TECHNICAL REPORT STANDARD PAGE

1. Title and Subtitle

Improved Signalized Intersection Performance Using Computer Vision and Artificial Intelligence

2. Author(s)

Milhan Moomen, Ph.D.; Xiangyu Meng, Ph.D.;
Tonmoy Sarker; Jones B. Essuman;
Collins Addo

3. Performing Organization Name and Address

Louisiana State University
Baton Rouge, LA 70803
University of Louisiana at Lafayette
Lafayette, LA 70503

4. Sponsoring Agency Name and Address

Louisiana Department of Transportation and Development
P.O. Box 94245
Baton Rouge, LA 70804-9245

5. Report No.

FHWA/LA.26/724

6. Report Date

June 2026

7. Performing Organization Code

LTRC Project Number: 24-4SS
SIO Number: DOTLT1000515

8. Type of Report and Period Covered

Final Report
January 2024 – December 2025

9. No. of Pages

108

10. Supplementary Notes

Conducted in Cooperation with the U.S. Department of Transportation, Federal Highway Administration.

11. Distribution Statement

Unrestricted. This document is available through the National Technical Information Service, Springfield, VA 21161.

12. Key Words

Intelligent transportation system; artificial intelligence; deep learning; multi-object tracking

13. Abstract

Signalized intersections are critical points in urban transportation networks where congestion, delays, and safety risks are most prominent. Traditional approaches for performance evaluation rely on manual field counts, loop detectors, or expensive infrastructure-based systems, which are often limited in accuracy, scalability, and adaptability. This project explored the use of computer vision and artificial intelligence to develop automated tools for intersection performance analysis. Three primary methods were investigated: (1) a vehicle counting framework based on detection–tracking–counting pipelines; (2) a queue detection approach integrated with traffic light state recognition; and (3) pedestrian behavior and interaction. Vehicle counting was implemented using a virtual line-crossing strategy combined with advanced detection and tracking models, enabling accurate measurement of turn movements across multiple lanes. Queue detection, in turn, was achieved by associating detected vehicles within lane-

specific regions of interest with real-time traffic light states, providing insights into demand and delay at intersections. The pedestrian-vehicle interaction is based on trajectory extraction at the intersection. All methods were tested on drone- and camera-based video datasets collected from Louisiana intersections. Results demonstrate that the proposed algorithms achieved high accuracy, robustness to environmental variations, and efficiency suitable for near-real-time applications. A user-friendly graphical interface was also developed to allow engineers to apply these methods to raw video data, facilitating data-driven decisions for signal timing, intersection design, and congestion mitigation. The study highlights the feasibility of AI-based computer vision systems as cost-effective, scalable, and reliable alternatives for traffic performance monitoring.

Project Review Committee

Each research project will have an advisory committee appointed by the LTRC Director. The Project Review Committee is responsible for assisting the LTRC Administrator or Manager in the development of acceptable research problem statements, requests for proposals, review of research proposals, oversight of approved research projects, and implementation of findings.

LTRC appreciates the dedication of the following Project Review Committee Members in guiding this research study to fruition.

LTRC Administrator/Manager

Julius Codjoe, Ph.D., P.E.
Special Studies Research Administrator

Members

Andre′ Fillastre
John Broemmelsiek
Jennifer Bonnette
Duc Ngo
Lei Wang
Benjamin Boudreaux

Directorate Implementation Sponsor

Chad Winchester, P.E.
DOTD Chief Engineer

Improved Signalized Intersection Performance Using Computer Vision and Artificial Intelligence

By
Milhan Moomen, Ph.D.
Xiangyu Meng, Ph.D.
Tonmoy Sarker
Jones B. Essuman
Collins Addo

Louisiana Transportation Research Center
4101 Gourrier Avenue
Baton Rouge, LA 70808

LTRC Project No. 24-4SS
SIO No. DOTLT1000515

conducted for

Louisiana Department of Transportation and Development
Louisiana Transportation Research Center

June 2026

Abstract

Signalized intersections are critical points in urban transportation networks where congestion, delays, and safety risks are most prominent. Traditional approaches for performance evaluation rely on manual field counts, loop detectors, or expensive infrastructure-based systems, which are often limited in accuracy, scalability, and adaptability. This project explored the use of computer vision and artificial intelligence to develop automated tools for intersection performance analysis. Three primary methods were investigated: (1) a vehicle counting framework based on detection–tracking–counting pipelines; (2) a queue detection approach integrated with traffic light state recognition; and (3) pedestrian behavior and interaction. Vehicle counting was implemented using a virtual line-crossing strategy combined with advanced detection and tracking models, enabling accurate measurement of turn movements across multiple lanes. Queue detection, in turn, was achieved by associating detected vehicles within lane-specific regions of interest with real-time traffic light states, providing insights into demand and delay at intersections. The pedestrian-vehicle interaction is based on trajectory extraction at the intersection. All methods were tested on drone- and camera-based video datasets collected from Louisiana intersections. Results demonstrate that the proposed algorithms achieve high accuracy, robustness to environmental variations, and efficiency suitable for near-real-time applications. A user-friendly graphical interface was also developed to allow engineers to apply these methods to raw video data, facilitating data-driven decisions for signal timing, intersection design, and congestion mitigation. The study highlights the feasibility of AI-based computer vision systems as cost-effective, scalable, and reliable alternatives for traffic performance monitoring.

Acknowledgements

The research team gratefully acknowledges the support of the Louisiana Transportation Research Center (LTRC) and the Louisiana Department of Transportation and Development (DOTD) in carrying out this project. We extend our appreciation to the Special Studies section of LTRC for their continued assistance throughout the study.

We would also like to thank the traffic engineers and the UAV team from both DOTD and LTRC for their efforts in video capture, as well as the engineers, operators, and technicians who facilitated traffic video data collection and transfer to the research team.

Finally, we are grateful to the Project Review Committee for their valuable feedback and to all individuals who contributed, whether directly or indirectly, to this project.

Implementation Statement

The findings of this report demonstrate that the AI algorithms developed for turn movement count, vehicle queue length estimation, and traffic signal recognition achieve high accuracy when compared to ground truth data. These results provide transportation engineers at the Louisiana Department of Transportation and Development (DOTD) and Louisiana Transportation Research Center (LTRC) with critical information that can be used to support data-driven decision-making.

In practice, this information can enhance both short-term and long-term planning by improving the management of intersections, optimizing traffic flow, and reducing congestion. By automating video-based analysis, the approach reduces reliance on manual observation and data collection, thereby saving time and resources for engineers, technicians, and operators.

