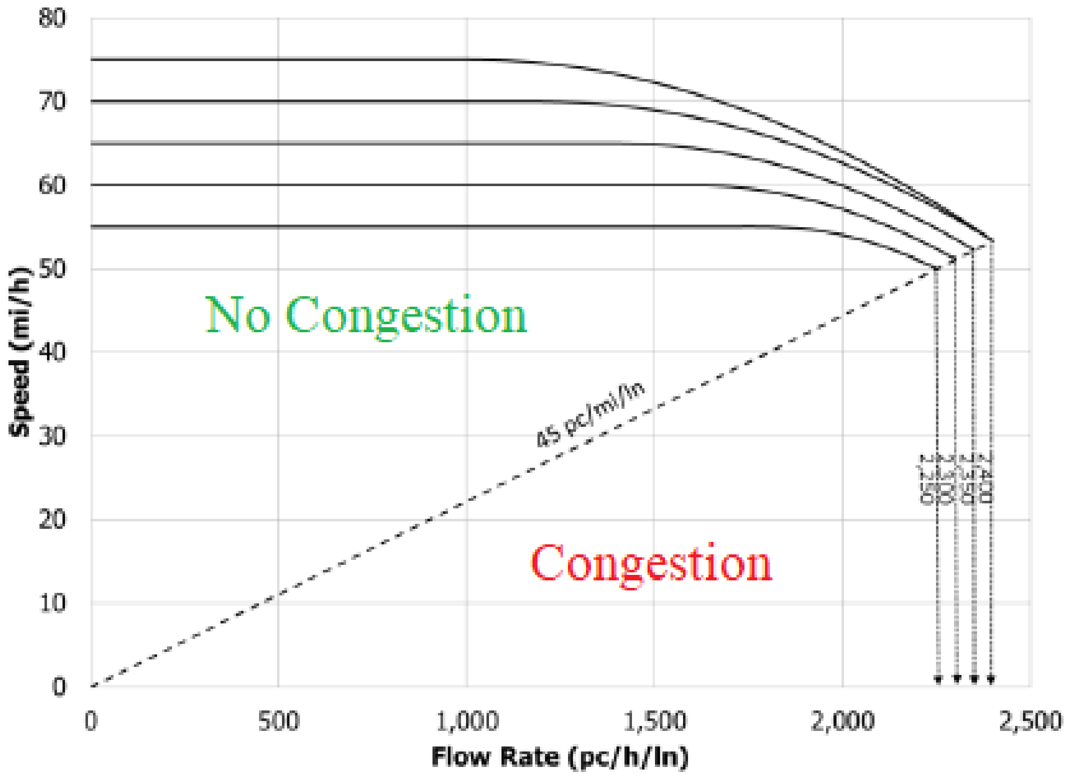


Figure 21. Speed-Flow curve.

