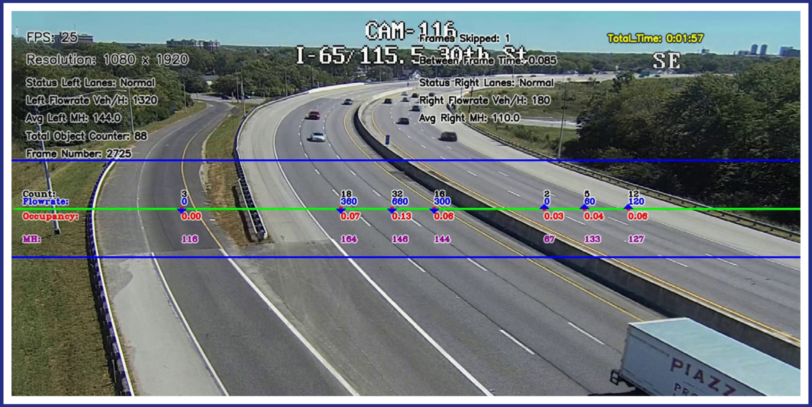
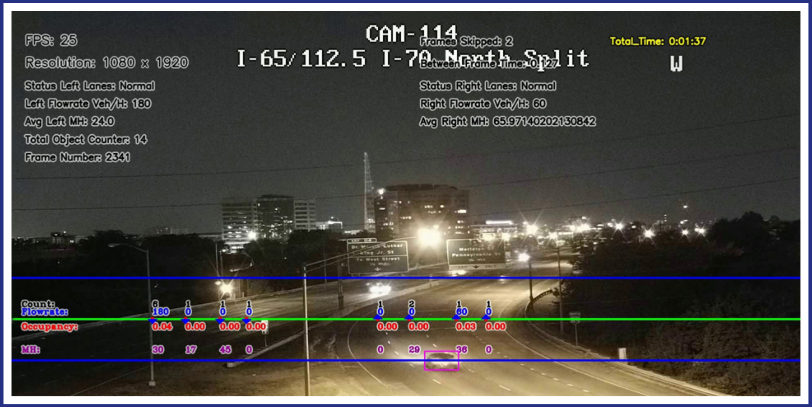
JOINT TRANSPORTATION RESEARCH PROGRAM
INDIANA DEPARTMENT OF TRANSPORTATION AND PURDUE UNIVERSITY

Road Condition Detection and Classification from Existing CCTV Feed





Stanley Chien, Yaobin Chen, Lauren Christopher,
Mei Qiu, Zhengming Ding

SPR-4436 • Report Number: FHWA/IN/JTRP-2022/02 • DOI: 10.5703/1288284317364

RECOMMENDED CITATION

Chien, S., Chen, Y., Christopher, L., Qiu, M., & Ding, Z. (2022). *Road condition detection and classification from existing CCTV feed* (Joint Transportation Research Program Publication No. FHWA/IN/JTRP-2022/02). West Lafayette, IN: Purdue University. https://doi.org/10.5703/1288284317364

AUTHORS

Stanley Chien, PhD
Elmore Associate Professor of Electrical and Computer Engineering
Associate Professor of Statistics Professor
School of Electrical and Computer Engineering
Indiana University-Purdue University Indianapolis
(317) 274-2760
schien@iupui.edu
Corresponding Author

Yaobin Chen, PhD
Director of TASI
Chancellor's Professor of Electrical and Computer Engineering
School of Engineering and Technology
Indiana University-Purdue University Indianapolis

Lauren Christopher, PhD
Associate Professor of Electrical and Computer Engineering
School of Electrical and Computer Engineering
Indiana University-Purdue University Indianapolis

Mei Qiu
Graduate Research Assistant
School of Electrical and Computer Engineering
Indiana University-Purdue University Indianapolis

Zhengming Ding, PhD
Assistant Professor of Computer and Information Technology
School of Engineering and Technology
Indiana University-Purdue University Indianapolis

JOINT TRANSPORTATION RESEARCH PROGRAM

NOTICE

ACKNOWLEDGMENTS

This work was supported by the Joint Transportation Research Program (JTRP) administered by the Indiana Department of Transportation (INDOT) and Purdue University. The authors would like to thank all members of the INDOT Study Advisory Committee (SAC), including Tim Wells, Project advisor (PA), Ed Cox Business owner (BO), Jim Sturdevant, for their guidance and advice throughout the project period. Special thanks go to Dr. Darcy Bullock, Director of JTRP, and his staff at Purdue University for their administrative support and service. The authors would also like to thank SAC members for their technical support and constructive suggestions.

TECHNICAL REPORT DOCUMENTATION PAGE

1. Report No. FHWA/IN/JTRP-2022/02	**2. Government Accession No.**	**3. Recipient's Catalog No.**
4. Title and Subtitle Road Condition Detection and Classification from Existing CCTV Feed		**5. Report Date** December 2021
		6. Performing Organization Code
7. Author(s) Stanley Chen, Yaobin Chen, Lauren Christopher, Mei Qiu, and Zhengming Ding		**8. Performing Organization Report No.** FHWA/IN/JTRP-2022/02
9. Performing Organization Name and Address Joint Transportation Research Program Hall for Discovery and Learning Research (DLR), Suite 204 207 S. Martin Jischke Drive West Lafayette, IN 47907		**10. Work Unit No.**
		11. Contract or Grant No. SPR-4436
12. Sponsoring Agency Name and Address Indiana Department of Transportation (SPR) State Office Building 100 North Senate Avenue Indianapolis, IN 46204		**13. Type of Report and Period Covered** Final Report
		14. Sponsoring Agency Code
15. Supplementary Notes Conducted in cooperation with the U.S. Department of Transportation, Federal Highway Administration.		

16. Abstract

The Indiana Department of Transportation (INDOT) has approximately 500 digital cameras along highways in populated areas of Indiana. These cameras are used to monitor traffic conditions around the clock, all year round. Currently, the videos from these cameras are observed one-by-one by human operators looking for traffic conditions and incidents. The main objective of this research was to develop an automatic, real-time system to monitor traffic conditions and detect incidents automatically.

The Transportation and Autonomous Systems Institute (TASI) of the Purdue School of Engineering and Technology at Indiana University-Purdue University Indianapolis (IUPUI) and the Traffic Management Center of INDOT developed a system that monitors the traffic conditions based on the INDOT CCTV video feeds. The proposed system performs traffic flow estimation, incident detection, and classification of vehicles involved in an incident.

The research team designed the system, including the hardware and software components added to the existing INDOT CCTV system; the relationship between the added system and the currently existing INDOT system; the database structure for traffic data extracted from the videos; and a user-friendly, web-based server for showing the incident locations automatically. The specific work in this project includes vehicle-detection, road boundary detection, lane detection, vehicle count over time, flow-rate detection, traffic condition detection, database development, web-based graphical user interface (GUI), and a hardware specification study. The preliminary prototype of some system components has been implemented in the *Development of Automated Incident Detection System Using Existing ATMS CCT* (SPR-4305).

17. Key Words incident detection, highway CCTV		**18. Distribution Statement** No restrictions. This document is available through the National Technical Information Service, Springfield, VA 22161.	
19. Security Classif. (of this report) Unclassified	**20. Security Classif. (of this page)** Unclassified	**21. No. of Pages** 31	**22. Price**

Form DOT F 1700.7 (8-72) Reproduction of completed page authorized

EXECUTIVE SUMMARY

Introduction

The Indiana Department of Transportation (INDOT) has approximately 500 digital cameras along highways in populated areas of Indiana. These cameras are used to monitor traffic conditions around the clock, all year round. Currently, the videos from these cameras are observed one by one by human operators looking for traffic conditions and incidents. It is time-consuming for the operators to scan through all the video data coming from all the cameras in real-time. The main objective of this research was to develop an automatic and real-time system to monitor traffic conditions and detect incidents automatically.

Findings

The Transportation and Autonomous Systems Institute (TASI) of the Purdue School of Engineering and Technology at Indiana University-Purdue University Indianapolis (IUPUI) and the Traffic Management Center of INDOT have worked together to conduct this 2-year research project to develop a system that monitors the traffic conditions based on the INDOT CCTV video feeds. The proposed system performs traffic flow estimation, incident detection, and classification of vehicles involved in an incident. The goal was to develop a system and prepare for future implementation.

The research team designed the system, which included the hardware and software components additions to the currently existing INDOT CCTV system; the relationship between the added system and the existing INDOT system; the database structure for traffic data extracted from the videos; and a user-friendly web-based server for showing the incident locations automatically.

Implementation

The preliminary prototype of some system components were implemented in *Development of Automated Incident Detection System Using Existing ATMS CCT* (SPR-4305). This 2-year project focuses on improving feature functionality and computation speed to make the program run in real-time. The specific work in this project included the following.

1. *Vehicle Detection*
   Automatic detection of vehicles is an essential part of this project. Various methods were tried and compared to improve the computation speed. The artificial intelligence method in YOLOv4 for vehicle detection currently generates the best results for daytime and lit conditions. Vehicle detection performance in an automatically selected region of interest reaches over 90% accuracy by YOLOv4.
2. *Road Boundary Detection*
   Since the aiming direction of each camera can change, it is not possible to specify, a priori, the road and lane locations on the video image. Therefore, the team used the tracking information of moving vehicles to determine road boundaries. Then only the vehicle detected on the road is considered for traffic condition checking. This helps to eliminate vehicle detection errors.
3. *Lane Detection*
   Since the aiming direction of each camera can change, it is not possible to specify, a priori, the lane locations on the video image. However, because most vehicles stay in their current lanes, the team developed lane detection and direction detection methods.
4. *Vehicle Count Over Time and Flow Rate Detection*
   There is a horizontal region on the video frames where the vehicle can be detected most accurately. The passing vehicles in each lane are counted all the time with timestamps. Each lane's flow rate (cars/hour) is derived based on vehicle counting and timestamps.
5. *Traffic Condition Detection*
   The traffic flow status is derived from the availability of the real-time flow rate on each detected lane. The traffic flow status is categorized as fast, normal, slow, and congested. The conditions for each camera-observed road segment are reported to the central database and displayed on the webpage for traffic operators.
6. *Database Development*
   This project uses a database as a central place to gather and distribute the information generated from camera videos and the incident detection results derived from sensory information in the database. The database also provides the information for the user interface. The database tables and their relationships have been redesigned and augmented. The database has been successfully implemented in MySQL and can be converted to PostgreSQL if needed.
7. *Web-Based Graphical User Interface (GUI)*
   A web-based Graphical User Interface (GUI) was developed with input and suggestions from INDOT. The GUI reads data from the database and generates user-interested information on the output display. The GUI displays four types of information: (1) location of all installed cameras on the Google Map, (2) all traffic incident locations (shown in red color) on the Google Map, (3) the real-time video of the focused incident location, and (4) all traffic information at the focused incident location. In addition, this GUI supports the selection of a focused traffic incident location among all incident locations.
8. *Hardware Specification Study*
   After implementing various system components, the research team performed experiments to run the system in different computer hardware and software environments. As a result, it is concluded that a mid-range GPU is sufficient for the real-time implementation of the system. On the other hand, the CPU power and the memory size are important factors (e.g., a $2,500 PC is sufficient to process one camera data in real-time).

The research team is currently testing the integrated systems for various camera and lighting conditions. In addition, the system is put into daily road traffic monitoring operation to identify the situations that need to be considered.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

Indiana Department of Transportation (INDOT) has installed over 500 cameras along highways in populated areas in Indiana. These cameras are used to observe and monitor traffic conditions around the clock, all year round. Currently, the videos from these cameras are observed individually by human operators. It is time-consuming for the operators to scan through all the real-time video data coming from cameras. In a 2018–2019 JTRP project, *Development of Automated Incident Detection System Using Existing ATMS CCTV*, we developed the framework of a prototype system to demonstrate the feasibility of automating the traffic monitoring system. The main objective of this research project is to continue the development of the automatic and real-time system to check the flow rate and traffic status for each lane on the road and make it ready for production.

2. SYSTEM ARCHITECTURE AND DEVELOPMENT

In the 2018–2019 JTRP project *Development of Automated Incident Detection System Using Existing ATMS CCTV*, we developed the system structure. The design was derived from the knowledge that INDOT currently uses hundreds of cameras all over the state of Indiana. Figure 2.1 shows the overall system structure for the entire system that is implemented. The entire system had three subsystems. The first subsystem is the Image Processing portion of the implementation running on the field computers. The second subsystem is the webserver and database server implemented on the Central Computer. The third subsystem is the Graphical User Interface and the Web Server.

In the 2019–2021 JTRP project *Road Condition Detection and Classification from Existing CCTV Feed*, we further developed the system. We made it run in real-time and cover more environments and lighting conditions. The same system structure defined in the 2018–2019 JTRP project is continuously used in this project. The current system is explained in the following sections.

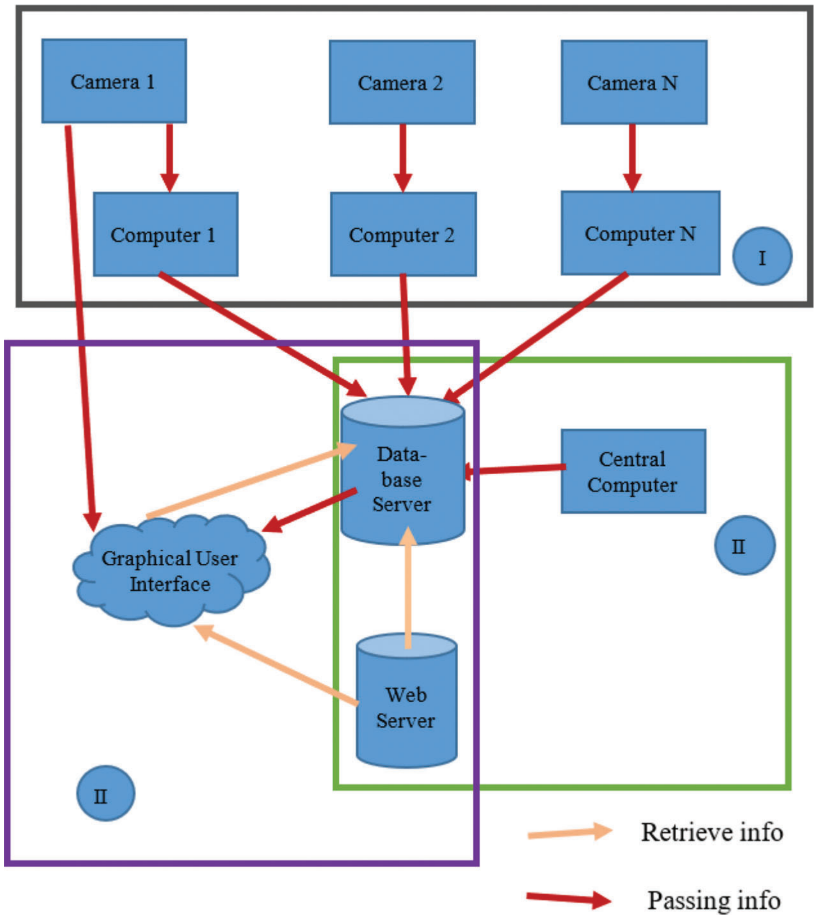


Figure 2.1 Overall system structure.

3. FIELD COMPUTER OPERATION

The incident detection for each camera is a program on a field computer. The input into computers is the raw video data from the cameras. INDOT provides the IP address of over 400 cameras on the road. The control of camera/Pan_Tilt_Zoom is manual and not implemented for computer control in the INDOT system.

Since the cameras in the INDOT system are purchased over many years in multiple batches, the frame per second rate and camera video resolution vary significantly. The flow rate varies from 6 to 60 fps. The resolution varies from 640×480 to 1920×1080. The frame rate setup of the cameras is also affected by the network speed at the installation location and method. Therefore, the software needs to adapt to all possible frame rates and resolutions of camera videos.

The outputs of each computer are real-time road condition data abstracted from video streams, which includes the following.

1. Number of lanes and lane locations
2. Traffic direction of each lane
3. The vehicle flow rate on each lane
4. Percentage of each type of vehicle on each lane
5. The traffic conditions

This output is inserted into a database every 30 seconds. The structure of the incident detection program is depicted in Figure 3.1. The program has two main parts, one is a one-time environment learning, and the other is repeated incident detection.

The environment learning part has the following modules.