Further, the use of UAV-based aerial video capture expands monitoring capabilities beyond fixed roadside cameras, enabling rapid deployment in locations where traditional infrastructure is unavailable. The findings provide consideration for its adoption and integration with existing DOTD traffic management systems to provide further insights.

Table of Contents

List of Tables

List of Figures

Introduction

The increasing complexity of modern transportation systems requires innovative approaches to monitor and optimize intersection performance. Computer vision and artificial intelligence (AI) offer promising solutions in this domain. As primary data sources, cameras provide rich visual information for traffic analysis. By extracting traffic information from video data, computer vision enables the development of intelligent systems for traffic management.

The goal of this project was to utilize computer vision and AI to enhance traffic flow, safety, and efficiency at signalized intersections and provide graphical user interface (GUI) tools for interactive analysis. By automating the collection and analysis of traffic data, this research aimed to provide transportation engineers with valuable insights into road user behavior. These insights can inform data-driven decisions on intersection design, signal timing optimization, and overall traffic management strategies, ultimately leading to safer, more efficient transportation networks.

Signalized intersections are among the most critical components of urban transportation networks, serving as the primary points where traffic flows converge, diverge, and interact. These intersections are also where congestion, delays, and conflicts among road users most frequently occur, making their efficient operation vital to overall roadway performance. As transportation systems grow increasingly complex, the need for innovative tools that can provide accurate, timely, and detailed performance measures has become more pressing. Traditional data collection methods such as loop detectors, pneumatic tubes, and manual field counts offer limited resolution, require substantial human effort, or impose high maintenance costs. Moreover, such methods often fail to capture the full dynamics of traffic behavior at intersections, including queue formation, turn movements, and the interactions between vehicles and signal phases.

Advances in computer vision and AI now present a powerful opportunity to address these limitations. When combined with AI-based detection, tracking, and classification algorithms, cameras can non-intrusively capture rich information about traffic conditions and automatically extract key performance measures. Unlike traditional sensors, vision-based systems provide spatial and temporal detail, enabling a deeper understanding of how vehicles move, queue, and respond to signal changes. This capability is particularly valuable for performance-based approaches to traffic signal operations, management, and design, which are increasingly emphasized by transportation agencies seeking cost-effective and data-driven solutions.

The focus of this research was on three interrelated methods; the first two focused on vehicle counting and queue detection with traffic light state recognition, while the third focused on pedestrian behavior and interaction with other vehicles. Vehicle counting provides insights into turn movements and approach demands at intersections, information that is essential for signal timing optimization, intersection capacity analysis, and traffic demand forecasting. By implementing a detection-tracking-counting pipeline that uses advanced deep learning detectors and multi-object tracking algorithms, this study automates the traditionally labor-intensive task of turn movement counts. Complementing this, queue detection enhanced with accurate traffic light state recognition provides real-time measures of congestion and delay. Together, these two methods capture both the demand side (e.g., vehicle arrivals and turn patterns) and the operational side (e.g., queues and delays at traffic signals), thereby offering a holistic view of intersection performance.

The significance of this work extends beyond the development of algorithms. By integrating these methods into a user-friendly graphical user interface (GUI), the project ensures that traffic engineers and practitioners can easily apply the tools to real-world video data. The system enables interactive region of interest (ROI) definition, lane-to-signal mapping, and automated output of turn counts, queue lengths, and signal phase information. Such functionality not only reduces reliance on manual data collection but also empowers transportation agencies to make data-driven decisions on signal timing adjustments, congestion mitigation, and intersection safety improvements.

This research builds upon recent national and international efforts to modernize traffic monitoring through video analytics and drone-based data collection. Drone footage provides a unique vantage point for simultaneously capturing queue dynamics and overall traffic movements across large intersections. When combined with AI, this data source becomes a powerful asset for performance evaluation. Through this project, the Louisiana Department of Transportation and Development (DOTD) aims to leverage these innovations to enhance intersection performance assessment, providing a replicable framework adaptable across diverse traffic conditions and intersection types.

In summary, the introduction of AI-based vehicle counting and queue detection, combined with traffic light recognition, addresses a critical need for accurate, scalable, and cost-effective traffic performance measures. By bridging the gap between raw video data and actionable insights, this research supports the development of safer, more efficient, and more sustainable urban transportation systems.

Literature Review

Vehicle Detection, Count, and Queue Length Estimation

A limited amount of research has been done in the area that collectively spans vehicle detection, counting, and queue length estimation. A survey of relevant works in this regard is presented below. A summarized breakdown of the algorithms and approaches is provided in Table 1 and Table 2 for queue detection and queue length estimation methods, respectively. Broadly, the reviewed work approaches can be categorized into:

- Image Processing Algorithms
- Image Processing with Machine Learning and Tracking Algorithms
- Deep Learning with Tracking Algorithms

Conventional Image-Processing Algorithms

Early research in vehicle detection and queue estimation relied primarily on conventional image-processing techniques developed before the widespread adoption of machine learning and deep learning. These methods typically employ operations such as background subtraction, thresholding, and edge detection to extract traffic information from video streams. Seenouvong et al. used background subtraction, hole filling, and thresholding to detect moving vehicles [1]. Vehicle counting was performed by tracking the centroid of each detected vehicle, with its presence in a virtual zone incrementing the count. Qi et al. implemented an image-processing-based approach for vehicle detection and queue length measurement using background subtraction [2]. Queue estimation began with identifying a lane median line to determine the coordinates of the head and tail of the vehicle queue. A region of interest (ROI) on the road was processed through transformations, filtering, thresholding, and edge detection. The Progressive Probabilistic Hough Transform (PPHT) was used for lane line detection. After image processing, the continuous white region along the median line represented the queue, while the following black region indicated the queue's end. To compute the true queue length, 2-D coordinates of the queue head and tail were transformed into 3-D world coordinates using camera calibration equations and parameters. Zanin et al. also proposed an image-processing method for vehicle and queue detection using edge detection rather than background subtraction [3]. For motion detection, a frame-difference technique with a motion probability threshold was used; if the motion probability in the region of interest fell below the threshold, a queue was detected. The approach was tested in a six-month field deployment for queue monitoring. However, false detections

caused by shadows of stationary vehicles occasionally led to overestimation of queues in adjacent lanes. Some studies later incorporated machine learning techniques to enhance the accuracy and robustness of these traditional methods.