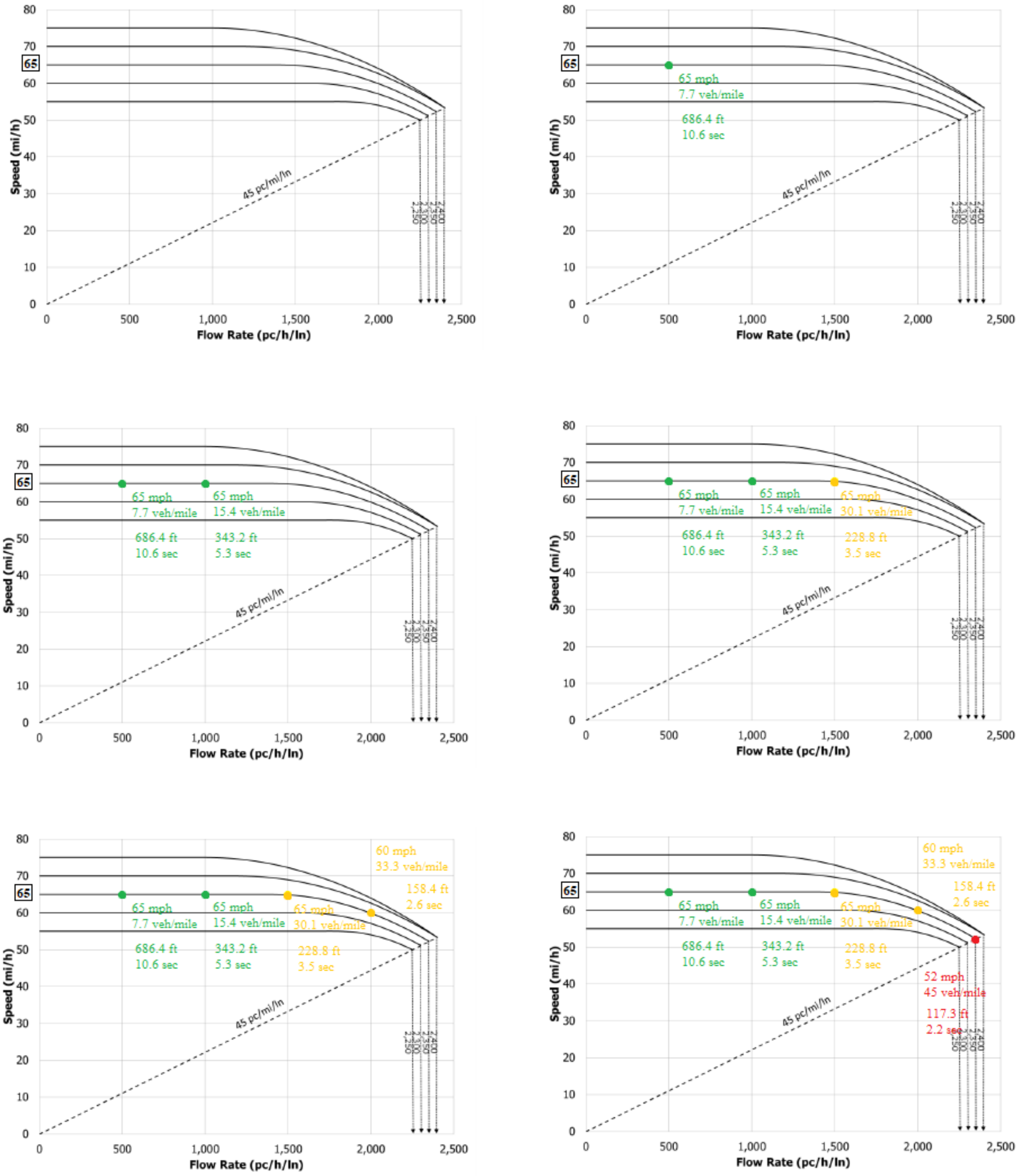