1. Road boundary detection
2. Vehicle detection (whole frame)
3. Reference line and region of interest (ROI) determination
4. Lane center at reference line
5. Lane detection based on vehicle clustering
6. Lane direction detection
7. Relationship of the pixel size to average vehicle size
8. Camera direction detection
9. Lane boundaries determination
10. Database interface
11. Camera viewing angle change detection

The real-time incident detection part has the following modules.

1. Video rate-computer speed synchronization
2. Vehicle detection
3. Filter out the unrelated vehicles
4. Vehicle tracking
5. Lane vehicle ID mapping
6. Vehicle count
7. Calculate vehicle flow rate
8. Calculate vehicle speed
9. Incidents detection
10. Camera direction change detection
11. Equipment error detection and reporting
12. Database interface
13. System diagnostics

3.1 Environment Learning

Environment learning aims to find lanes, lane boundaries, lane directions, and the region where the vehicles can be detected and traced accurately. The environment needs to be relearned each time after the camera viewing angle or zoom level is changed. The advantage of this learning method is that the lane can be detected for any camera directly facing the road. It indeed takes some time to relearn the road with a new camera direction. However, the camera direction is constant most times unless the operator changes it. Suppose the operator is changing the camera direction. In that case, the operator is looking for the traffic condition, so that the automated traffic condition tracking by the computer is not essential.

3.1.1 Road Boundary Detection

Road boundary detection aims to mask out the non-road part of the image to reduce vehicle detection and tracking errors. The input for road boundary detection is a raw RGB stream (Figure 3.2) of N consecutive image frames starting from the first frame obtained from the camera whenever the camera changes its physical orientation (i.e., repositions itself). (The bigger the number N, the better the quality of output will be. However, we must also consider the time complexity of such an operation. For example, in our current program, N = 500.) The output is a greyscale image, which darkens the areas that are not the road.

The algorithm used for road boundary detection is a traditional image processing method as described as follows.

A. *Road Mask Generation*

1. For each frame *i* from 1 to N: Use an OpenCV:BackgroundSubtractorMOG object to obtain the foreground in a frame, *i*, in its greyscale equivalent (Figure 3.3). For *i* = 1, the *0*th image frame is black. The *i*th image subtracts (*i*-1)th raw (greyscale) image frame.
2. For each output frame in Step 1: Apply the OpenCV: blur function with pre-specified parameters (a kernel size of 10 in our program) to obtain its filtered equivalent (Figure 3.4).
3. Logically OR all filtered images obtained in Step 2 with each other, thus obtain a single greyscale image, which depicts the detected roads and other noisy shapes (Figure 3.5).

B. *Contour Detection and Noise Removal*

Contour is a curve joining all the contiguous points along the boundary of an area that has the same color or intensity. Here we are interested in using the contour to describe the boundary of roads.

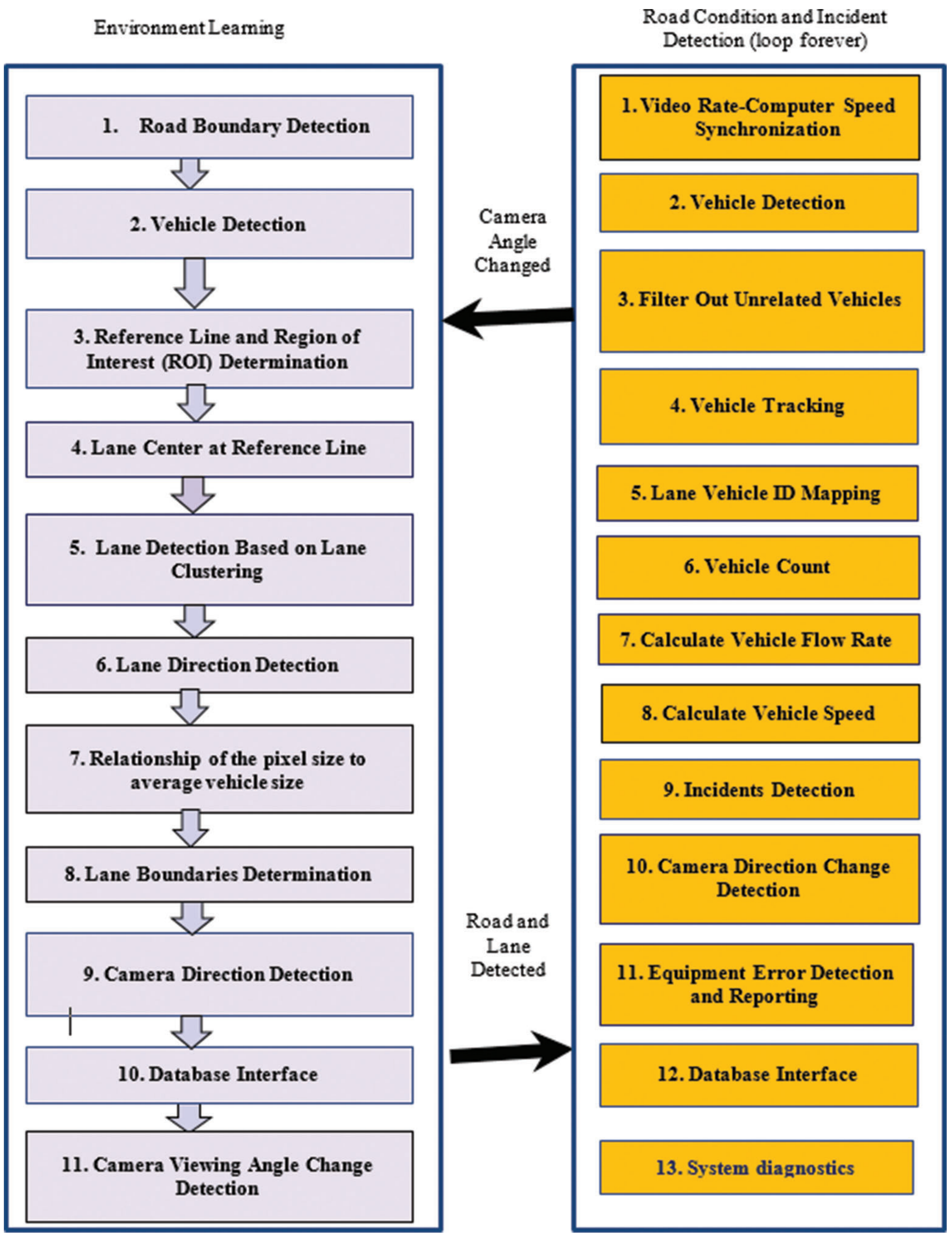

Environment Learning
1. Road Boundary Detection
2. Vehicle Detection
3. Reference Line and Region of Interest (ROI) Determination
4. Lane Center at Reference Line
5. Lane Detection Based on Lane Clustering
6. Lane Direction Detection
7. Relationship of the pixel size to average vehicle size
8. Lane Boundaries Determination
9. Camera Direction Detection
10. Database Interface
11. Camera Viewing Angle Change Detection
Camera Angle Changed
Road and Lane Detected
Road Condition and Incident Detection (loop forever)
1. Video Rate-Computer Speed Synchronization
2. Vehicle Detection
3. Filter Out Unrelated Vehicles
4. Vehicle Tracking
5. Lane Vehicle ID Mapping
6. Vehicle Count
7. Calculate Vehicle Flow Rate
8. Calculate Vehicle Speed
9. Incidents Detection
10. Camera Direction Change Detection
11. Equipment Error Detection and Reporting
12. Database Interface
13. System diagnostics

Figure 3.1 Software components.

The input is the final road mask image, as shown in Figure 3.5. The output is a contour describes by an array of points (x, y coordinates) of the boundary of the roads and an image showing only the road without the noise from the input image.
The algorithm of *Contour Detection and Noise Removal* is as follows.

1. Use the OpenCV library function, *findcontours*, with the input image as an argument to obtain the contours (as arrays of points) of all white shapes in the image. Do not use any chaining approximation while finding the contour. Every single border point must be obtained. This is essential for later ROI findings. Note that a contour traces white points to circle either a white area or a black area. Therefore, image boundary cannot be part of a contour. Figure 3.6 is an illustration.
2. Calculate the area of each contour and find the maximum area among all contours.
3. Delete those contours from the list of contours with an area less than a specified percentage (e.g., 10% currently used in our program, but may need to be changed after testing) of the maximum found in Step 2.
4. Generate a black/white image with the same dimensions as the original input image. In this image, fill

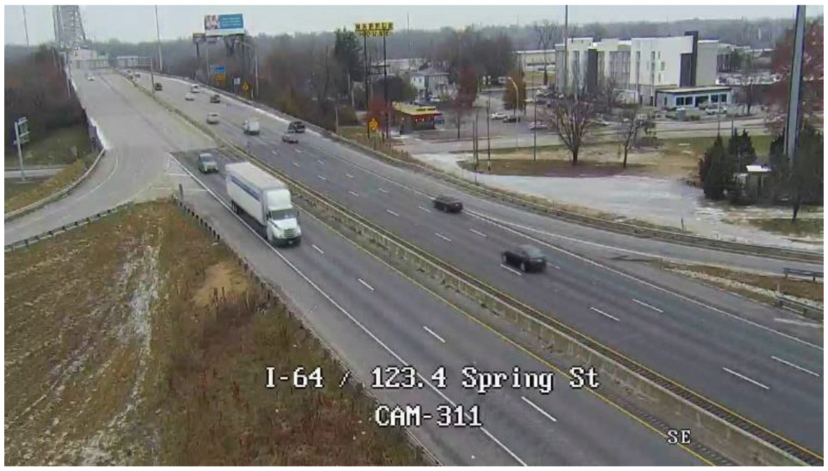


Figure 3.2 An example frame.

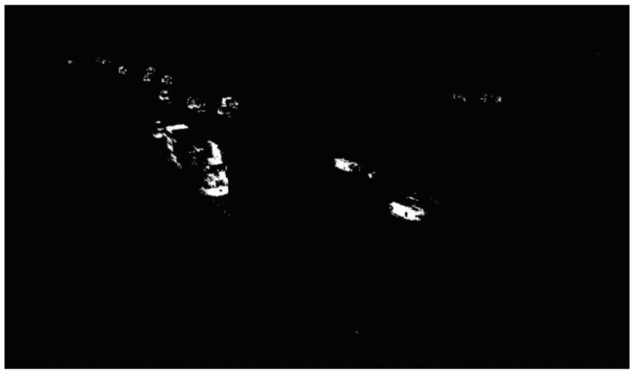


Figure 3.3 Current frame *i* operates on (*i*-1) frame, $i > 1$.

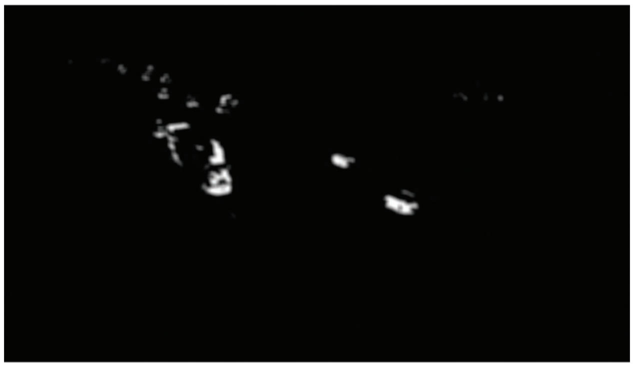


Figure 3.4 Blurred foreground.

the shapes formed by the contours obtained from Step iii with white using the OpenCV function, drawContours. The result will be like the image in Figure 3.7.

3.1.2 Vehicle Detection

Vehicle detection is the basis for lane detection. It is assumed that most of the cars are moving within a lane on the highway.

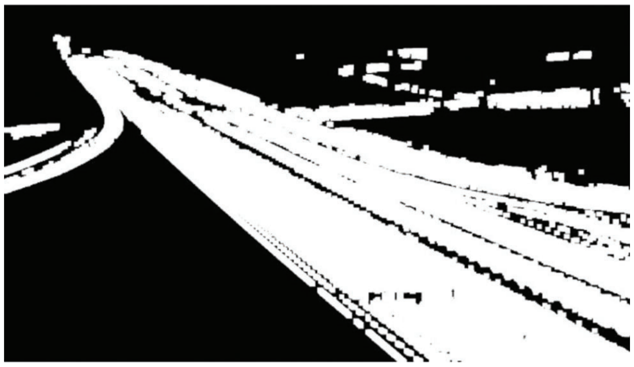


Figure 3.5 Generated road mask.

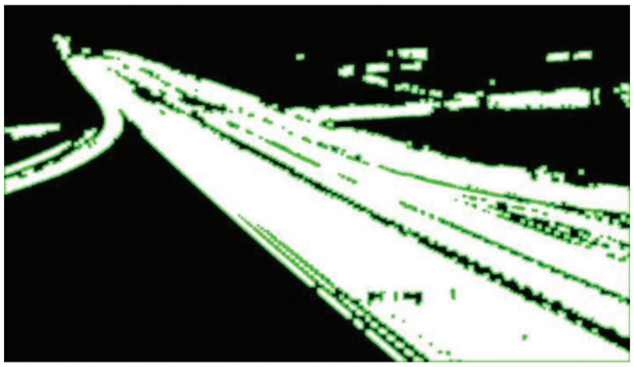


Figure 3.6 Image depicting initial contours of the mask image (illustration).

3.1.2.1 Introduction to YOLOv4. To detect and classify the vehicles in video frames, we explored various deep learning algorithms. One deep learning algorithm, You Only Look Once (YOLO, https://pjreddie.com/darknet/yolo/), matches what we need. YOLO can be trained to recognize hundreds of different objects (such as humans, cars, trucks, trains, motorcycles, etc.) in a video frame and directly generates bounding boxes around recognized objects in real-time. In this INDOT project, we implemented YOLO to achieve very high accuracy in various vehicle detection in real-time with the help of graphic processing units (GPUs).

All the object detection algorithms before YOLO use regions to localize the object within the image. The network does not look at the whole image each time. These algorithms take different regions of interest from the input image and use a CNN to classify the object's presence within that region. Using this approach, it may have to select many regions, which could highly increase the computation cost. YOLO uses a different approach. It applies a single neural network to the full image. This network divides the image into grids and predicts bounding boxes and probabilities for each grid. These bounding boxes are weighted by the predicted probabilities.

YOLO has several advantages over classifier-based systems. It looks at the whole image, so its predictions are informed by the global context in the image. It also makes predictions with a single network evaluation. Other systems (such as R-CNN) require thousands of evaluations for a single image. This makes YOLO more than 1,000 × faster than R-CNN and 100 × faster than Fast R-CNN. YOLO has been improved in several versions. The third version of the algorithm, YOLOv3, uses a few tricks to improve training and increase performance, including multi-scale predictions, a better backbone classifier, etc. Based on YOLOv3, YOLOv4 has improved the mean average precision (mAP) by as much as 10% and the number of frames per second by 12%. The architecture of YOLOv4 has three main parts, as shown in Figure 3.8, including the backbone, the neck, and the head. The backbone is the architecture to extract features. The neck block is used to aggregate features from different layers to enhance accuracy. The head block also localizes the boxes and does the classification. Moreover, the head block does both the dense prediction and the sparse prediction. By default, YOLOv4 only displays objects detected with a confidence of 0.5, which the user can change. Figure 3.9 shows the inference time and average performance on the coco dataset of the popular DNN based detectors. Compared with others, YOLOv4 achieved both higher accuracy and speed.

Figure 3.7 Clear road image.

YOLOv4 was built on Darknet, which is an open-source neural network framework written in C and CUDA. It is fast, easy to install, and supports CPU and GPU computation. YOLOv4 can also be implemented in the framework of TensorFlow and Pytorch. We built YOLOv4 using Darknet and implemented it on the Ubuntu system (Ubuntu 20.04.2 LTS).

3.1.2.2 Vehicle detection using YOLOv4. Figure 3.10 shows an example frame of vehicle detection using YOLOv4. YOLOv4 generated the box. The number associated with each box was the vehicle type and the confidence percentage of the vehicle recognition. Figure 3.11 shows another example of YOLOv4 based vehicle detection in lit night conditions. Even with low lighting in the evening, YOLOv4 still achieved high detection accuracy; however, the detection performance in the whole frame was not consistent. Some region with more lighting was much better than the darker part. So, in the next section, we will show why we define the ROI (region of interest) and perform the detection in the ROI rather than in the whole frame.