Image-Processing Algorithms with Machine Learning

Machine learning, particularly classification-based algorithms, enhances traditional image-processing techniques by enabling multi-class object detection and improved feature discrimination. Hybrid approaches integrate conventional image processing for feature extraction with machine learning for classification, improving robustness under varying environmental conditions. Jiang et al. combined image processing and machine learning for vehicle detection and counting [4]. The foreground was obtained by subtracting the current frame from the background and binarizing it through thresholding, followed by morphological erosion and dilation to preserve relevant features. To account for vehicle size variation with distance, a transformation was applied to scale distant vehicles before recalculating the foreground. A lane congestion ratio, R, defined as the ratio of lane background to foreground area, was used to detect congestion; if R fell below a threshold, congestion was declared. The queue length was then estimated by interpolating between the maximum and minimum R values, using a lane adjustment factor. Vehicle detection employed cascaded Gentle AdaBoost classifiers with Local Binary Pattern (LBP) features, achieving lower mean relative error and higher count accuracy than traditional AdaBoost. Another approach for traffic queue estimation using UAV footage was proposed by Yang et al. [5], employing the Viola–Jones and Support Vector Machine (SVM) algorithms for detection. Queue length, L, in pixel coordinates was computed as the difference between the maximum and minimum x-coordinates of detected vehicles within a defined region of interest parallel to the road. This method assumes an aerial, approximately top-down camera view. Beyond detection and counting, some extended frameworks incorporate tracking to minimize redundant counts and deep learning models to further enhance detection accuracy.

Deep Learning with Tracking Algorithms

For large datasets, deep learning methods have become the dominant approach for computer vision-based object detection because they can learn rich, hierarchical features directly from data and handle substantial variation. With advances in computing hardware, these models can be trained and deployed at scale, enabling robust detection and tracking in complex scenes. In intelligent transportation, deep learning combined with tracking has therefore emerged as a widely adopted computer vision strategy.

For example, Al-Ariny proposed a vehicle detection and counting framework that integrates mask Region-Based Convolutional Neural Network (R-CNN) with the Kanade–Lucas–Tomasi (KLT) feature tracker [6]. Mask R-CNN, a two-stage detector, first generates region proposals, then performs classification and bounding box regression; the video source in this work was a fixed surveillance camera. The network detects vehicles, and their salient corners are tracked to ensure each vehicle is counted only once. The authors report an accuracy of 97.9%, with reduced performance on lower-resolution inputs due to missed detections. Faster R-CNN was also employed for vehicle detection by Zinanyuca et al. [7], demonstrating the broader applicability of two-stage detectors within traffic monitoring pipelines. The bounding boxes of the detected vehicles are used to calculate their centroids. The road section is divided into tracks (lanes), and the coordinates are extracted from them. Given the vehicle's bounding box, the specific lane it is in is determined. Next, the queue length is computed by incrementing it by the actual vehicle length plus an estimated inter-vehicle distance. For example, a length of 4.8 meters is used for a car, and 10 meters is used for a bus. An alternative to the two-stage deep learning algorithm is a single-stage detection method, which has been found to have a relatively better inference speed.

Al Okaishi et al. developed a real-time queue detection and length estimation system using a frame-by-frame difference method for motion detection and a modified Single Shot Multibox Detector (SSD) for vehicle detection and counting [8]. SSD performs object detection in a single forward pass, enabling faster inference compared to R-CNN [9]. For each lane, a virtual motion area is defined, and mean pixel intensity differences are compared against a threshold to determine motion status. When no motion is detected, the queue length is incremented based on the sum of the heights of vehicle bounding boxes in the queue region, and reset to zero when motion resumes at the queue head. To ensure stability, short-term transitions between motion states are ignored using a threshold time. Evaluation on the MIO-TCD dataset [10] achieved an F1-score of 0.84, outperforming Faster R-CNN and YOLOv2 in detection and counting accuracy. An enhanced SSD framework, termed Enhanced-SSD, was later proposed to improve small-vehicle detection in UAV imagery by Zhu et al. [11]. The approach replaces the VGG backbone with ResNet for better feature extraction and uses bounding box outputs for motion estimation and vehicle tracking via a Kalman Filter (KF). Vehicle counting is performed based on unique tracking IDs and classification by vehicle type. Beyond SSD-based approaches, improved versions of the YOLO architecture have also been widely adopted alongside advanced tracking algorithms for vehicle detection and queue length estimation.

Vehicle counting using YOLOv3 for object detection and the ORB (Oriented FAST and Rotated BRIEF) algorithm for tracking was proposed by Song et al. [12]. YOLOv3, a single-

stage detection framework, achieves high detection speed and accuracy, while ORB combines FAST keypoint detection with BRIEF descriptors and orientation encoding to enhance robustness against rotation. The algorithm detects and matches keypoints across frames, incrementing the vehicle count each time a detected object crosses a virtual line. Real-time vehicle detection from UAV aerial videos was implemented on hardware using the YOLOv3 model, with the total detected instances serving as the vehicle count [13]. For signalized intersections, Bui et al. combined YOLOv3 with DeepSORT to perform detection and tracking [14]. DeepSORT extends the Simple Online and Realtime Tracking (SORT) algorithm by integrating a re-identification network that uses object features as a tracking matching criterion.

A method for vehicle detection, counting, and queue length estimation using UAV footage was developed by combining YOLOv3 with the Sparse Pyramid Lucas–Kanade (SP L–K) tracking algorithm by Mansour et al. [15]. The SP L–K algorithm, a feature-based optical flow technique, tracks selected keypoints between successive frames using a multi-scale (i.e., pyramidal) Lucas–Kanade approach for improved motion accuracy. Vehicle counting is performed when the centroid of a detected vehicle crosses a predefined virtual line between frames. Queue estimation employs two rectangular zones: Zone A (incoming traffic) and Zone B (near the signal). A queue is detected when no motion occurs in Zone B, and its length is estimated by extracting pixel coordinates from the queue head to tail. However, the method does not specify how pixel distances are converted to physical measurements. Holla et al. employed an alternative approach, YOLOv3 for vehicle detection, integrating Histogram of Oriented Gradients (HOG) and Local Binary Patterns (LBP) to reduce redundant counts [16]. HOG captures gradient features across vehicle regions in successive frames, while LBP analyzes texture. Extracted features from feature vectors whose Euclidean distances between frames are compared against a threshold to identify new or recurring vehicles, thereby minimizing duplicate counts.