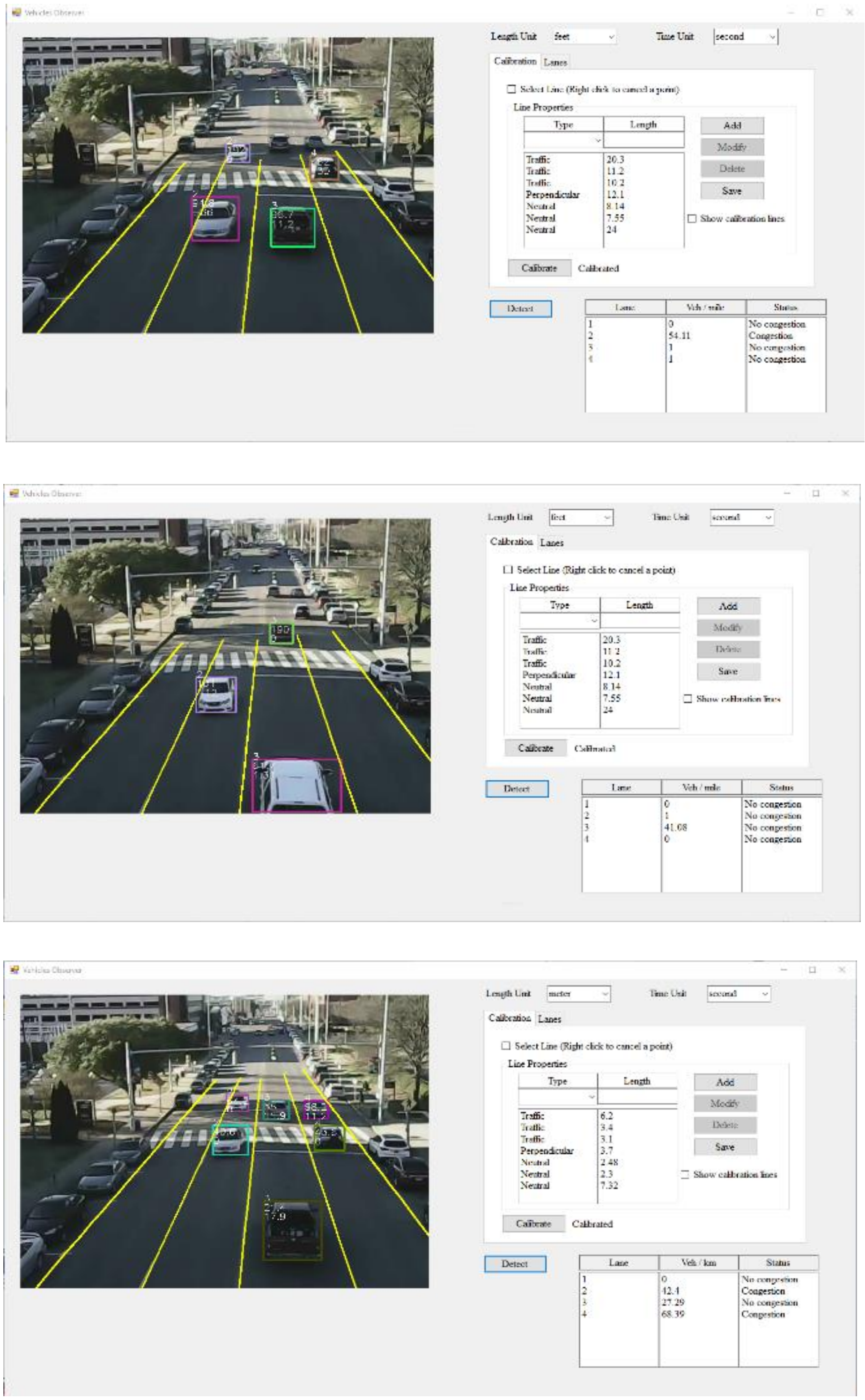