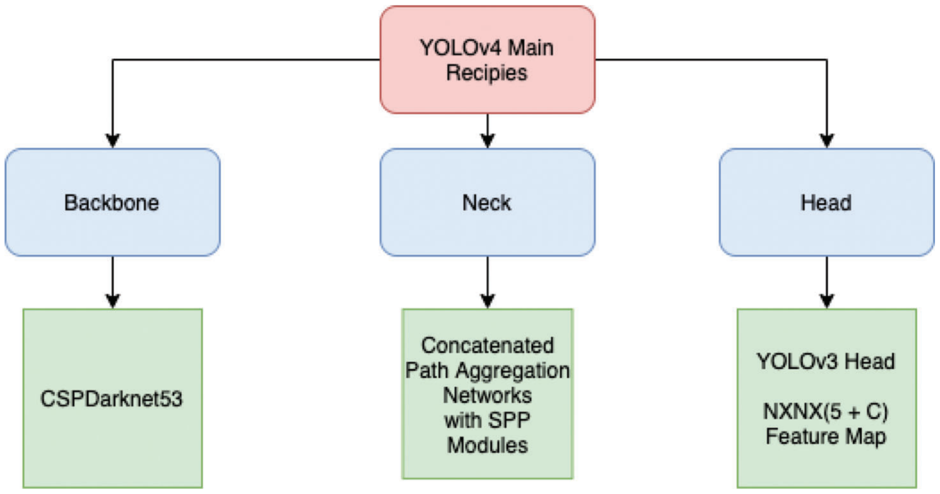


Figure 3.8 YOLOv4 network architecture.

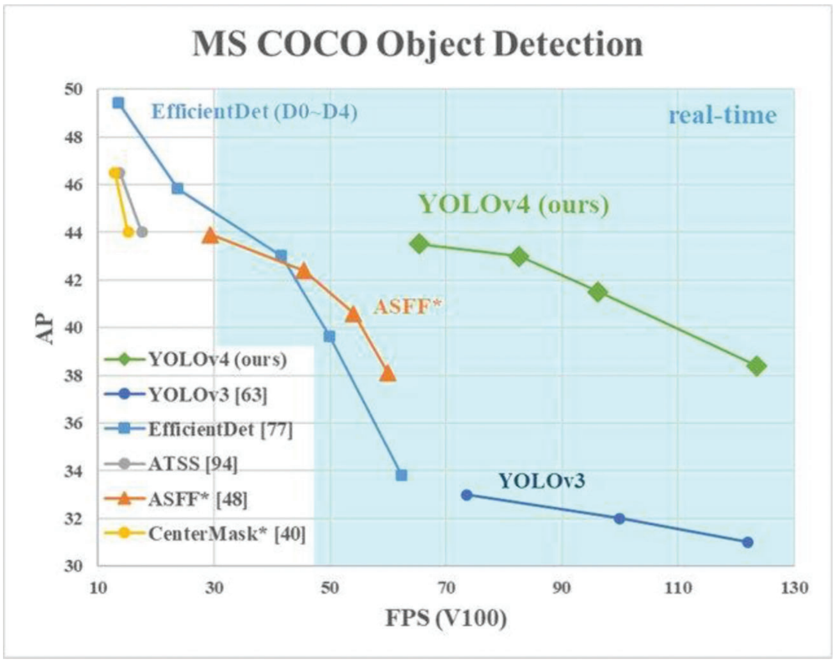


Figure 3.9 Inference time (milliseconds) and performance of YOLOv4 on COCO dataset.

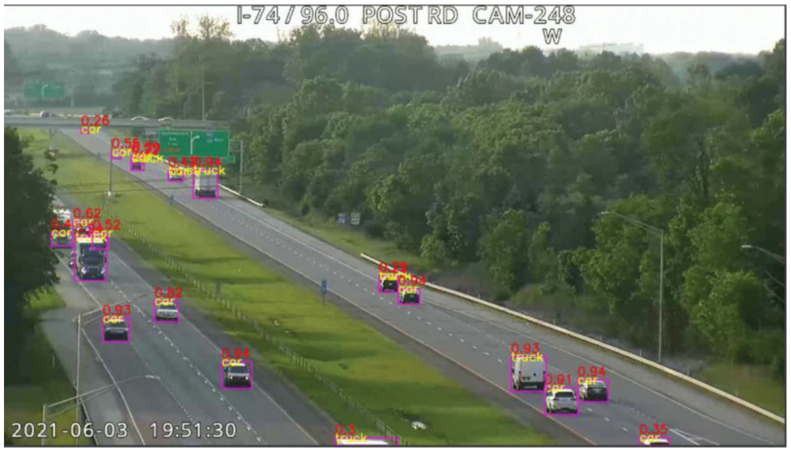


Figure 3.10 Results of YOLOv4 detection.

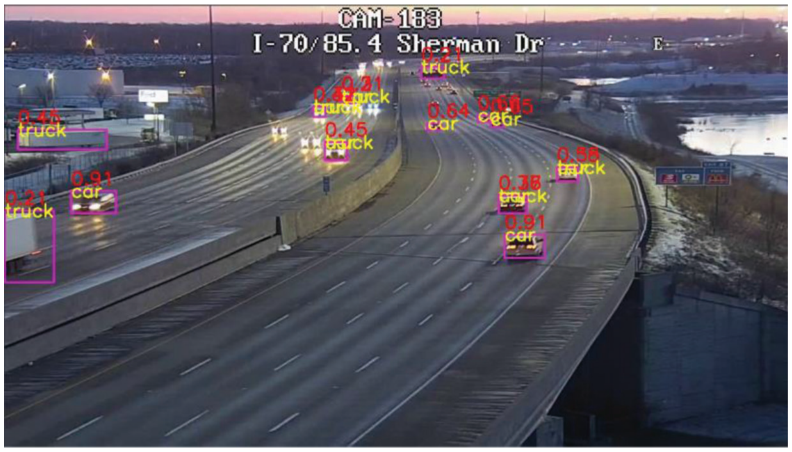


Figure 3.11 Detection results at nighttime (lit by light poles).

3.1.2.3 Vehicle detection accuracy evaluation. The vehicle detection performance was checked at an easy vehicle detection horizontal line by YOLOv4 with 21 videos under different lighting conditions. The result was the following.

- Day time detection rate is high (>95%).
- The detection rate at daytime decreased to 15% range when the images were not stable (vibrating).
- During the daytime, if the camera angle is oblique, the detection rate decreased.
- Evening time (before totally dark) detection rate was in 60%–98%.
- The detection rate was in 40%–70% range in dark-lit conditions. Thus, in the dark scene, YOLOv4 is not much better than YOLOv3.
- The detection rate was extremely low in the dark condition.

3.1.2.4 Computation speed of YOLOv4 per frame. Table 3.1 shows the execution time of YOLOv4 on the INDOT video frame in four computers. Based on Table 3.1, we can conclude the following.

- GPU is necessary.
- CPU's computation ability is more important for this INDOT project.
- For processing one video stream, a powerful GPU is not essential.

3.1.2.5 Remaining work for YOLOv4 based vehicle detection. Since the accuracy of YOLOv4 are essential to the applicability of this system to be developed, we still need to study several issues.

1. *Accuracy improvement.* There are still many situations in that YOLOv4 does not work well, such as the following.
   - Inconsistent vehicle detection in a sequence of video frames. Figure 3.10 shows a vehicle in a blue circle undetected in the current frame but detected in previous frames.
   - Poor vehicle detection in the unlit dark road. Since the vehicles cannot be seen on the video images and only the light halo can be seen, YOLOv4 cannot classify the light halo as vehicles. Therefore, other algorithms need to be developed.
   - Camera lens covered with raindrops.
   - False detection, as shown in a blue circle in Figure 3.12, the traffic sign on the road was detected as a truck.
2. *Computation cost improvement.*
   - Although we can achieve real-time operation, more efficient computation is desirable to reduce computing power requirements. There are two ways to reduce the cost, the first is to optimize the coding efficiency, and the second is to use parallel computation.

TABLE 3.1
The execution time of YOLOv4 based on Darknet for the INDOT video frame on four computers

YOLOv4 Implementation	GPU	CPU	YOLOv4 Time Per Frame
Darknet	GeForce RTX 2060 Memory: 15.6 G	Intel@Xeon(R)ES1650 v3@3.50 GHz × 12	≈0.039 s
Darknet	NVIDIA Corporation GP102 [GeForce GTX 1080 Ti] Memory: 15.6 G	Intel@Xeon(R)ES1650 v3@3.50 GHz × 12	≈0.031 s
Darknet	NVIDIA Corporation Quadro RTX 4000; Memory: 31 G	Intel@Xeon(R) W-2223 CPU@3.60 GHz × 8	≈0.038 s
Darknet	NVIDIA Corporation TU104GL [Quadro RTX 5000]; Memory: 31 G	Intel® Core^{TM} i7-9800X CPU@3.80 GHz × 16	≈0.035 s

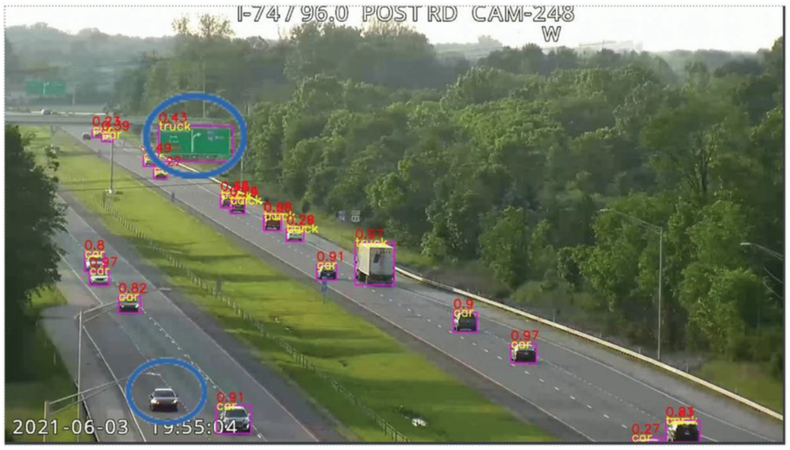


Figure 3.12 YOLOv4 missed vehicle detection and false classification.

3.1.3 Reference Line and Region of Interest (ROI) Determination

The goal of this project is to monitor the road conditions. When there is no entry and exit, all highway vehicles must pass one point on the highway. Therefore, road traffic conditions can be checked at the easiest checking point on the image. YOLOv4 detects the vehicles on the input images and associates a confidence index to each vehicle. The detected vehicles at a particular frame region have a higher confidence index than those in other regions. With the collection of confidence indices of thousands of vehicles detected in a sequence of image frames, the image's reference line and region of interest are derived based on a relatively constant and high vehicle detection confidence index. The ROI is mainly at the lower 1/4 to lower 1/2 of the image. Figure 3.13 shows an example of the reference line and region of interest.

3.1.4 Lane Center at Reference Line

Lane center detection is the detection of lanes on the reference line. It is based on vehicle detection and the assumption that most vehicles are driven within the lanes. The approach tracks the center location of the vehicles passing a given horizontal line in the image and counts the number of vehicles passing different pixels on the line. Over some time (e.g., 2 minutes), or over the number of vehicles passing the horizontal line, we can get a histogram where the horizontal axis is the pixel location on the horizontal line, and the vertical axis is the number of cars passed the horizontal line. Assuming that most vehicles are driven within a lane most time, the peaks on the plot indicate the location of each lane. Figure 3.14 shows a camera view of the road and generated histogram. Using mean filters to smooth out the close-by local peaks, a lane center is found based on the peaks of the filtered histogram. Figure 3.15 shows the examples of the lane center detected. The details of the lane center detection are published in *Intelligent Highway Lane Center Identification from Surveillance Camera Video* (Qiu et al., 2021). This approach does not rely on detecting lane marking that may not be detectable at night or in bad lighting conditions.

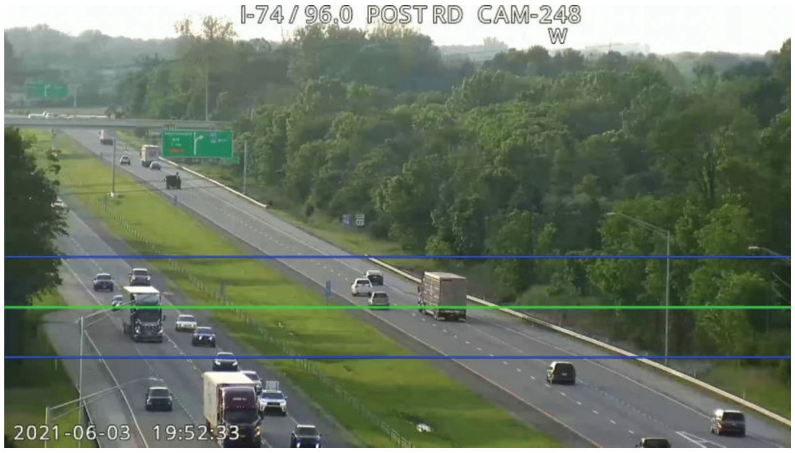
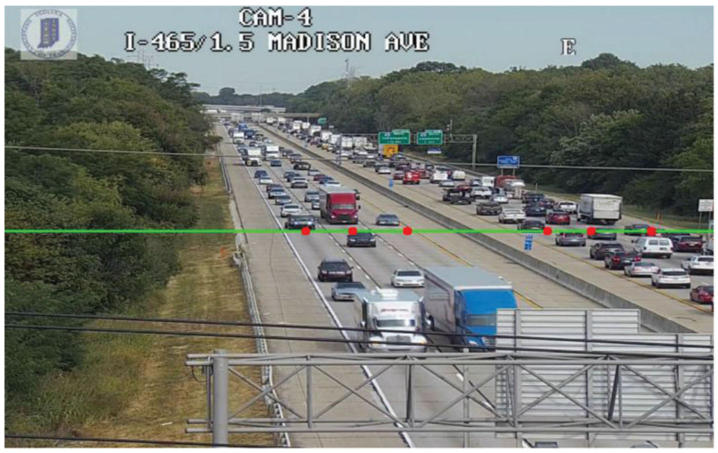


Figure 3.13 An example of the reference line and region of interest.



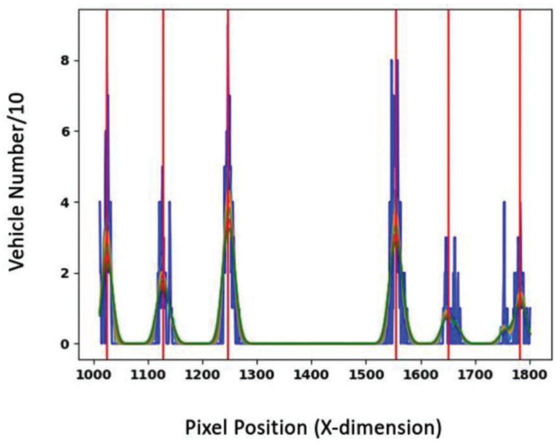


Figure 3.14 Vehicles pass the reference line and corresponding histogram.

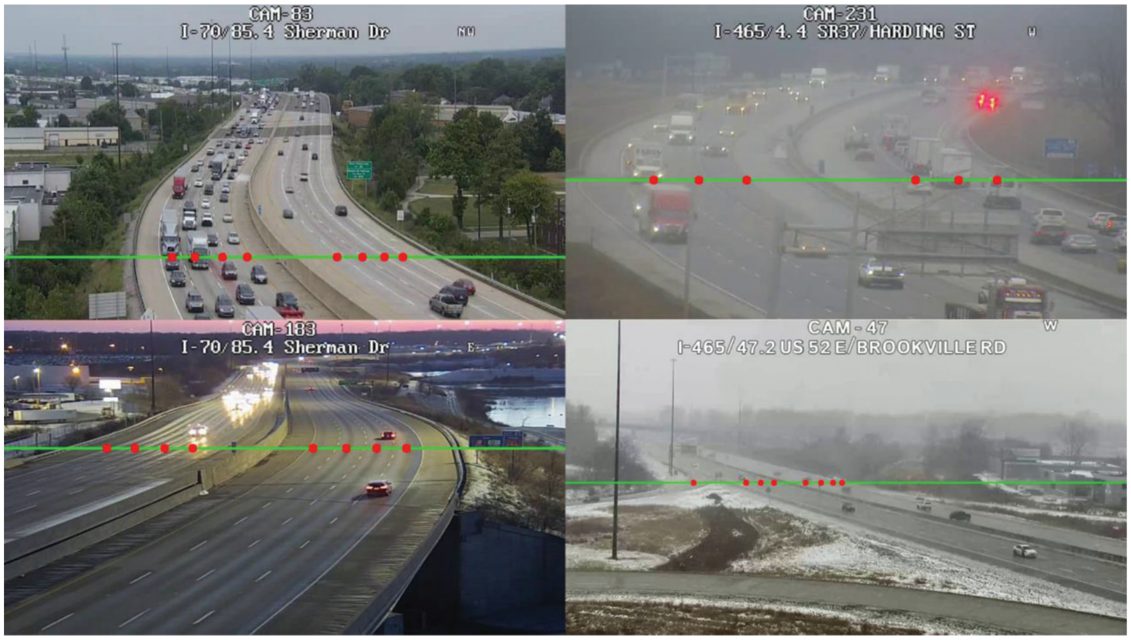


Figure 3.15 Detection of lane centers.