An integrated approach combining YOLOv4 and DeepSORT algorithms was developed for vehicle detection and queue length estimation by Umair et al. [17]. YOLOv4, an enhanced version of YOLOv3, performed vehicle detection, while the Deep Simple Online and Real-time Tracking (DeepSORT) algorithm handled tracking. Queue detection occurred when no motion was detected between consecutive frames, identifying the first and last stationary vehicles as queue boundaries. The pixel distance between these vehicles was converted to a queue length, assuming each vehicle is 5 meters long. Reported results showed over 90% accuracy in vehicle counting across three video streams. Limitations include manual lane assignment, partial occlusion due to camera angle, and the use of a single detection class with a fixed vehicle length, all of which limit accurate queue estimation. A modified YOLOv5

framework, FFCA-YOLO, was later proposed for small-object detection in UAV imagery, where vehicles appear smaller than in a ground-based view, by Zhang et al. [18]. Incorporating feature enhancement, fusion, and context awareness (FFCA), the model achieved 0.748 mAP50 on datasets such as VEDAI [19], outperforming YOLOv8m (0.686 mAP50), though it focused solely on vehicle detection.

Transformer-Based Algorithms

Recently, transformer-based architectures such as the Vision Transformer and DETR (End-to-End Object Detection with Transformer) have gained traction for object detection due to their ability to treat detection as a direct set prediction problem [20]. However, many face challenges in real-time applications because of high computational demands. More recent variants, such as RT-DETR [21], have shown improved efficiency and competitive performance with established detectors. Hybrid CNN–Transformer approaches have also been explored to enhance efficiency. For vehicle detection, CNN-Transformer-based detectors for aerial images have demonstrated strong applicability to traffic monitoring [22] [23]. Ye et al. proposed a Convolutional Multihead Self-Attention mechanism that replaces the position-linear projection of traditional MHSA with a convolutional projection, improving recognition of occluded objects and reducing computational cost [22]. Despite this progress, recent YOLO versions (YOLOv9–12) continue to demonstrate strong performance, better or on par with transformer models, maintaining YOLO as the most widely used deep learning framework for object detection, including aerial vehicle detection [24]. A summary of the queue detection and queue-length estimation algorithms are provided in Table 1 and Table 2, respectively.

Based on the literature review, a combination of a suitable YOLO detection model with a tracking algorithm such as DeepSORT, SORT, KLT, or Bytetrack and a custom algorithm for counting, queue detection, and length estimation shows potential as a feasible approach for the LTRC project. However, this approach may evolve as new findings or conditions are encountered during implementation.

Table 1. Vehicle queue detection

Ref.	Year	Methods	Traffic video / image source
[2]	2018	Employ background subtraction; the region of continuous white pixels indicates a head (located at the stop line) and the region of continuous black pixels indicates the tail of the queue.	Surveillance camera (intersection)

Ref.	Year	Methods	Traffic video / image source
[17]	2021	Track the motion of vehicles using the Deep-SORT algorithm; no motion of vehicles indicates queue status.	Surveillance camera (signalized intersection)
[3]	2023	Frame (pixel) difference method with a probability threshold is used to detect vehicle motion; no motion indicates a queue.	Surveillance camera (highway)
[8]	2021	Frame pixel difference method with a time threshold is used to detect the motion of the vehicle in the virtual queue head.	Surveillance camera (Miovision traffic camera dataset)
[4]	2019	Lane congestion detection using a threshold of congestion ratio is used to indicate queue state.	Surveillance camera (highway)
[15]	2020	Utilize a two-zone approach: Zone B after the signalized stop and Zone A, following Zone B. The queue is detected if no motion is in Zone B, and the queue in Zone A if there is no motion in A and B.	UAV

Table 2. Vehicle queue length estimation

Ref.	Year	Methods
[2]	2018	The queue length is derived from a 2-D pixel to 3-D world coordinate projection. Camera calibration implemented. ($Q_{\text{length}} = \sqrt{(X_H - X_T)^2 + (Y_H - Y_T)^2}$)
[17]	2021	($Q_{\text{length}} = L_{\text{AV}} \times N_B$): L_{AV} where average vehicle length (5m used): N_B number of tracked vehicle bounding boxes.
[8]	2021	Increment the queue length by the height of the bounding boxes. H_{B_i} is the height of the bounding box ($Q_{\text{length}} = \sum_{i=1}^{N} H_{B_i}$).
[4]	2019	Linear interpolation. ($Q_{length} = k_i \left[N_{\min} + \frac{(N_{\max} - N_{\min})(R_i - R_{\min})}{1 - R_{\min}}\right]$) where Nmin: number of vehicles when R is minimum, Nmax: number of vehicles when R is maximum, k_i: adjustment factor.
[7]	2020	Incrementing the queue length by the length of the detected vehicle class and inter-vehicle distance. ($Q_{\text{length}} = \sum(L_{\text{CV}} + D_{\text{VV}})$), where L: length of a vehicle, D: inter-vehicle distance.
[5]	2019	Finds a minimum circumscribed rectangle covering all vehicles in the queue ($Q_{\text{length}} = P_{\text{UL}}(x) - P_{\text{LR}}(x)$) where PUL and PLR: pixel coordinates of the minimum rectangle. x: horizontal coordinate along the road.

Ref.	Year	Methods
[15]	2020	Extracting pixel coordinates difference of the region from the queue head to the tail of the queue.

Trajectory Extraction

Coordinate Transformation

Coordinate transformation is typically required during the object tracking step. This transformation is necessary to map the coordinates of detected vehicles from the camera's 2-D image plane to a global coordinate system, such as geographic coordinates (e.g., latitude and longitude) or a local coordinate system representing the road network. Moreover, intersection camera videos often capture scenes from an angle, leading to perspective distortion. Therefore, distortion correction is required. A homography transformation is a mathematical technique used in computer vision to map points from one plane to another. The transformation describes the relationship between points in two different images of the same scene and allows geometrically aligned corresponding points in these images, if a planar surface exists within that scene. It helps to map the positions of vehicles accurately across different frames, even when they appear at different locations due to changes in perspective. By mapping the positions of vehicles in each frame to a common reference plane, their trajectories can be extracted reliably. Calibrating the camera beforehand can improve the accuracy of the homography matrix.