Figure 22. Congestion development with increased car numbers.



Length Unit feet Time Unit second

□ Select Line (Right click to cancel a point)

Type	Length
Traffic	20.3
Traffic	11.2
Traffic	10.2
Perpendicular	12.1
Neutral	8.14
Neutral	7.55
Neutral	24

□ Show calibration lines

Calibrated

Lane	Veh / mile	Status
1	0	No congestion
2	54.11	Congestion
3	1	No congestion
4	1	No congestion

Length Unit feet Time Unit second

□ Select Line (Right click to cancel a point)

Type	Length
Traffic	20.3
Traffic	11.2
Traffic	10.2
Perpendicular	12.1
Neutral	8.14
Neutral	7.55
Neutral	24

□ Show calibration lines

Calibrated

Lane	Veh / mile	Status
1	0	No congestion
2	1	No congestion
3	41.08	No congestion
4	0	No congestion

Length Unit meter Time Unit second

□ Select Line (Right click to cancel a point)

Type	Length
Traffic	6.2
Traffic	3.4
Traffic	3.1
Perpendicular	3.7
Neutral	2.48
Neutral	2.3
Neutral	7.32

□ Show calibration lines

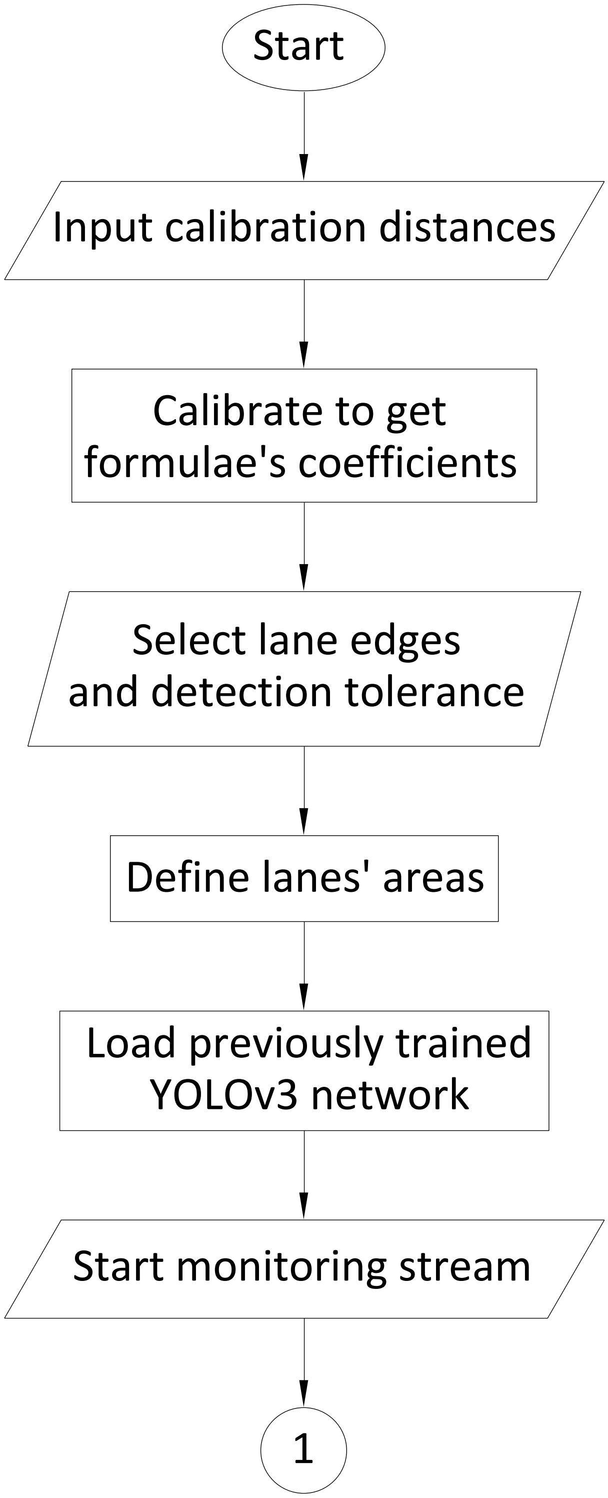
Calibrated

Lane	Veh / km	Status
1	0	No congestion
2	42.4	Congestion
3	27.29	No congestion
4	68.39	Congestion

Figure 23. Congestion estimation.

12. PROGRAM FLOWCHART

In Figure 24., the complete flowchart of the program is illustrated.


Start
Input calibration distances
Calibrate to get formulae's coefficients
Select lane edges and detection tolerance
Define lanes' areas
Load previously trained YOLOv3 network
Start monitoring stream
1