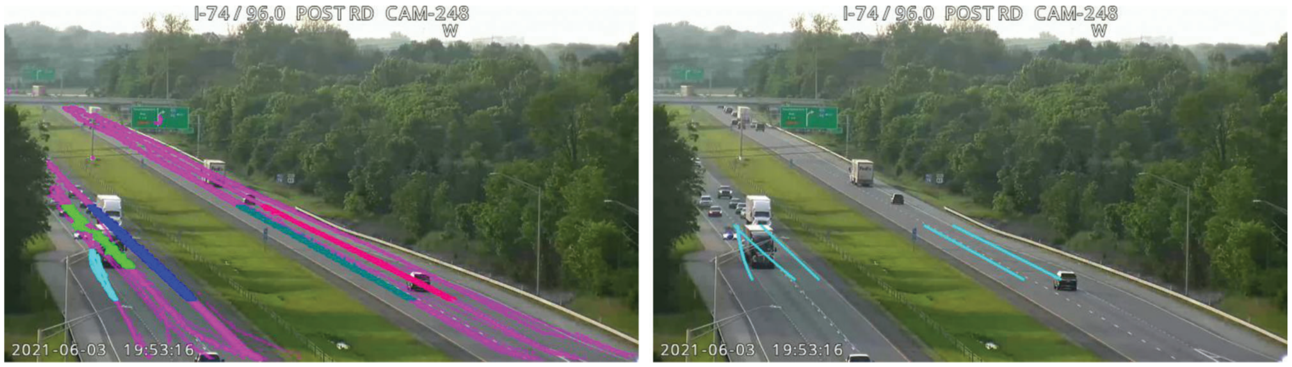


Figure 3.16 Vehicle clusters (left) and the centerlines (right) obtained from second-order curve fitting.

3.1.5 Lane Detection Based on Vehicle Clustering

Finding the lane center on a reference line is insufficient to find which vehicle is on which lane and lane direction. Since the lane on the video frame may have an angle to the vertical line and may not even be straight, the lane location in the ROI is needed to determine which vehicle is on which lane. We assumed that most vehicles are driven within a lane most of the time. So, starting at the lane centers detected in 3.1.4, we cluster all vehicle centers adjacent to several other vehicle centers in the same cluster. The result is that each cluster contains the vehicle in one lane. Then we use a second-order curve fitting function to find a curve function representing the lane's centerline. Since the vehicle detection in the ROI is most accurate, the segment of the derived centerline is more accurate within the ROI. Figure 3.16 shows an example of the vehicle clusters and the centerlines obtained from second-order curve fitting.

3.1.6 Lane Direction Detection in ROI

The lane direction detection is based on the vehicle motion direction. If the vehicle is not moving super-fast, the same vehicle can be seen at the close by location of the reference line in several consecutive video images. By tracking the vehicle positions on the consecutive images, we can determine the vehicle's moving direction. For example, if most vehicles in the same lane at the reference line move in the same direction, that is the lane direction. Figure 3.17 shows the lane directions with blue triangles.

3.1.7 Relationship of the Pixel Size to the Average Vehicle Size

Since the camera views the road at an angle, the actual physical area corresponding to a frame pixel varies from small to large as the pixels from the top to the bottom of the image frame. As the camera

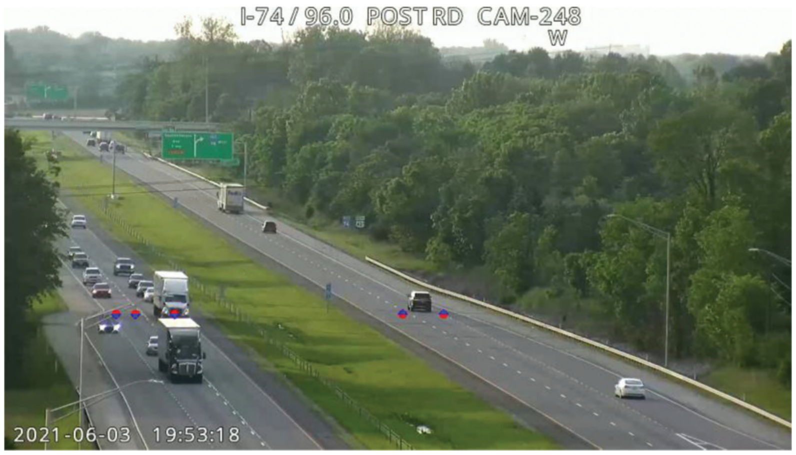


Figure 3.17 Automatically detected lanes and lane direction.

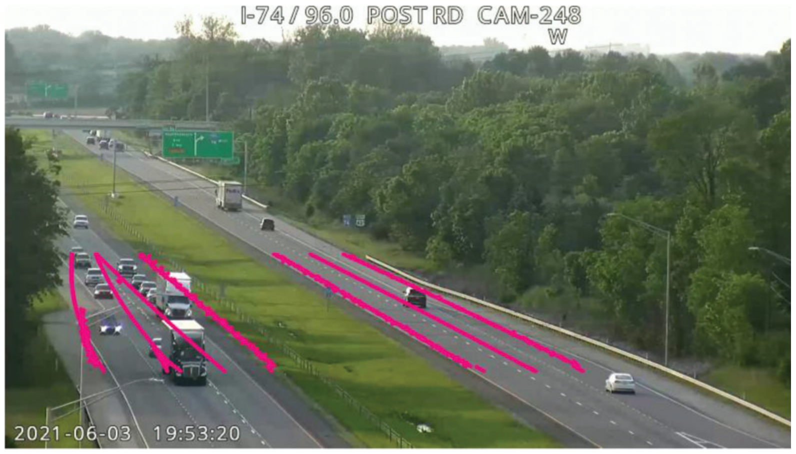


Figure 3.18 The example of lane boundaries generated.

viewing angle and zoom level can change arbitrarily, there is no accurate reference to calibrate the relationship between the pixel and the physical area. We use the average small car size as a reference for estimating the pixel to size relationship. Therefore, we calculate all detected small vehicles' average width and length at their detected vehicle location in each lane. The lane width can be estimated in the unit of average car width.

3.1.8 Boundaries of Each Lane

To count the vehicles for each lane, each YOLOv4 detected vehicle needs to be determined which lane it is on. Therefore, the boundary of each lane needs to be determined. We use the previously determined lane center curve as references. For the adjacent lanes of the same direction, the boundary between the adjacent lanes is the midpoint between the lane centers of the adjacent lane of the same direction. If a lane is an edge lane, the lane boundary to the roadside is defined as the average car width from the lane center. Figure 3.18 shows the lane boundaries of all detected lanes.

3.1.9 Camera Viewing Angle Change Detection

Since the traffic controllers operate the cameras, their viewing angles and zoom levels may change unpredictably. As the camera viewing angle changed, the learned lane information obsoletes, and the environment needs to be relearned. Therefore, this module is to detect the camera angle change based on the image background changes. The idea is to check the image difference for the areas that are not the road. There are some solutions available for detecting a camera angle change. However, the problem with most of them is that they require heavy computation. Therefore, the fundamental constraint of this project was to create a computationally inexpensive module. We developed a hash function module to overcome these problems and form a reliable method that can work very effectively in this project.

There are multiple benefits of using hashing, but the most important ones include the following.

1. The image hash will not change if the aspect ratio of our input image changes (as we disregard the aspect ratio).
2. Adjusting the brightness or contrast (1) will not change our hash value, or (2) will vary it slightly, ensuring that the hashes remain intact.

The description of the ImageHash algorithm is shown in Figure 3.19. The algorithm works straightforwardly while taking the input frame from the camera. In order to detect camera angle change, we need a background that does not change much, which means the part of the image not containing the road. After cropping the required area, we convert the RGB channel image into the Grayscale Image. Discarding color enables us to do the following.

- Hash the image faster as we only need to examine only one channel.
- Match the same images even with slightly modified color spaces (because color details have been removed).

The ImageHash algorithm converts the whole input image pixel data into 16-bit hexadecimal string data. This conversion is done internally by the ImageHash library, which is available as an open-source library. The 16-bit hash number output is of fixed length for every frame of any dimension. The Hash value is used to determine if the camera angle changed or not.

An important factor in using this method is how to select the two images to hash. If two frames are too far apart in time, like one early morning and one at noon, the background changes significantly, creating a large hash value. On the other hand, if two frames are too close in time, the slow background changes will not generate a large hash value, so the angle change is not detected. Through many test cases, we found that frames of 2 seconds apart with a hash value threshold of 20 are sufficient to detect camera angle changes but do not miss the camera angle changes. Hashing is *extremely fast;* it takes less than one millisecond to run on a PC.

3.1.10 Camera Direction Detection

The environment learning can only determine the road and lanes on the frames fed by the camera. We still need to know where the camera is facing to tell which incidents occur on which side of the road. This problem is partially resolved by automatically recognize the camera direction information on the top right of the frame. Tesseract OCR (Optical Character Recognition) is used to recognize and read the texts from an image. Following if the algorithm created.

For every video, the direction text has a fixed range of locations. It appears at the top right area right after the intersection name ends. For better recognition of text, it is recommended to focus more on the desired texts. Since other texts are not much of a concern for the application, the image is cropped and focused on direction text. Figure 3.20 shows the cropped region.

It is observed that text detection works more accurately for darker text fonts and lighter backgrounds. Moreover, it is known that the texts appearing on the frame are always white or closer to white (RGB value ranges from 220 to 255). The background is black, and then both colors (text and background) are inverted using PIL. The image is then enhanced and smoothened using the same library. The result is displayed in Figure 3.21.

It is then fed to the tesseract OCR method that reads the string from the image. The OCR Engine Mode is set to 3, and Page Segmentation Mode is set to 7 for cropped and processed images. Hence, the configuration used is such '—oem 3 –psm 7'. This results in output, as shown in Figure 3.22.

Text recognition is not 100% accurate since several factors affect the text in the image. The aim here is to read the direction, and it is known that there are at most eight different directions and their abbreviations are E, W, N, S, NW, NE, SE, and SW. Any other characters outside these are considered invalid texts.

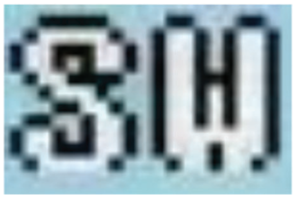


Figure 3.20 Cropped image.

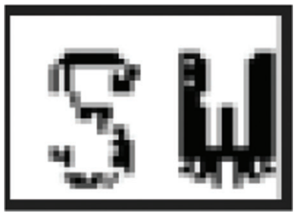
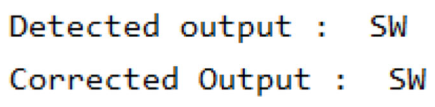


Figure 3.21 Cropped image in black and white.



```
Detected output :   SW
Corrected Output :   SW
```

Figure 3.22 Output of OCR.

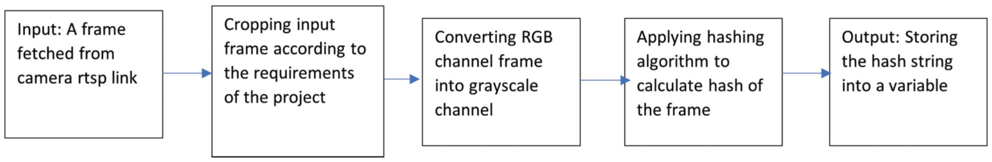

Input: A frame fetched from camera rtsp link
Cropping input frame according to the requirements of the project
Converting RGB channel frame into grayscale channel
Applying hashing algorithm to calculate hash of the frame
Output: Storing the hash string into a variable

Figure 3.19 Block diagram of the ImageHash method.

Hence, the output generated from the OCR is filtered through a method where similar texts are corrected. For example, if the OCR reads the text as "5", some inaccuracy turns in "S" into "5."

The accuracy of the text detection with the known text location is about 98% in over three hundred test cases.

3.1.11 Database Interface

At the end of the environment learning, the following information is sent to the database.

- The camera ID
- The direction that the camera is facing
- Baseline found by the program
- The number of lanes detected
- Timestamp
- Information for each lane:
    - The lane ID
    - Lane annotation
    - Location of the lane center is also stored

3.2 Incident Detection

Incident detection is conducted in real-time in an infinite loop. The real-time incident detection part has the following modules.

1. Video rate-computer speed synchronization
2. Vehicle detection
3. Filter out unrelated vehicles
4. Vehicle tracking
5. ID mapping
6. Vehicle count
7. Calculate vehicle flow rate
8. Calculate vehicle speed
9. Incidents detection
10. Camera direction change detection
11. Equipment error detection and reporting
12. Database interface
13. System diagnostics

They are explained in the following subsections.

3.2.1 Video Rate-Computer Speed Synchronization

There are many different types of cameras on the INDOT network. The frames per second and frame resolution of these cameras vary significantly. These factors affect vehicle tracking, which is essential for incident detection. To support the real-time vehicle detection and tracking, it is desirable to process a frame and skip from 0 to N frames. This module observes the computer processing time for each frame and the incoming image frames per second and determines how many frames should be skipped for each frame processed, so there will be no backlogged frames. Since the processing time for each frame is affected by the vehicles captured the frames, the number of frames skipped fluctuates. The incident detection program generates good results if the computer can process about 10 frames or above per second.

3.2.2 Vehicle Detection

This vehicle detection is the same as the one described in Section 3.1.2.

3.2.3 Filter Out Unrelated Vehicles

YOLOv4 detect vehicles on the whole frame. It has higher detection errors not in the ROI. These errors cause vehicle counting and vehicle tracking, and lane mapping errors. Therefore, we filer out all detected vehicles not in ROI and outside the road boundary.

3.2.4 Vehicle Tracking

This step matches the vehicles detected in the current frame to the vehicles detected in previous frames. When a vehicle is associated in the past three consecutive frames, and the average detection confidence score of the vehicle in the three frames is greater than 0.25, the vehicle is considered tracked and assigned a tracked ID in the frame. In unfavorable lighting conditions, a vehicle may be detected in one frame and not detected in the past and future frames, which causes the vehicle not to be tracked. Any tracked ID becomes invalid when it is not tracked again in more than 30 consecutive frames. Karmen filter is used for predicting the potential position of all the previous tracked vehicles in the current frame. The Cascade matching method is used to match the new detection and the predictions from Kalman Filter. IOU distance and Mahalanobis distance matrixes are used to set the threshold in the matching process. Hungarian assignment method uses the shortest distance for the data association/matching of the detected vehicle location and the predicted vehicle location.

3.2.5 Lane Vehicle ID Mapping

The ID mapping step determines which lane a new tracked vehicle belongs to. When a vehicle is first tracked in the region of interest, it is assigned a lane vehicle ID, depends on if its bounding box center is within which lanes boundaries. A lane ID indicates that the vehicle belongs to a specific lane. When a tracked vehicle already has a lane ID obtained in previous frames, it will not be assigned with a new lane ID. As the vehicle goes through the ROI, it may change lanes. To prevent a vehicle be counted in multiple lanes, the lane vehicle ID does not change even if the vehicle changes the lanes.

If there is no entry or exit, all vehicles on the road must pass any specific location. Therefore, we only count the vehicles at the reference line we learned in the environment learning part, where the vehicle detection and tracking are most accurate. Since the bounding box of a fast-moving vehicle may not intersect with the reference line, we use a small region above and below the reference line for vehicle mapping. The width of the region is about the length of the car bounding box of small vehicles. This narrow car mapping region ensures that a car is not missed from mapping and prevents the car from being multiple-mapped (multiple-mapped means a car is mapped multiple times along the road in the ROI region.