Trajectory Extraction

After tracking the vehicles, their trajectories need to be extracted. Trajectory extraction involves analyzing the movement patterns of vehicles over time to obtain their paths within the intersection. This step often involves smoothing the trajectories and handling occlusions or missing data. The Rauch-Tung-Striebel (RTS) algorithm is a technique that has been commonly used for more accurate and robust tracking to compensate for sensor noise and occlusions and to smooth the extracted trajectories [25] [26] [27]. Unlike the KF, which works in a forward pass using only past measurements for the current state estimate, the RTS algorithm operates in two passes: (1) a 'Forward Pass' that estimates the state and its uncertainty based on the system model and measurements up to the current time step (similar to KF), and (2) a 'Backward Smoothing Pass' that leverages measurements from future time steps (in addition to the ones used in the forward pass) to refine the state estimates for all time steps, providing a smoother and more accurate overall trajectory. By incorporating information from future measurements, the RTS algorithm corrects errors that accumulate in the forward pass of the KF, leading to more accurate state estimates across all time steps.

Post-processing techniques may be applied to refine the extracted trajectories and analyze various parameters, such as vehicle speed, acceleration, lane changes, and interactions between vehicles.

Table 3. Intersection camera-based trajectory extraction

Ref.	Hardware	Detection	Trajectory extraction	Smoothing (error filtering)	Trajectory prediction/ evaluation
[28]	Multiple intersection cameras	ResNet-50	Re-ID	-	-
[29]	Multi-camera	YOLOv4	IoU tracker + KF	Linear interpolation, regression	CNN
[30]	Intersection camera	R-CNN	KF	-	Minimum jerk method, Monte Carlo
[27]	Camera, radar	YOLOv3	KF	RTS	dGPS fitted car trajectory
[25]	Monocular camera	Mask-RCNN	IoU tracker	RTS	Simulation (CARLA)
[31]	Intersection camera	Gaussian mixture models (GMM)	KF	-	LSTM and DNN
[26]	Intersection camera	Background subtraction	KF	RTS	-

Table 4. Drone-based datasets for trajectory extraction at road intersections

Dataset	Objects	Detection/ segmentation	Lane identification	Coordinate transform	Tracking and trajectory extraction
CitySim	Vehicles	SIFT based stabilization, Mask-RCNN	-	Homography transformation	Channel and spatial reliability tracker (CSRT)
HIGH-SIM	Vehicles	Gaussian mixture-based background foreground segmentation, YOLOv3	Monte Carlo	ORB feature matching, FAST feature, BRIEF descriptor, RANSAC perspective transformation	Features matching from Detection
rounD	Vehicles, Pedestrians, Bicyclists	U-Net semantic segmentation	Monte Carlo	Homography transformation	KF
inD	Vehicles, Pedestrians, Bicyclists	U-Net semantic segmentation	Monte Carlo	Homography transformation	KF

Dataset	Objects	Detection/ segmentation	Lane identification	Coordinate transform	Tracking and trajectory extraction
HighD	Vehicles	U-Net semantic segmentation	-	Homography transformation	KF

Model-Based Traffic Signal Recognition

Most of the model-based TSR studies extensively use raw color or pixel information, BLOB analysis, and morphological filters.

Color Segmentation

Color is a prominent characteristic in images. As traffic signals are illuminating objects, pixel information can be directly exploited to distinguish them from other objects. Although RGB is a primary color space choice, it does not decouple the luminance and chrominance components of color. Hence, RGB is less robust to illumination, exposure, and color distortion. Several other color spaces are prominently used for ROI generation and feature extraction, such as HSV (Hue, Saturation, Value), HSI (Hue, Saturation, Intensity) [32] [33] [34] [35], HSL (Hue, Saturation, Lightness), YCbCr [36] [37], LUV [38] [39], YUV [36] [40], and CIELab [41].

Morphological Analysis

Color segmentation-based TSR could lead to false predictions due to the presence of other illuminating objects in the input images. Therefore, various structural characteristics of traffic signal heads, such as shape, size, texture, and aspect ratio of traffic light housing and bulbs, are often used in addition to color segmentation. BLOB and spot-light detection are two frequent techniques that involve the morphological analysis of traffic signals.

A BLOB is a region of connected pixels with similar color intensity. In BLOB analysis, the output pixel values are set based on the comparison between a corresponding pixel and its neighboring pixels in an input image. Diaz-Cabrera et al. utilized a shape filter for TL detection [42], while Li et al. used a filled circle filter [36]. Hough transform is a useful technique for feature extraction and has been adopted by many TSR studies. Chiang et al. applied Hough transform to the edge map of a Laplacian edge detection filter (Sobel Kernel) [43]. For detecting circular objects, such as traffic light bulbs, the circular Hough transform can provide better results than the traditional Hough transform algorithm. Therefore, the circular Hough transform was applied in several studies [44, 45, 46].

The spot-light detection method highlights the regions of an image that are brighter than their surroundings. Top-hat filter is a common spot-light detection technique used in several TSR studies [47, 48, 49]. Fast Radial Symmetry Transform (FRST) is another algorithm used for

detecting circular dark or bright objects based on a given radius. FRST was applied to detect traffic light bulbs of a given radius. Location and context of traffic signals can be used in addition to shape for prior feature learning. A sliding-window-based Aggregate Channel Features (ACF) method for fusing three separate detectors is used: a multi-size detector, a fuzzy detector, and a bright bulb detector.

Classic Machine Learning-Based Traffic Signal Recognition

While simple classifiers can be built upon extracted object features, they often perform weakly. In classic ML-TSR pipelines, various extracted features from candidate ROIs are fed to classifier algorithms. The classic ML models optimize their parameters based on the training data. Therefore, the signal state recognition task typically consists of the following stages: feature extraction, classifier training, and classifier application to new samples.

Support Vector Machine (SVM) is a popular ML classifier that identifies an optimal hyperplane in a multi-dimensional space to maximize class separation. Many classic ML-based TSR studies used SVM with features extracted from regions of interest (ROI), including HOG [35], Dense HOG, and other morphological features [46]. HOG efficiently captures edges, textures, and local shapes, and is robust to illumination, orientation, and scale variations [50, 51]. Kim et al. applied SVM to night-time images using basic, brightness-based, and geometry-based moment features [52]. SVM can be combined with spatial texture layout extraction, color segmentation, and a genetic algorithm for ellipse detection.

Boosting is a common machine learning technique in TSR classification and a form of ensemble learning, where multiple weak learners are combined into a strong predictive model. AdaBoost, a popular boosting algorithm, integrates weighted weak classifiers into a single strong classifier and has been widely used in ML-TSR studies [36]. The authors proposed a three-stage method using color segmentation, BLOB shape filtering, and an adaptive multi-classifier with Haar-like features. Gong et al. applied the CAMSHIFT algorithm with color histograms to reduce false negatives [53]. Haltakov et al. used color, texture, and geometry-based segmentation with a tracking algorithm for temporal consistency [54]. Additionally, a multi-phase detection approach incorporating a Kalman filter-based tracker has been proposed, while other researchers have leveraged Gabor wavelet transforms combined with improved 2-D ICA for feature extraction and dimensionality reduction.