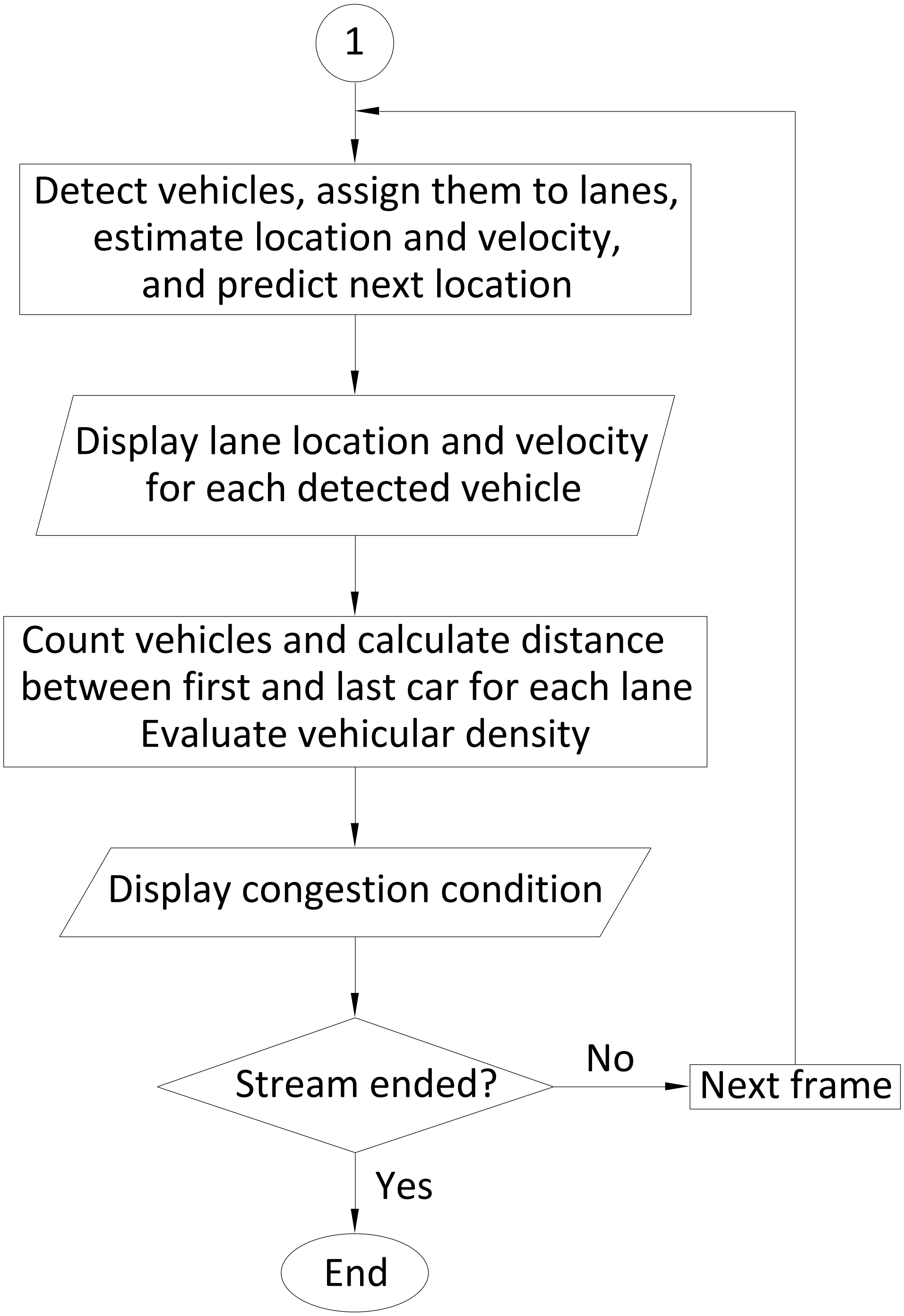

1
Detect vehicles, assign them to lanes, estimate location and velocity, and predict next location
Display lane location and velocity for each detected vehicle
Count vehicles and calculate distance between first and last car for each lane Evaluate vehicular density
Display congestion condition
Stream ended?
No
Next frame
Yes
End

Figure 24. Detection program flowchart.

13. CONCLUSIONS AND FUTURE WORK

Cameras can be calibrated to estimate locations from its images by using real measurements. Then, the calibrated camera can be used to detect vehicles using YOLOv3 and estimate their locations with good accuracy. In detection, different types of networks can be used based on the required efficiency, quality of the camera, and the available processors. After that, congestion can be evaluated based on the distance between vehicles. To accelerate detection, a GPU can be used.

For future improvements, YOLOv4 developed by Bochkovskiy et al. (2020) will be tested for better performance. It is described to be more efficient by its developers so it will be considered for future improvement. Also, different types of stronger processers will be used.

14. REFERENCES LIST

1. Redmon, J. and Farhadi, A., 2018. Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767.

2. Schoepflin, T. N., and Dailey, D. J., 2003. Dynamic camera calibration of roadside traffic management cameras for vehicle speed estimation. IEEE Transactions on Intelligent Transportation Systems, 4(2), 90-98.

3. Cathey, F. W., and Dailey, D. J.,2005. A novel technique to dynamically measure vehicle speed using uncalibrated roadway cameras. In IEEE Proceedings. Intelligent Vehicles Symposium, 2005. pp. 777-782.

4. Grammatikopoulos, L., George K., and Elli, P., 2005. Automatic estimation of vehicle speed from uncalibrated video sequences. International Symposium on Modern Technologies, Education and Professional Practice in Geodesy and Related Fields.

5. Dubská, M., Herout, A., Juránek, R., and Sochor, J., 2014. Fully automatic roadside camera calibration for traffic surveillance. IEEE Transactions on Intelligent Transportation Systems, 16(3), 1162-1171.

6. Dubská, M., Adam, H., and Jakub, S., 2014. Automatic camera calibration for traffic understanding. In BMVC, vol. 4, no. 6, p. 8.

7. Wang, K., Huang, H., Li, Y., and Wang, F. Y., 2007. Research on lane-marking line-based camera calibration. In 2007 IEEE International Conference on Vehicular Electronics and Safety, 2007. pp. 1-6.

8. Redmon, J., Divvala, S., Girshick, R., and Farhadi, A., 2016. You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 779-788.

9. Bochkovskiy, A., Wang, C. and Liao, H., 2020. YOLOv4: Optimal Speed and Accuracy of Object Detection. arXiv preprint arXiv:2004.10934.