3.2.6 Vehicle Count

The lane vehicle ID of the first vehicle detected in that lane is assigned as a 1 at the beginning of the program execution. Each time a newly tracked car is mapped, the lane ID increment by 1 and being modular with the maximum integer. So, the lane vehicle ID is essentially the vehicle count over time.

3.2.7 Calculate Vehicle Flow Rate

Vehicle count is not meaningful for traffic management since it depends on when the count started. Vehicle count per time unit, flow rate, is a good measure of the traffic status. In this module, the flow rate uses the unit of passing vehicles per hour. The flow rate in each lane is estimated based on the lane vehicle ID increment over the last 60 seconds. The vehicle count per hour is updated every 1 minute. The time interval for the flow rate calculation cannot be too long since it will not show the short-term traffic status changes. On the other hand, the time interval for the flow rate calculation cannot be too short since it will fluctuate too much to understand it meaning.

3.2.8 Calculate Vehicle Speed

The vehicle speed (mph) is a desirable parameter to be detected. Unfortunately, due to the unpredictable camera viewing angle on the road, the camera video lacks a distance reference. Therefore, it is not possible to get accurate vehicle speed. The closest approximation is to use the length of the car bounding box at the reference line as the distance reference, such as 1 pixel equals x meters. However, the lengths of the vehicle bounding boxes vary significantly as the vehicle lengths vary significantly. Therefore, we use the lower 20 percentile of all cars (eliminating the trucks) bounding box length, L, as the car bounding box length in pixels. We assign aL as the average actual small car length in meters, where a is a multiplier to associate pixels to meters. Thus, we can find how many pixels the vehicle moves between adjacent frames when it passes the reference line. Since we do know the time between the adjacent frames, we can derive the vehicle mph using the average pixels travelled by multiple vehicles from the adjacent frames.

3.2.9 Traffic Status and Incidents Detection

At present, the traffic status and incident to be detected are traffic jams. Since the vehicle speed detected is not accurate and the detected flow rate can be accurate, we use flow rate to determine the traffic conditions. The traffic flow rate status is in four categories, high, normal, low, and jam based on the flow rate values. Since a low flow rate can be due to no vehicle or too many vehicles on the road, the estimated vehicle speed, or the recent average number of vehicles in the ROI can be used to separate these two situations. The traffic flow rate status and incidents are updated periodically every time the vehicle flow rate is updated and submitted to the database. This traffic flow rate status is then used to alert the traffic operators on the webpage. The thresholds to decide high, normal, and low and jam status will be decided by INDOT.

3.2.10 Camera Direction Change Detection

Since the traffic incident detection program runs 24 hours a day, the traffic operator may change the camera viewing angle. An algorithm was developed to detect significant background changes on the image. The algorithm is described in Section 3.1.9.

3.2.11 Equipment Error Detection and Reporting

As the program will be used for any of the 500+ cameras in Indiana, it is not convenient to detect equipment errors, such as camera malfunction, network disruption, camera not facing any road, etc. Therefore, this module constantly monitors the availability of the video input and image change pattern to decide equipment errors. Then, the error is reported to the database that notifies the traffic operators.

3.2.12 Database Interface

This module reports all lane-based information such as flow rates, speeds, and incidents with timestamps to the database every 30 seconds. The data are formatted as the SQL query based on the known database table structure. Two types of data are sent to the database: traffic data, and incidence data. For each data, we have two tables to store the information, one for the current traffic status, the other stores a history of the traffic status.

- *Traffic Data*
  - This data will include the traffic flow rate (for each lane), speed, density, sum traffic flow rate of each side of the road, the average speed of each side of the time, and the time that the data was generated and sent to the database.
- *Incident Data*
  - The incident detection module calculates incident data. The result is sent to the database as a text that is either "Fast," "Normal," "Slow," or "Jam." In addition, the creation time of the data will also be sent to the database.

3.2.13 System Diagnostics

Since the environmental conditions for the input videos from hundreds of cameras change significantly, many situations can cause the system not to work as expected. Therefore, we have implemented a system diagnostic module that can selectively output the intermediate results of any implemented module. In addition, this diagnostic module helps to pinpoint the reason for the unsatisfactory output. For example, when the vehicles are standstill jammed, this system diagnostic module indicates that the lane direction learning module does not work due to the lack of motion information.

4. DATABASE DEVELOPMENT

The automatic traffic incidents detection needs the information of the locations of the cameras, the adjacency graph of all cameras in the road network, number of lanes of each road, traffic directions of each lane, traffic flow rates for each road/lane, past incidents, and so on. This project uses a database as a central place to gather the information generated from camera images, the incident detection results derived from sensory information in the database and provide user interface information. Figure 4.1 shows the database tables and the relationship of the system being developed. Some tables describe the property of the cameras and their installation information. Some tables store the real-time traffic flow information generated by the image processing computers on the field. Some tables store the information of the incident generated from the traffic information. The database has been implemented in MySQL https://www.mysql.com/. It can convert it to PostgreSQL https://www.postgresql.org/ if needed. This database has

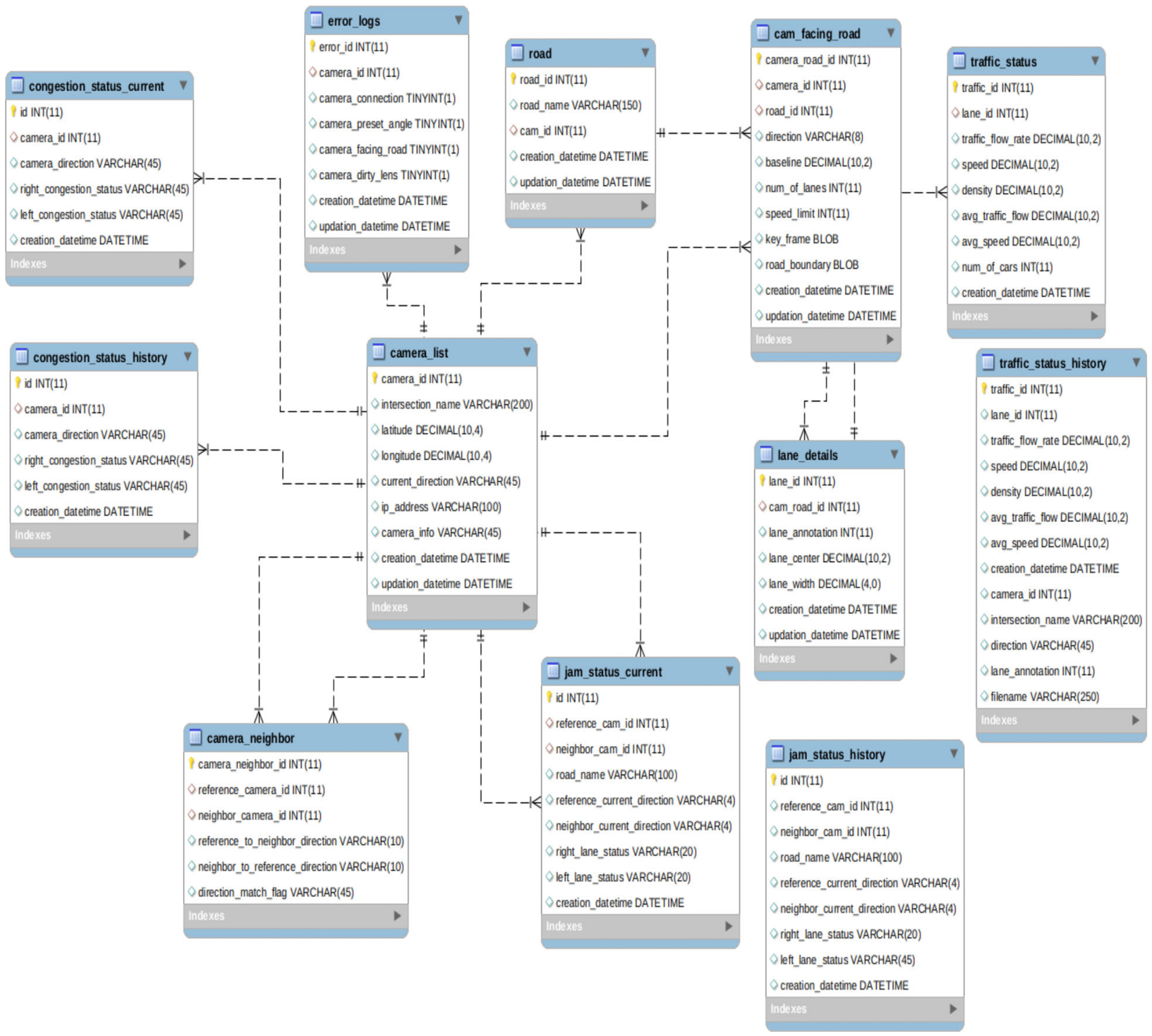

error_logs
error_id INT(11)
camera_id INT(11)
camera_connection TINYINT(1)
camera_preset_angle TINYINT(1)
camera_facing_road TINYINT(1)
camera_dirty_lens TINYINT(1)
creation_datetime DATETIME
updation_datetime DATETIME
Indexes
road
road_id INT(11)
road_name VARCHAR(150)
cam_id INT(11)
creation_datetime DATETIME
updation_datetime DATETIME
Indexes
cam_facing_road
camera_road_id INT(11)
camera_id INT(11)
road_id INT(11)
direction VARCHAR(8)
baseline DECIMAL(10,2)
num_of_lanes INT(11)
speed_limit INT(11)
key_frame BLOB
road_boundary BLOB
creation_datetime DATETIME
updation_datetime DATETIME
Indexes
traffic_status
traffic_id INT(11)
lane_id INT(11)
traffic_flow_rate DECIMAL(10,2)
speed DECIMAL(10,2)
density DECIMAL(10,2)
avg_traffic_flow DECIMAL(10,2)
avg_speed DECIMAL(10,2)
num_of_cars INT(11)
creation_datetime DATETIME
Indexes
congestion_status_current
id INT(11)
camera_id INT(11)
camera_direction VARCHAR(45)
right_congestion_status VARCHAR(45)
left_congestion_status VARCHAR(45)
creation_datetime DATETIME
Indexes
congestion_status_history
id INT(11)
camera_id INT(11)
camera_direction VARCHAR(45)
right_congestion_status VARCHAR(45)
left_congestion_status VARCHAR(45)
creation_datetime DATETIME
Indexes
camera_list
camera_id INT(11)
intersection_name VARCHAR(200)
latitude DECIMAL(10,4)
longitude DECIMAL(10,4)
current_direction VARCHAR(45)
ip_address VARCHAR(100)
camera_info VARCHAR(45)
creation_datetime DATETIME
updation_datetime DATETIME
Indexes
lane_details
lane_id INT(11)
cam_road_id INT(11)
lane_annotation INT(11)
lane_center DECIMAL(10,2)
lane_width DECIMAL(4,0)
creation_datetime DATETIME
updation_datetime DATETIME
Indexes
traffic_status_history
traffic_id INT(11)
lane_id INT(11)
traffic_flow_rate DECIMAL(10,2)
speed DECIMAL(10,2)
density DECIMAL(10,2)
avg_traffic_flow DECIMAL(10,2)
avg_speed DECIMAL(10,2)
creation_datetime DATETIME
camera_id INT(11)
intersection_name VARCHAR(200)
direction VARCHAR(45)
lane_annotation INT(11)
filename VARCHAR(250)
Indexes
jam_status_current
id INT(11)
reference_cam_id INT(11)
neighbor_cam_id INT(11)
road_name VARCHAR(100)
reference_current_direction VARCHAR(4)
neighbor_current_direction VARCHAR(4)
right_lane_status VARCHAR(20)
left_lane_status VARCHAR(20)
creation_datetime DATETIME
Indexes
camera_neighbor
camera_neighbor_id INT(11)
reference_camera_id INT(11)
neighbor_camera_id INT(11)
reference_to_neighbor_direction VARCHAR(10)
neighbor_to_reference_direction VARCHAR(10)
direction_match_flag VARCHAR(45)
Indexes
jam_status_history
id INT(11)
reference_cam_id INT(11)
neighbor_cam_id INT(11)
road_name VARCHAR(100)
reference_current_direction VARCHAR(4)
neighbor_current_direction VARCHAR(4)
right_lane_status VARCHAR(20)
left_lane_status VARCHAR(45)
creation_datetime DATETIME
Indexes

Figure 4.1 ER-diagram of the designed database.

been hosted on a TSI computer on the IU network and kept running for months.

5. WEB-BASED GRAPHICAL USER INTERFACE

A web-based Graphical User Interface (GUI) was developed with the requirement input from INDOT. The web-based GUI reads data from the database as described in Section 4 and generates the user-interested information in an easy-to-read output display. Figure 5.1 shows a screenshot of the road traffic monitored by camera ID 175 by the field computer. The environment was learned by the traffic status monitoring and incident detection program on one computer, and the result is also displayed in Figure 5.1. The result data was sent to the database and reflected on the webpage (Figure 5.2). Figure 5.2 has four major components. The upper left shows a Google Map with camera locations. The right side shows a table of all cameras. In this table, the jammed roads are shown in red and at the top of the list (Figure 5.3). The lower left graphs show the flow rates on the road's left and right sides in the past one hour. Finally, the middle bottom shows the road status information observed by the currently focused camera. This window supports the following operations.

- When the user selects a specific camera on the table on the right of the screen, the webpage displays the traffic

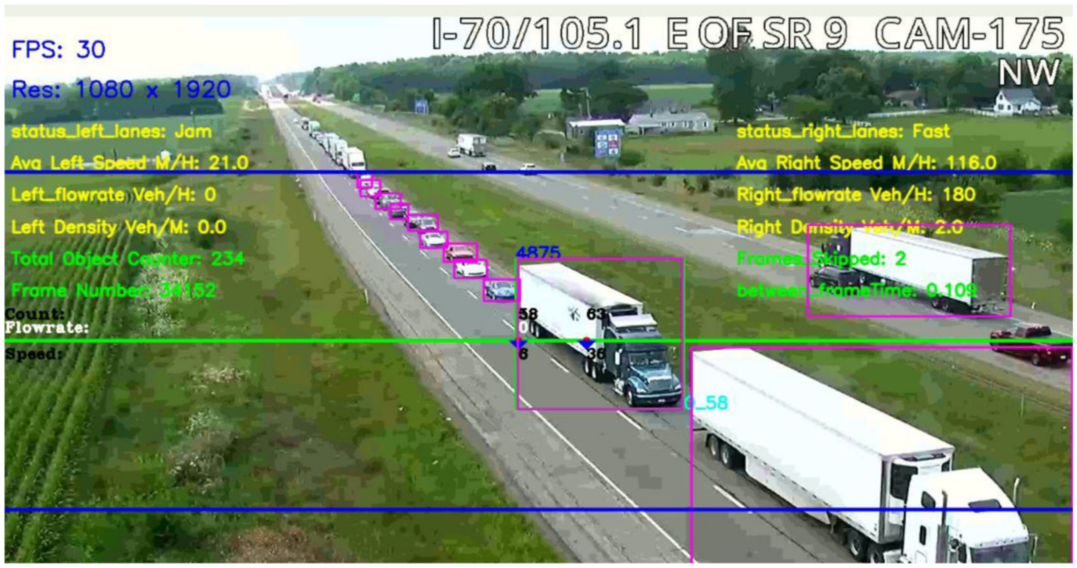
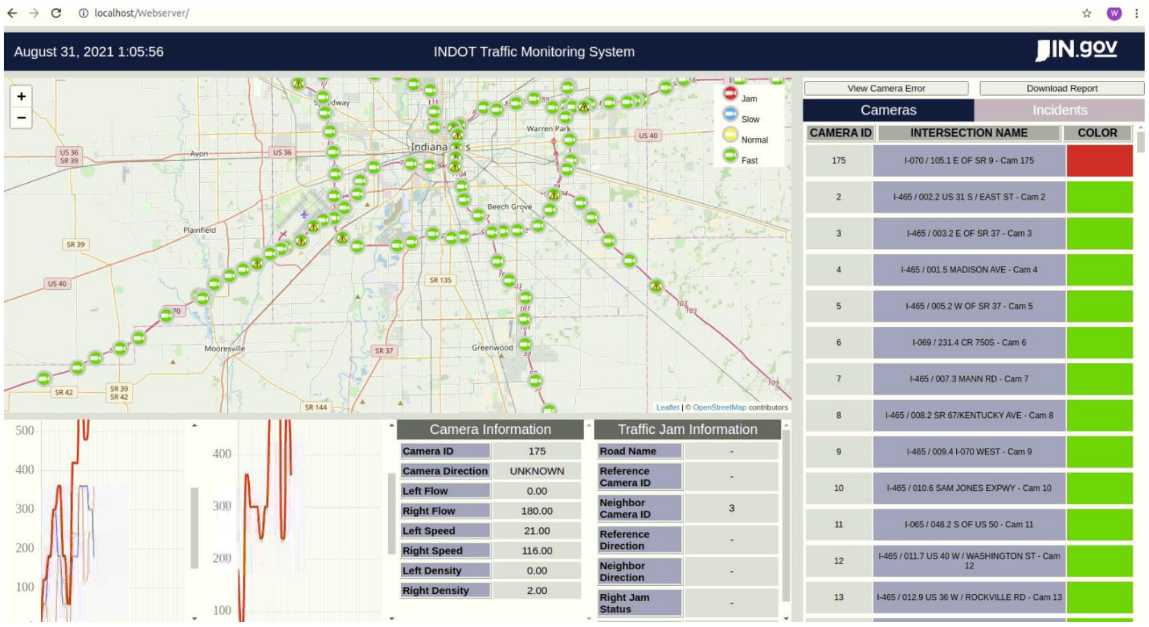


Figure 5.1 The environment was learned by the traffic status monitoring and incident detection program on the field computer.