Classic ML algorithms for traffic signal state estimation in the literature can overcome many of the limitations of model-based algorithms. Compared to traditional image processing algorithms, classic ML classifiers, such as SVM and AdaBoost, can improve the accuracy and robustness of the detection model to a great extent. However, one of the downsides of the classic ML classifier is that the detection speed could be slower compared to many traditional

image processing algorithms. Additionally, the classifier algorithms require a training data set, although most training pipelines are usually fast.

Deep Learning-Based Traffic Signal Recognition

Deep learning is a specialized branch of machine learning that involves the use of Artificial Neural Networks (ANNs) to model and solve complex tasks. ANNs are composed of multiple layers of interconnected nodes or artificial neurons categorized as input, output, and hidden layers. Typically, a network is referred to as deep if it has more than one hidden layer between the input and output layers. CNN is a type of deep learning framework commonly used for visual data analysis tasks, such as image classification and object detection. CNNs are well-suited for recognizing visual patterns with minimal pre-processing and no manual feature engineering, providing an end-to-end computer vision pipeline that propagates from the input image to the desired output while extracting meaningful feature representations directly from the raw pixels of the training images. In the most common type of CNNs, feature extraction from an input image is executed in a hierarchical manner, meaning the network extracts more complex features as the network progresses.

In general, the greater the number of convolutional layers, the more complex features the network can learn. Convolutional layers apply a series of filters, which are matrices of weights applied to a small region of the input image. Each convolutional layer outputs a feature map, which is a map of the activation functions of the filters. Usually, different filters are applied to extract different features, such as edges, textures, and shapes. Therefore, a CNN model that is fed with traffic signal images is able to extract features relevant to traffic signals, such as the shape and orientation of the signal heads and bulbs, their colors (i.e., Red, Green, or Yellow), and background patterns.

For TSR applications, common challenges such as varying illumination and occlusion can be addressed with rich training data. Automated feature map generation from input data makes CNNs somewhat invariant to illumination variation, as various training samples are fed to the network. Various data augmentation techniques implemented in TSR studies, including photometric distortions (e.g., brightness, contrast, hue, saturation, and noise adjustments) [55], geometric distortions (e.g., random scaling, cropping, flipping, and rotating), and techniques to reduce bias in the semantic distribution (e.g., random erase, grid mask, mosaic) [56, 57], also proved effective in this regard.

Instead of training CNN models from scratch, most DL object detection algorithms use large image classifier models as their backbone network, which are already pre-trained on massive diverse image data sets, such as ImageNet [58] and MS COCO [59]. By removing the last few layers of the backbone network, a rich feature map at low spatial resolution can be

obtained. Thereafter, these rich feature maps can be repurposed for use in object detection frameworks. Among the successful earlier CNN architectures, AlexNet achieved unprecedented accuracy and introduced many state-of-the-art DL methods, such as pooling after convolution design, dropout, ReLU activation function, data augmentation, and GPU parallelization. Among other notable CNN classifiers, VGGNet introduced a very deep architecture [60], and InceptionNet (GoogLeNet) introduced variable-size filters [61]. The ResNet framework fixed the vanishing gradient issue by introducing shortcut connections for the cross-layer connections [62]. A brief history of development and a comprehensive comparison among the most dominant CNN image classifiers can be found in studies conducted by Jiao et al. [63] and Liu et al. [64].

Based on the working principle, DL object detection algorithms can be categorized into two-stage and single-stage detectors. Two-stage algorithms break down the detection task into two segments: (1) ROI proposal and (2) classification. These algorithms typically scan every region of the image to determine whether a specific region contains an object of interest. The output is usually fed to a separate CNN for classification. Two-stage object detector algorithms present highly accurate detection results. However, their high computational cost and slow inference speed make them unsuitable for real-time applications. Some of the most popular two-stage algorithms are Region-based CNN (R-CNN) and its variants, such as Fast R-CNN, Faster R-CNN, and Mask R-CNN, as well as Region-based Fully Convolutional Networks (R-FCN). Kim et al. compared the performance of Faster RCNN with Inception-ResNet-v2, Faster RCNN with ResNet-101, and R-FCN with ResNet-101, and the first model produced the best detection results for small traffic signals [65]. Faster RCNN based on InceptionNet-v2 was used by Janahiraman et al. [66] and Kulkarni et al. [67], where the former paper achieved up to 97% detection accuracy. Single-stage DL algorithms perform the detection and classification tasks simultaneously through a single network. They are computationally more efficient than two-stage algorithms and are therefore preferred for real-time use cases.

Single Shot Multibox Detection (SSD) and You Only Look Once (YOLO) are two single-stage detector algorithms frequently adopted in TSR studies. The original SSD algorithm was built on the VGG-16 network [68], and later, ResNet was adopted. In SSD, additional multibox convolution layers are added at the end of the base layer to improve multi-scale detection. The original SSD VGG-16 backbone network was replaced by Inception-v3 in the TL-SSD algorithm [69], and by MobileNet-v2 for a combined traffic signal and sign detection. TL-SSD authors adopted prior box generation to allow a smaller stride in the network layers in order to improve smaller object detection, which reached up to a 95% Recall score.

YOLO is a real-time detection algorithm that performs object detection as a regression problem. It has a unified architecture for feature extraction, classification, and bounding box prediction. While SSD uses multiple boxes or filters with different sizes and aspect ratios for detection, YOLO uses a fixed grid cell aspect ratio. YOLO introduced many novel techniques, including batch normalization, anchor boxes, fine-grained features, and multi-label classification for complex data sets with a large number of overlapping object labels. YOLOv4 offered a major performance leap with the CSPDarkNet-53 architecture and introduced several new concepts, such as weighted residual connections, Cross-Stage-Partial (CSP) connections, self-adversarial training, mosaic augmentation, and the Mish activation function [70]. A shallow feature enhancement in the YOLOv4 network was proposed by Wang et al. [71]. They achieved an 82.15% mean Average Precision (mAP) with YOLOv4, while the Faster RCNN offered a three times slower detection speed and an 81.29% mAP score. Yan et al. presented a comparison between the YOLOv5 algorithm variants for TSR using the BDD100K autonomous driving data set [72]. However, no classification of traffic signal phases was performed. The study concluded that the baseline performance of the detector algorithm can be improved by a great margin by applying corrections, removal of bias, and mosaic data augmentation in the input training image. The DL approaches for TSR are summarized in Table 5.