August 31, 2021 1:05:56

INDOT Traffic Monitoring System

View Camera Error | Download Report

Cameras | Incidents

CAMERA ID	INTERSECTION NAME	COLOR
175	I-070 / 105.1 E OF SR 9 - Cam 175	🟥
2	I-465 / 002.2 US 31 S / EAST ST - Cam 2	🟩
3	I-465 / 003.2 E OF SR 37 - Cam 3	🟩
4	I-465 / 001.5 MADISON AVE - Cam 4	🟩
5	I-465 / 005.2 W OF SR 37 - Cam 5	🟩
6	I-069 / 231.4 CR 750S - Cam 6	🟩
7	I-465 / 007.3 MANN RD - Cam 7	🟩
8	I-465 / 008.2 SR 67/KENTUCKY AVE - Cam 8	🟩
9	I-465 / 009.4 I-070 WEST - Cam 9	🟩
10	I-465 / 010.6 SAM JONES EXPWY - Cam 10	🟩
11	I-065 / 048.2 S OF US 50 - Cam 11	🟩
12	I-465 / 011.7 US 40 W / WASHINGTON ST - Cam 12	🟩
13	I-465 / 012.9 US 36 W / ROCKVILLE RD - Cam 13	🟩

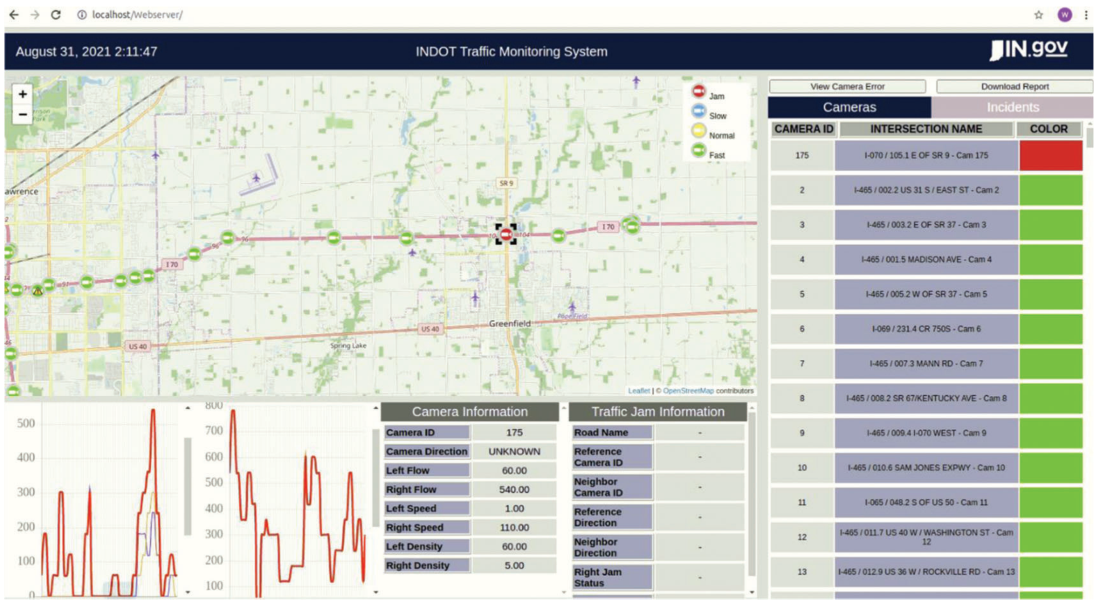
Camera Information

Camera ID	175
Camera Direction	UNKNOWN
Left Flow	0.00
Right Flow	180.00
Left Speed	21.00
Right Speed	116.00
Left Density	0.00
Right Density	2.00

Traffic Jam Information

Road Name	-
Reference Camera ID	-
Neighbor Camera ID	3
Reference Direction	-
Neighbor Direction	-
Right Jam Status	-

Figure 5.2 The webpage displays the traffic status submitted by the field computer.



August 31, 2021 2:11:47

INDOT Traffic Monitoring System

IN.gov

View Camera Error | Download Report

Cameras | Incidents

CAMERA ID	INTERSECTION NAME	COLOR
175	I-070 / 105.1 E OF SR 9 - Cam 175	🟥
2	I-465 / 002.2 US 31 S / EAST ST - Cam 2	🟩
3	I-465 / 003.2 E OF SR 37 - Cam 3	🟩
4	I-465 / 001.5 MADISON AVE - Cam 4	🟩
5	I-465 / 005.2 W OF SR 37 - Cam 5	🟩
6	I-069 / 231.4 CR 750S - Cam 6	🟩
7	I-465 / 007.3 MANN RD - Cam 7	🟩
8	I-465 / 008.2 SR 67/KENTUCKY AVE - Cam 8	🟩
9	I-465 / 009.4 I-070 WEST - Cam 9	🟩
10	I-465 / 010.6 SAM JONES EXPWY - Cam 10	🟩
11	I-065 / 048.2 S OF US 50 - Cam 11	🟩
12	I-465 / 011.7 US 40 W / WASHINGTON ST - Cam 12	🟩
13	I-465 / 012.9 US 36 W / ROCKVILLE RD - Cam 13	🟩

Camera Information

Camera ID	175
Camera Direction	UNKNOWN
Left Flow	60.00
Right Flow	540.00
Left Speed	1.00
Right Speed	110.00
Left Density	60.00
Right Density	5.00

Traffic Jam Information

Road Name	-
Reference Camera ID	-
Neighbor Camera ID	-
Reference Direction	-
Neighbor Direction	-
Right Jam Status	-

Figure 5.3 The webpage displayed the traffic status and zoomed-in map with a specific camera selected.

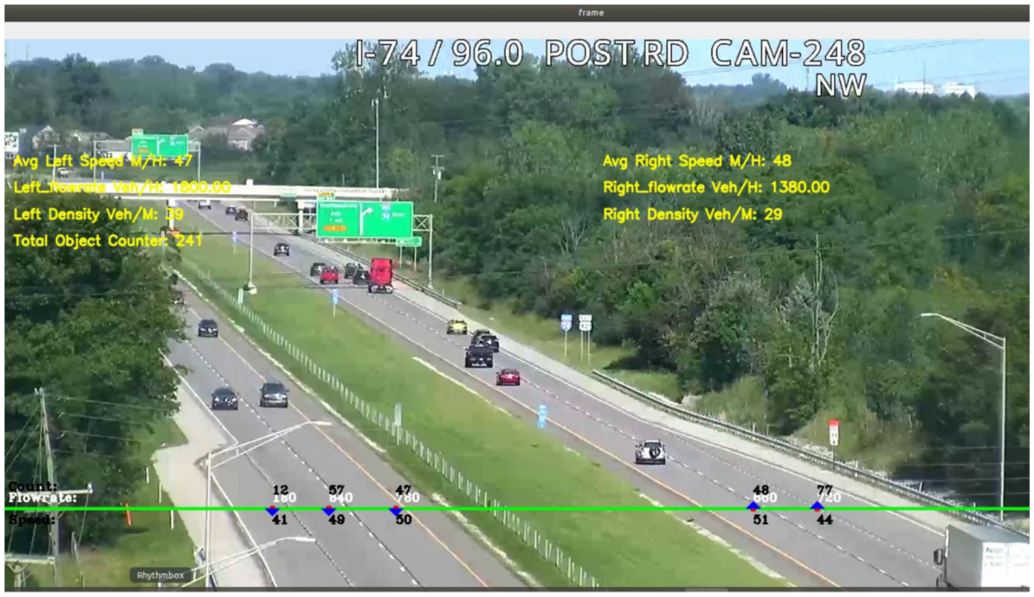


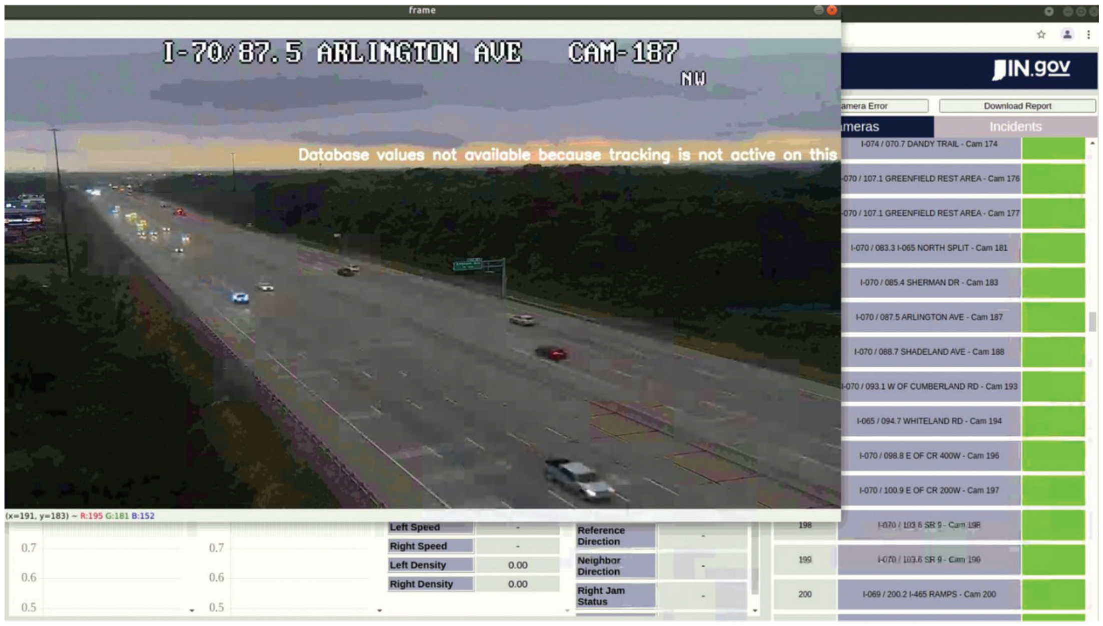
Figure 5.4 The current traffic status overlays on the live video from that camera.

status and zoomed-in map with a specific camera selected (see Figure 5.3).

- By clicking a camera on the camera symbol on the right, we can change the focus to a different camera location and show the current traffic status overlays on the live video from that camera (see Figure 5.4). For example, if there are no values in the database for this camera, it will show that "Database values are not available because tracking is not active on this camera" (Figure 5.5).
- When a user clicks the "Incidents" tab in the right-side component, the UGI switches to a screen, as shown in Figure 5.6. The dynamic incident information is pulled from the database.

The web page also has the following functions. By clicking on the "View Camera Error" tab located at upper right (above the Camera and Incidents tab). This table will pop up. An "x" means there is an error. (Figure 5.7). Clicking on the download report next to the View Camera Error allows download report with specified DateTime. Clicking on one of the cameras in the table will show the live camera feed. If there are values associated with that camera in the database, it will also display the values.

The webpage can provide a report for the traffic status history of any camera for the specified time period (see Figure 5.8) with an output Excel file in the format shown in Figure 5.9.

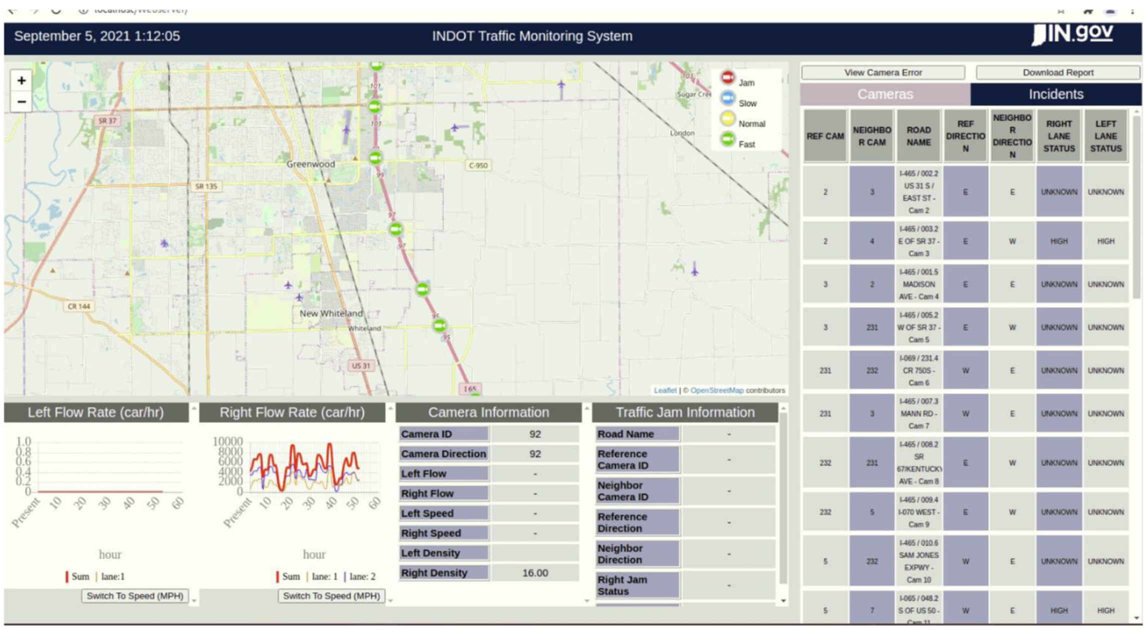

frame
I-70/87.5 ARLINGTON AVE CAM-187
NW
Database values not available because tracking is not active on this
(x=191, y=183) ~ R:195 G:181 B:152
IN.gov
Download Report
Incidents

Left Speed	-
Right Speed	-
Left Density	0.00
Right Density	0.00

Reference Direction	-
Neighbor Direction	-
Right Jam Status	-

	I-074 / 070.7 DANDY TRAIL - Cam 174	🟩
	070 / 107.1 GREENFIELD REST AREA - Cam 176	🟩
	070 / 107.1 GREENFIELD REST AREA - Cam 177	🟩
	I-070 / 083.3 I-065 NORTH SPLIT - Cam 181	🟩
	I-070 / 085.4 SHERMAN DR - Cam 183	🟩
	I-070 / 087.5 ARLINGTON AVE - Cam 187	🟩
	I-070 / 088.7 SHADELAND AVE - Cam 188	🟩
	I-070 / 093.1 W OF CUMBERLAND RD - Cam 193	🟩
	I-065 / 094.7 WHITELAND RD - Cam 194	🟩
	I-070 / 098.8 E OF CR 400W - Cam 196	🟩
	I-070 / 100.9 E OF CR 200W - Cam 197	🟩
198	I-070 / 103.6 SR 9 - Cam 198	🟩
199	I-070 / 103.6 SR 9 - Cam 199	🟩
200	I-069 / 200.2 I-465 RAMPS - Cam 200	🟩

Figure 5.5 Display of camera when there is no value in the database.