Table 5. Traffic signal recognition deep learning approaches

Ref.	Framework / architecture	Image resolution	mAP (%)	Detection speed
[73]	Detection: Residual CNN for semantic features + MSER; Recognition: ICFT tracking	1280 x 720 (BDD100K)	-	21.4 fps
[71]	Improved YOLOv4: Shallow feature enhancement + Bounding box uncertainty prediction	1280 x 960 (LISA) 640 x 480 (LaRA)	82.15 79.97	33.74 ms 40.17 ms
[72]	Recalculated anchors with K-means YOLOv5xl YOLOv5s	1280 x 720 (BDDTL)	AP: 1.61.6 2.63.3	1.142 fps 2.55 fps
[55]	Two layer SSD with MobileNetV2	1280 x 720	73.8	0.443 s
[74]	Detection: Heuristic (Gaussian Filter +Top-Hat + OTSU + HSI transform); CNN classifier: i.RTTLD, ii.m-RTTLD	LISA:1280 x 960 WPI:1920 x 1080	91.2 87.4 99.7	37 fps 26.8 fps 52.6 fps
[75]	(HSV color + HOG feature: edge & shape +SVM); Classification: AlexNet	640 X 480 (LaRA) 1280 X 960 (LISA)	79.6	13 ms
[76]	YOLOv3 + residual network	1920 x 1080 (VTLD)	-	0.106 s

Ref.	Framework / architecture	Image resolution	mAP (%)	Detection speed
[77]	ROI: Dual channel (HDR image); CaffeNet + YOLO-v2; Saliency map + Temporal trajectory tracking	1600 x 1200 *with ROI:	96.0 97.9	130 ms 35 ms
[69]	TL-SSD-Inception-v3	2048 x 512 (DTLD)	-	-
[70]	Coarse-grained Caffe-SSD, Reward-based Spatiotemporal ROI filtering	1280x720 (LISA),1280 x720(BSTLD)	F-score: 38.3 (avg)	0.33 fps
[78]	Hierarchical DeepTLR (HDTLR): AlexNet, ii. VGG, iii. GoogLeNet	1280 x 960 (HDTLR)	F1 score: 94.0, 88.8 ,85.8	12 fps
[79]	ROI: HSV color segmentation+ MSER(TL structure)+(HOG+ SVM) Recognition: CNN classifier	1280 x 800	F-score: 99.03 CNN:99.57	39 fps
[80]	YOLOv2	1280 x 960 (LISA-day)	AUC: 90.49	-
[81]	Detection- YOLOv1; Classification- CNN; Odometry based motion model tracking	1280 x 720 (BSTLD)	Accuracy: 99.0(val.) 95.1(test)	10 fps 0.06 ms
[82]	Deep TLR: Overfeat, Caffe-BVLC-AlexNet	1280 x 960 (DeepTLR)	-	13 ∼33 Hz

Hybrid Methods for Traffic Signal Recognition

To exploit the advantages of both traditional image processing and learning based methods, a few recent TSR studies adopted hybrid strategies that achieved 99.7% accuracy by combining a heuristic ROI detector with a lightweight CNN model, named Real-Time Traffic Lights Detector (RTTLD) [79] [74]. RTTLD achieved 10 frames per second (fps) on NVIDIA Jetson edge devices. Saini et al. used color segmentation and SVM-based traditional detectors with a CNN framework for efficient recognition of the traffic signals [79]. Wang et al. employed a method to use a CNN module with an Integrated Channel Feature Tracking (ICFT) module simultaneously [83]. In their work, the CNN is designed to obtain the positions of the traffic signals, which act as the initial information for the tracking module. The ICFT algorithm is then applied to track the targets continuously and determine the traffic light colors (i.e., phases).

Whether TSR detection and classification are performed using traditional model-based or machine learning methods, incorporating prior location information can greatly enhance accuracy. In this approach, a digital map of the ego vehicle's route is built, containing the position, orientation, and relevance of traffic signals. Real-time data from GPS, IMU, and LiDAR help refine ROI candidates by matching them with mapped signal locations. The authors in [84] proposed a digital mapping method using back-projection and triangulation,

leveraging GPS-IMU temporal data for probabilistic detection and HSV histogram filtering for classification. Semantic prior maps were employed by Fairfield et al. [85]. They mapped traffic signals via image association and linear triangulation, achieving 99% precision with LiDAR-based KD-tree 3-D localization under normal lighting. Li et al. further enhanced detection by introducing inter-frame correlation to refine proposal ROIs using previous frames [38].

Among other localization techniques, Possatti et al. proposed the use of a Simultaneous Localization and Mapping (SLAM)-based prior map with particle filter localization systems [86]. The localization data were corrected via matching 2-D local occupancy grid maps transformed from LiDAR 3-D point cloud data. Hirabayashi et al. combined a 3-D point cloud map from a 360-degree LiDAR with a 3-D feature map that contains separate IDs for traffic signals, poles, lanes, and light bulb angles [87]. They extracted ROIs from the combined maps with a normal distribution transform (NDT) for pose estimation. Li et al. introduced a multi-sensor data fusion assist (MSDA) algorithm that combines camera, LiDAR, real-time kinematic (RTK) GPS positioning, and high-precision IMU for real-time detection [88]. They analyzed the relationship between sensor error and ROI size and built an Adaptive Dynamic Adjustment (ADA) conversion model for optimum ROI generation. YOLOv4 was applied to extract and identify image features. For more accurate ROI generation, Jang et al. integrated road slope information from a digital map [89]. A Kalman filter-based tracking algorithm was applied to compensate for the perspective deformation by exploiting the ego vehicle’s distance to the traffic signals.

A few TSR studies utilized saliency maps with prior map information. A saliency map primarily highlights the region of an image that contains an object of interest with unique features. John et al. proposed a saliency map-based TSR model, where they used color, intensity, and shape information from the GPS-assisted ROI and fed them to the CNN classifier under normal illumination scenarios [90]. Even in low-light scenarios, they achieved 96.25% accuracy by leveraging recognition results from normal illumination and GPS-indexed information. The authors further extended their work with a modified Multi-dimensional Density-Based Spatial Clustering of Applications with Noise (M-DBSCAN) algorithm. For real-time detection, generated saliency maps were retrieved and used as they were previously, with a template matching method.