Camera Information

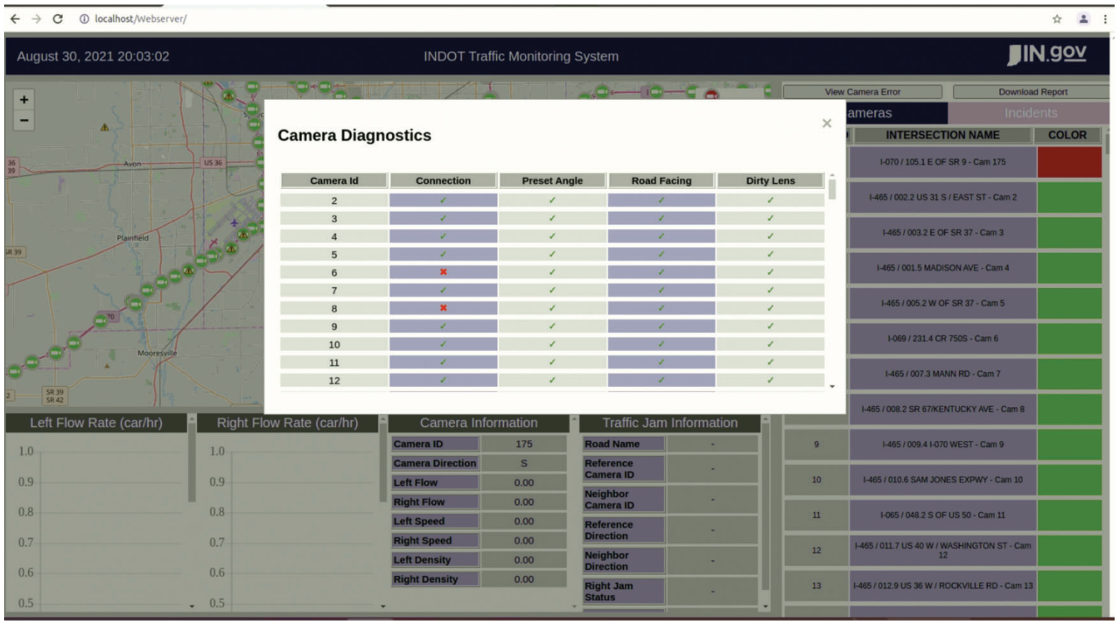
Camera ID	92
Camera Direction	92
Left Flow	-
Right Flow	-
Left Speed	-
Right Speed	-
Left Density	
Right Density	16.00

Traffic Jam Information

Road Name	-
Reference Camera ID	-
Neighbor Camera ID	-
Reference Direction	-
Neighbor Direction	-
Right Jam Status	-

REF CAM	NEIGHBOR CAM	ROAD NAME	REF DIRECTION	NEIGHBOR DIRECTION	RIGHT LANE STATUS	LEFT LANE STATUS
2	3	I-465 / 002.2 US 31 S / EAST ST - Cam 2	E	E	UNKNOWN	UNKNOWN
2	4	I-465 / 003.2 E OF SR 37 - Cam 3	E	W	HIGH	HIGH
3	2	I-465 / 001.5 MADISON AVE - Cam 4	E	E	UNKNOWN	UNKNOWN
3	231	I-465 / 005.2 W OF SR 37 - Cam 5	E	W	UNKNOWN	UNKNOWN
231	232	I-069 / 231.4 CR 750S - Cam 6	W	E	UNKNOWN	UNKNOWN
231	3	I-465 / 007.3 MANN RD - Cam 7	W	E	UNKNOWN	UNKNOWN
232	231	I-465 / 008.2 SR 67/KENTUCKY AVE - Cam 8	E	W	UNKNOWN	UNKNOWN
232	5	I-465 / 009.4 I-070 WEST - Cam 9	E	W	UNKNOWN	UNKNOWN
5	232	I-465 / 010.6 SAM JONES EXPWY - Cam 10	W	E	UNKNOWN	UNKNOWN
5	7	I-065 / 048.2 S OF US 50 - Cam 11	W	E	HIGH	HIGH

Figure 5.6 Clicking on the incidents tab, the webpage will grab information from the database to display the incident table.



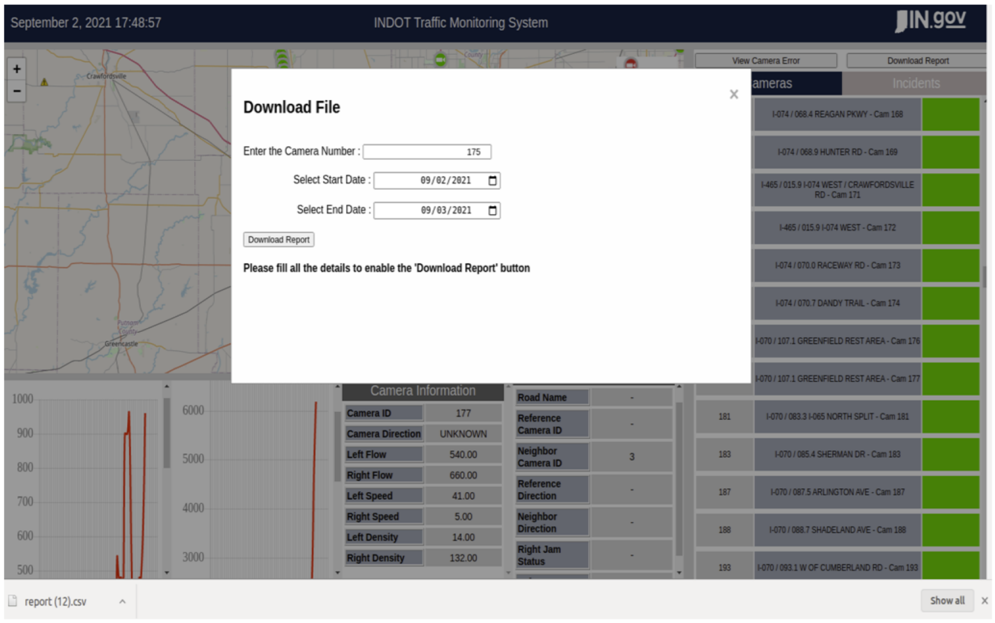
Camera Id	Connection	Preset Angle	Road Facing	Dirty Lens
2	✓	✓	✓	✓
3	✓	✓	✓	✓
4	✓	✓	✓	✓
5	✓	✓	✓	✓
6	✗	✓	✓	✓
7	✓	✓	✓	✓
8	✗	✓	✓	✓
9	✓	✓	✓	✓
10	✓	✓	✓	✓
11	✓	✓	✓	✓
12	✓	✓	✓	✓

	INTERSECTION NAME	COLOR
	I-070 / 105.1 E OF SR 9 - Cam 175	🟥
	I-465 / 002.2 US 31 S / EAST ST - Cam 2	🟩
	I-465 / 003.2 E OF SR 37 - Cam 3	🟩
	I-465 / 001.5 MADISON AVE - Cam 4	🟩
	I-465 / 005.2 W OF SR 37 - Cam 5	🟩
	I-069 / 231.4 CR 750S - Cam 6	🟩
	I-465 / 007.3 MANN RD - Cam 7	🟩
	I-465 / 008.2 SR 67/KENTUCKY AVE - Cam 8	🟩
9	I-465 / 009.4 I-070 WEST - Cam 9	🟩
10	I-465 / 010.6 SAM JONES EXPWY - Cam 10	🟩
11	I-065 / 048.2 S OF US 50 - Cam 11	🟩
12	I-465 / 011.7 US 40 W / WASHINGTON ST - Cam 12	🟩
13	I-465 / 012.9 US 36 W / ROCKVILLE RD - Cam 13	🟩

Camera ID	175
Camera Direction	S
Left Flow	0.00
Right Flow	0.00
Left Speed	0.00
Right Speed	0.00
Left Density	0.00
Right Density	0.00

Road Name	-
Reference Camera ID	-
Neighbor Camera ID	-
Reference Direction	-
Neighbor Direction	-
Right Jam Status	-

Figure 5.7 Webpage showing the status of all cameras.



Download File

Enter the Camera Number : 175

Select Start Date : 09/02/2021

Select End Date : 09/03/2021

Download Report

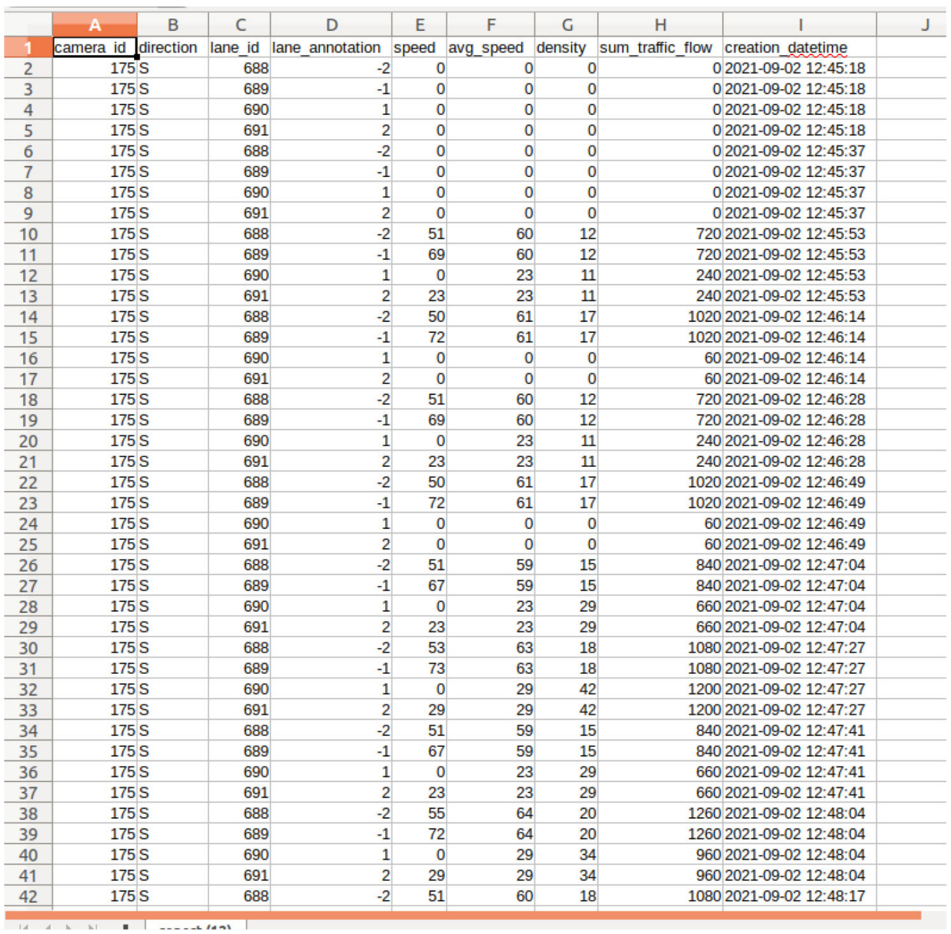
Please fill all the details to enable the 'Download Report' button

	I-074 / 068.4 REAGAN PKWY - Cam 168	🟩
	I-074 / 068.9 HUNTER RD - Cam 169	🟩
	I-465 / 015.9 I-074 WEST / CRAWFORDSVILLE RD - Cam 171	🟩
	I-465 / 015.9 I-074 WEST - Cam 172	🟩
	I-074 / 070.0 RACEWAY RD - Cam 173	🟩
	I-074 / 070.7 DANDY TRAIL - Cam 174	🟩
	I-070 / 107.1 GREENFIELD REST AREA - Cam 176	🟩
	I-070 / 107.1 GREENFIELD REST AREA - Cam 177	🟩
181	I-070 / 083.3 I-065 NORTH SPLIT - Cam 181	🟩
183	I-070 / 085.4 SHERMAN DR - Cam 183	🟩
187	I-070 / 087.5 ARLINGTON AVE - Cam 187	🟩
188	I-070 / 088.7 SHADELAND AVE - Cam 188	🟩
193	I-070 / 093.1 W OF CUMBERLAND RD - Cam 193	🟩

Camera ID	177
Camera Direction	UNKNOWN
Left Flow	540.00
Right Flow	660.00
Left Speed	41.00
Right Speed	5.00
Left Density	14.00
Right Density	132.00

Road Name	-
Reference Camera ID	-
Neighbor Camera ID	3
Reference Direction	-
Neighbor Direction	-
Right Jam Status	-

Figure 5.8 Generating a report file of a vehicle in a time period.



1	camera_id	direction	lane_id	lane_annotation	speed	avg_speed	density	sum_traffic_flow	creation_datetime
2	175	S	688	-2	0	0	0	0	2021-09-02 12:45:18
3	175	S	689	-1	0	0	0	0	2021-09-02 12:45:18
4	175	S	690	1	0	0	0	0	2021-09-02 12:45:18
5	175	S	691	2	0	0	0	0	2021-09-02 12:45:18
6	175	S	688	-2	0	0	0	0	2021-09-02 12:45:37
7	175	S	689	-1	0	0	0	0	2021-09-02 12:45:37
8	175	S	690	1	0	0	0	0	2021-09-02 12:45:37
9	175	S	691	2	0	0	0	0	2021-09-02 12:45:37
10	175	S	688	-2	51	60	12	720	2021-09-02 12:45:53
11	175	S	689	-1	69	60	12	720	2021-09-02 12:45:53
12	175	S	690	1	0	23	11	240	2021-09-02 12:45:53
13	175	S	691	2	23	23	11	240	2021-09-02 12:45:53
14	175	S	688	-2	50	61	17	1020	2021-09-02 12:46:14
15	175	S	689	-1	72	61	17	1020	2021-09-02 12:46:14
16	175	S	690	1	0	0	0	60	2021-09-02 12:46:14
17	175	S	691	2	0	0	0	60	2021-09-02 12:46:14
18	175	S	688	-2	51	60	12	720	2021-09-02 12:46:28
19	175	S	689	-1	69	60	12	720	2021-09-02 12:46:28
20	175	S	690	1	0	23	11	240	2021-09-02 12:46:28
21	175	S	691	2	23	23	11	240	2021-09-02 12:46:28
22	175	S	688	-2	50	61	17	1020	2021-09-02 12:46:49
23	175	S	689	-1	72	61	17	1020	2021-09-02 12:46:49
24	175	S	690	1	0	0	0	60	2021-09-02 12:46:49
25	175	S	691	2	0	0	0	60	2021-09-02 12:46:49
26	175	S	688	-2	51	59	15	840	2021-09-02 12:47:04
27	175	S	689	-1	67	59	15	840	2021-09-02 12:47:04
28	175	S	690	1	0	23	29	660	2021-09-02 12:47:04
29	175	S	691	2	23	23	29	660	2021-09-02 12:47:04
30	175	S	688	-2	53	63	18	1080	2021-09-02 12:47:27
31	175	S	689	-1	73	63	18	1080	2021-09-02 12:47:27
32	175	S	690	1	0	29	42	1200	2021-09-02 12:47:27
33	175	S	691	2	29	29	42	1200	2021-09-02 12:47:27
34	175	S	688	-2	51	59	15	840	2021-09-02 12:47:41
35	175	S	689	-1	67	59	15	840	2021-09-02 12:47:41
36	175	S	690	1	0	23	29	660	2021-09-02 12:47:41
37	175	S	691	2	23	23	29	660	2021-09-02 12:47:41
38	175	S	688	-2	55	64	20	1260	2021-09-02 12:48:04
39	175	S	689	-1	72	64	20	1260	2021-09-02 12:48:04
40	175	S	690	1	0	29	34	960	2021-09-02 12:48:04
41	175	S	691	2	29	29	34	960	2021-09-02 12:48:04
42	175	S	688	-2	51	60	18	1080	2021-09-02 12:48:17

Figure 5.9 Example report table of one camera during a time period.

6. TEST CASES

The developed traffic status monitoring system is under production tests. Two test cases have been conducted. These two cases are described in the following subsections.

6.1 Test Case 1

The first test case was at the ramp from South Shadeland Avenue to I-465 southbound. INDOT was interested in the vehicle count and passing patterns (platooning or not) during busy hours at 7 to 9 am and afternoon 2 to 6 pm daily. INDOT was also interested in the number of passenger cars and trucks. Since the existing highway cameras were far from this ramp, a camera on a temporary trailer was moved to the roadside of the ramp. The height of the camera was 40 feet. TASI was privileged to control the camera pan_tilt_zoom, so the camera angle was set to the best possible viewing angle for easier image processing. The test lasted over two weeks. Since the communication between this temporary camera and the communication tower uses Wi-Fi, the frame per second is only 6. This slow fps is different from all cameras installed on the highway. We need to adjust the vehicle tracking algorithm to adapt to the slow fps. During this first long-term test, many issues of the program surfaced. The project team solved most of the issues and kept the program running. Figure 6.1 is the ramp's location on Google Map and its street view (https://goo.gl/maps/9W2N6LAMPJBTGYXv7). Figure 6.2 is a screenshot of the traffic monitoring output. Table 6.1 depicts the format of the data output. Each row is a new car in each lane detected at a specific time.

6.2 Test Case 2

The second test case was to monitor the traffic status in the road construction segment around I-70 105 mile marker. INDOT was interested in the traffic jam conditions between 1 to 7 pm daily and the ratio between the passenger cars and the trucks. Any of the three consecutive cameras with ID 175, 176, and 177 could be used.

- Camera 175 I-70/105.1 E of SR 9 (IP: 10.7.109.21).
- Camera 176 I-70/107.1 Greenfield Rest Area (IP: 10.7.109.22).

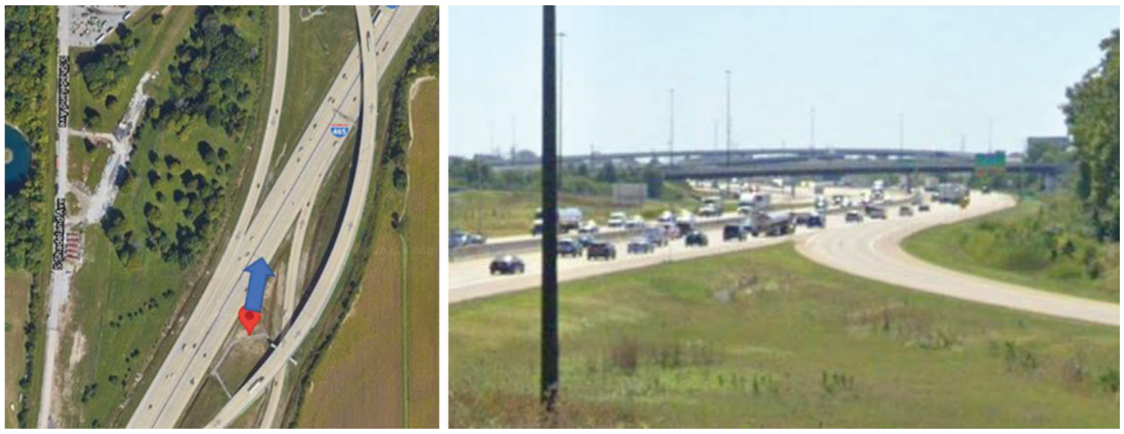
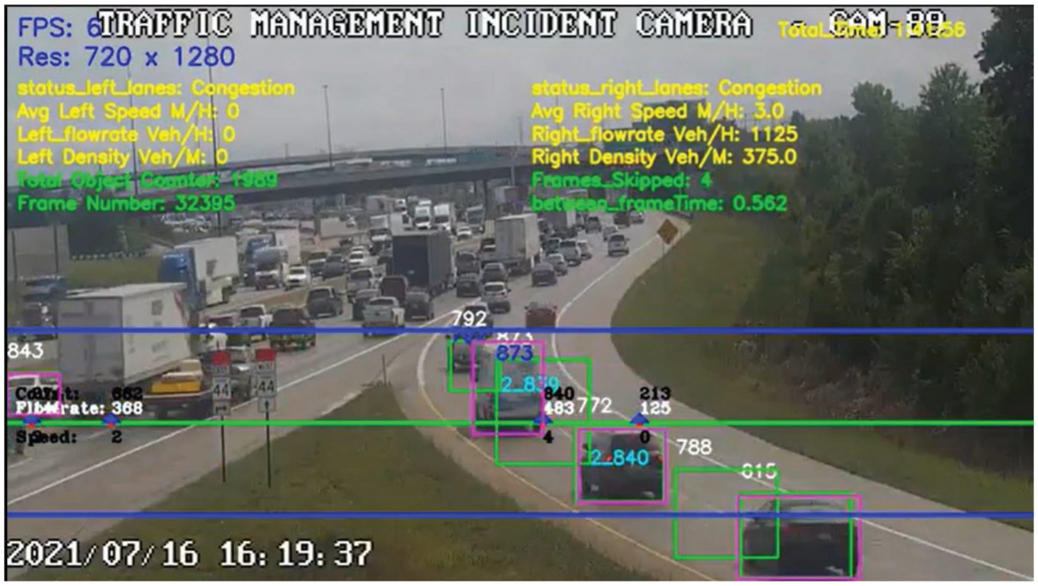


Figure 6.1 The location of the ramp on Google Map and its street view.



FPS: 6
Res: 720 x 1280
status_left_lanes: Congestion
Avg Left Speed M/H: 0
Left_flowrate Veh/H: 0
Left Density Veh/M: 0
Total Object Counter: 1989
Frame Number: 32395
status_right_lanes: Congestion
Avg Right Speed M/H: 3.0
Right_flowrate Veh/H: 1125
Right Density Veh/M: 375.0
Frames_Skipped: 4
between_frameTime: 0.562

Figure 6.2 A screenshot of the traffic monitoring output.

- Camera 177 I-70/107.1 Greenfield Rest Area (IP: 10.7.109.23).

Figure 6.3 shows the camera views from cameras 175, 176, and 177, respectively, from top to bottom. The daily test lasted over 5 weeks. Figure 6.4 shows the lane detection in the output of the environment learning. Figure 6.5 is a screenshot of the traffic monitoring. This test also revealed some situations that were not considered before. (1) When the traffic is jammed standstill, the road boundary detection and the lane direction detection in the environment learning part did not work well since they rely on the vehicle motion. (2) There are many trucks on this road. During very slow traffic or traffic jam, adjacent vehicles are close to each other, which causes occlusion of small vehicles and hence causing vehicle detection errors and tracking issues. Algorithms are still being modified to handle these situations. Meanwhile, we chose one of the cameras that the corresponding road segment does not have a standstill traffic jam. (3) During the traffic jam, vehicles were moving on the road shoulders occasionally. When this happened during the environment learning, this shoulder was learned as a lane. Therefore, a detected lane would be a false positive. Although the extra lane was detected, it will not affect the vehicle counting and flow rate calculation for each lane since few vehicles are moving on this "lane." However, it affected the decision of the traffic jam, which is based on the average flow rate of all lanes. Since the road construction caused slow traffic was a constant interest by INDOT traffic operators, the traffic operators change the camera aiming angle several times a day and sometimes doing 180 degrees pan angle change. This demonstrated the need for automatic camera angle change detection and environment relearning. The vehicle counting accuracy per lane is about 90%. The flow rate is at the same accuracy

TABLE 6.1
The format of the data output

Timestamp in yy-mm-dd hh:mm:ss	Timestamp in ms	Lane ID	Lane Car ID	Vehicle Type	Flow Rate
2021-07-16 18:00:21	1626472822	2	804	car	236
2021-07-16 18:00:21	1626472822	0	847	car	250
2021-07-16 18:00:22	1626472823	0	848	car	250
2021-07-16 18:00:22	1626472823	1	322	car	94
2021-07-16 18:00:22	1626472824	0	849	car	250
2021-07-16 18:00:22	1626472824	1	323	car	94
2021-07-16 18:00:23	1626472825	0	850	car	250
2021-07-16 18:00:23	1626472825	1	324	car	94
2021-07-16 18:00:25	1626472826	0	851	car	250
2021-07-16 18:00:26	1626472827	0	852	car	250
2021-07-16 18:00:27	1626472829	0	853	car	250
2021-07-16 18:00:32	1626472833	2	805	car	236
2021-07-16 18:00:36	1626472837	1	325	car	95
2021-07-16 18:00:41	1626472842	1	326	car	95
2021-07-16 18:00:43	1626472844	0	854	car	252
2021-07-16 18:00:45	1626472846	1	327	car	95

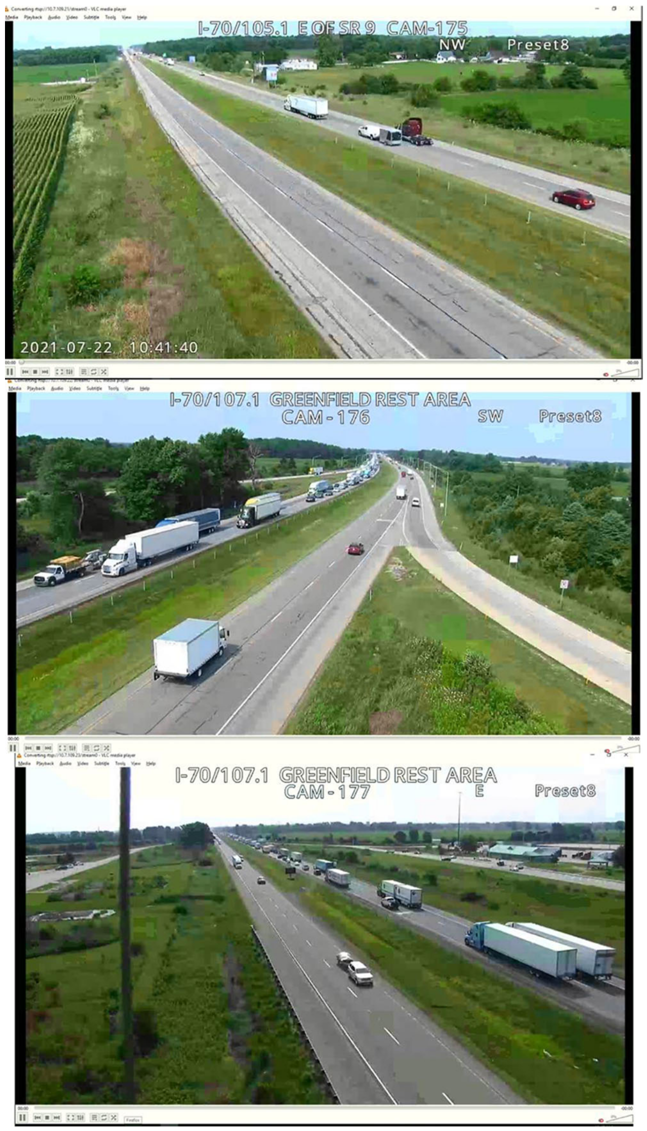

I-70/105.1 E OF SR 9 CAM-175
NW
Preset8
2021-07-22 10:41:40
I-70/107.1 GREENFIELD REST AREA
CAM - 176
SW
Preset8
I-70/107.1 GREENFIELD REST AREA
CAM - 177
E
Preset8

Figure 6.3 The camera views from cameras 175, 176, and 177, respectively from top to bottom.

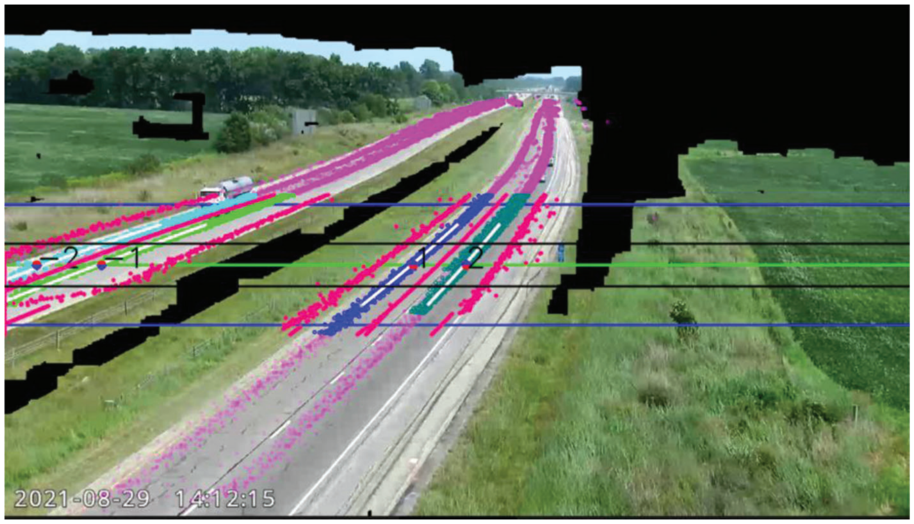


Figure 6.4 The output of the environment learning.

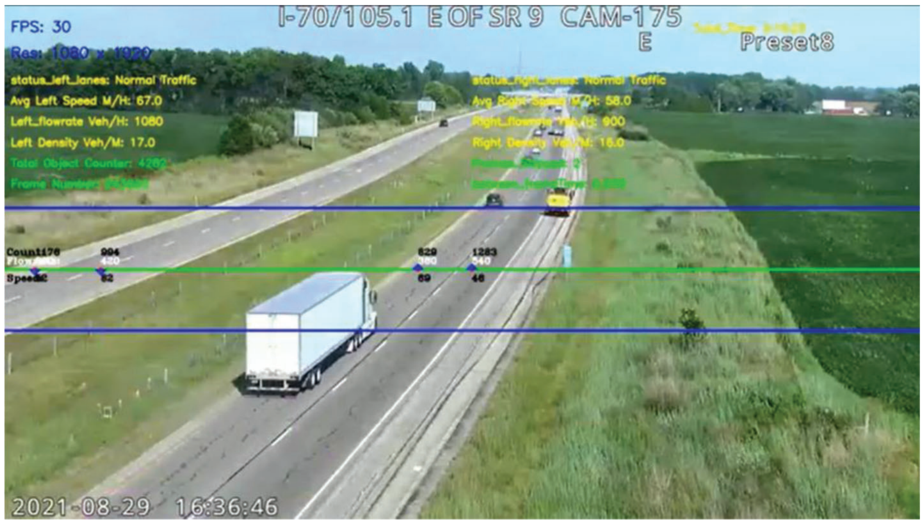

I-70/105.1 E OF SR 9 CAM-175
E
Preset8
FPS: 30
status_left_lanes: Normal Traffic
Avg Left Speed M/H: 67.0
Left_flowrate Veh/H: 1080
Left Density Veh/M: 17.0
Avg Right Speed M/H: 58.0
Right Density Veh/M: 16.0
2021-08-29 16:36:46

Figure 6.5 A screenshot of the traffic monitoring.

level. Due to the lack of distance reference, the vehicle speed detection was not accurate.

7. CONCLUSIONS AND FUTURE WORK

In this project, we have developed the traffic monitoring system and implementation system for monitoring traffic conditions using the INDOT CCTV video feeds automatically in real-time. More specifically, an AI-based deep learning algorithm provided in YOLOv4 was used for detecting vehicles on the video frames. This vehicle information was used for automatic learning the lane locations on the video frames. We also developed tracking and mapping algorithms to identify all vehicles to specific lanes and used this information to count the vehicles over specified periods and monitor the flow rate changes on each lane in real-time. The traffic jam can be automatically detected using the detected flow rate and vehicle density information. The system has been successfully tested in daytime, dawn, dusk, and well-lit roads at night. However, the system does not function well in totally dark conditions as the vehicle headlights generated hallow make vehicle detection difficult. A database was designed as the central place to gather and distribute the information generated from all camera videos. The result of traffic conditions generated from each camera video feed can be uploaded to the database in real-time. The webpage was developed to extract the information from the database and display the immediate past and current traffic status observed from each camera.

The team is currently testing the system in natural production environments daily and keeps improving the accuracy of the traffic monitoring results. Furthermore, in a follow-up research project that INDOT has approved, the research team will support INDOT to implement a small-scale system on the INDOT site and continue to enhance the functionality of the systems.

REFERENCES

Qiu, M., Chien, S., Mulay, A. A., Christopher, L., Ding, Z., Chen, Y., Sturdevant, J., & Cox, E. (2021, September). Intelligent highway lane center identification from surveillance camera video. *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. https://doi.org/10.1109/ITSC48978.2021.9564560

About This Report

An open access version of this publication is available online. See the URL in the citation below.

Chien, S., Chen, Y., Christopher, L., Qiu, M., & Ding, Z. (2022). *Road condition detection and classification from existing CCTV feed* (Joint Transportation Research Program Publication No. FHWA/IN/JTRP-2022/02). West Lafayette, IN: Purdue University. https://doi.org/10.5703/1288284317364