Traffic Signal Data Sets

Some of the prominent AV perception data sets in the literature are: KITTI [91], CityScape [92], ApolloScape [93] , nuScenes [94] , and BDD100K [95]. However, most of these data sets do not include separate classification labels for different light states (e.g., Green, Red,

Yellow, or Off) or for different orientations (e.g., horizontal, vertical, or occluded). Therefore, these data sets can only be used for traffic light localization, not for recognizing different states. To the best of our knowledge, only a few high-quality image data sets offer annotation classifications based on different traffic light states, orientations, or relevance to an ego vehicle. The most well-known public traffic signal data sets include the DriveU, Bosch, and LISA.

DriveU Traffic Light Data Set

The DriveU Traffic Light Dataset (DTLD) is one of the largest public traffic signal benchmarks, comprising driving sequences from 11 German cities. It includes object attributes such as orientation, pictogram, and tracking class IDs for traffic signals. Alongside camera images, it provides disparity data, calibration, and vehicle information (e.g., GPS, speed, yaw rate) to support depth-based detection and evaluation. DTLD-v2.0 contains 292,245 annotated traffic light images across 620 label classes, categorized by object attributes [96].

Bosch Small Traffic Lights Data Set

The Bosch Small Traffic Lights Dataset (BSTLD) contains 13,427 images with 24,242 annotated traffic signals [81]. It provides 12-bit HDR images reconstructed as 8-bit RGB, covering diverse urban scenarios with varying traffic density, lighting, and weather conditions. The training set includes 5,093 images with 10,756 annotated signals across 15 labels, annotated every 2 sec. The test set comprises 8,334 images with 13,486 annotations and four labels (e.g., Red, Yellow, Green, Off).

LISA Traffic Light Data Set

The LISA Traffic Light Dataset includes 43,007 frames and 113,888 annotated signals from driving sequences recorded in San Diego, California [97]. Captured using a stereo camera under varying lighting and weather conditions, it provides 13-day and five-night clips for training, and four-day and two-night clips for testing, with a 66° field of view. Two annotation types are available: one for the entire traffic light area, and another for the illuminated region only. Six signal classes are provided across both sets. Several studies also use locally collected datasets, referred to here as local datasets.

Objective

The primary objectives of this research were to:

- Assess the feasibility and accuracy of using computer vision technology for performance evaluation at signalized intersections;
- Provide intersection video footage data captured by drones;
- Use computer vision and artificial intelligence to automatically convert data from video recordings at selected intersections into trajectories of road users;
- Use computer vision and artificial intelligence to count road users and detect queuing and demand for each approach at selected intersections using drone footage; and
- Develop tools to facilitate DOTD traffic engineers in understanding road users' behavior, evaluating intersection performance measures, and assisting in determining effective measures for improving safety and efficiency at intersections.

Scope

This study focused on the development and evaluation of computer vision-based methods for intersection performance analysis, with emphasis on vehicle counting and queue detection integrated with traffic light recognition.

The scope of the project is defined as follows:

- Development of a detection-tracking-counting pipeline for the accurate measurement of vehicle turn movements.
- Implementation of a lane-based queue detection algorithm combined with traffic signal recognition to capture real-time demand and delay.
- Testing and validation of both methods on drone- and camera-based video datasets collected from selected intersections in Louisiana.
- Integration of the methods into a graphical user interface (GUI) to support practical application by engineers.

Methodology

This chapter presents the development of the artificial intelligence (AI) algorithms implemented in the project, which leverage computer vision techniques to process video data captured from traffic cameras and drones at selected intersections.

The objective was to accurately detect and track vehicles for both turn-movement counts and queue-length estimation. Additionally, the system detects the traffic signal states to provide signal state information alongside vehicle data. To achieve these tasks, a Detection-Tracking-Counting (DTC) computer vision pipeline was adopted, in which object counting is performed through the sequential processes of detection, tracking, and counting. Meanwhile, the corresponding traffic signal states are detected and lane-assigned using defined regions [98]. Further, the system incorporates pedestrian–vehicle interaction analysis to evaluate pedestrian behavior at intersections. To support the use of the system, a software tool with a graphical user interface (GUI) is provided to perform the analysis; see Appendix B.

Vehicle and Traffic Light Signal Detection Model Development

Model Development, Training, and Validation

Before detection model development, the required dataset is first generated from the video data collected from drones and traffic cameras; this process is detailed in the chapter “Data Collection and Dataset Generation,” which follows this chapter.

To develop the detection model, the YOLOv11 [99] architecture for vehicle detection is adopted, benefiting from its enhanced performance in both accuracy and speed over its predecessors. YOLOv11 features multiple variants, each tailored to specific trade-offs between accuracy and speed, based on the number of parameters, depth, and width scaling factors in the architecture. The architecture follows a backbone-neck-head structure. The backbone uses a cross-stage partial (CSP)-DarkNet network for improved gradient flow and efficient feature extraction, integrating a multi-card residual mechanism for better gradient propagation. The neck component utilizes a path aggregation network (PANet) combined with bi-directional feature pyramid network (BiFPN) layers, enhancing feature aggregation and scale invariance. The head employs anchor-free detection layers with decoupled classification and regression branches, predicting bounding boxes, class labels, and confidence scores simultaneously. Additionally, the use of the efficient layer aggregation network (ELAN) in the backbone improves feature reuse, further boosting inference speed and detection precision.

The model is trained using the dataset generated and enhanced with augmented data. Afterward, the model is tested on sample validation data. Figure 1 and Figure 2 show sample validation images for the traffic light states and vehicles, respectively. It was observed that the models can predict the class, as seen in the ground truth.

Figure 1. Sample test of the traffic signal detection model

Figure 2. Sample test of the vehicle detection model

Summary of Data Size, Labeling Man-Hours, and Computing Resources

This section provides the analysis and summary of the estimated data volume from the drone-captured videos, the man-hours in labeling, and the computing resources employed.

Data Volume

A total of five drone video sets were collected for analysis, totaling approximately 70 GB of data: an initial 20 GB dataset, followed by an additional 50 GB during subsequent collection and processing phases.

Dataset Generation

Preparation and labeling required over 180 man-hours of cumulative effort by the research team.

Model Training and Validation

Initial training ran for approximately nine hours, followed by multiple fine-tuning sessions to improve performance and generalization. The computing resources utilized for training and validation are listed in Table 6.

Table 6. Training, validation, and fine-tuning hardware

System	Purpose	CPU	GPU / Nodes
LSU High Performance Computing (HPC) *Cluster: SuperMike-3*	Primary training	(cluster nodes)	NVIDIA Tesla A100
Dell Precision 5820 Tower	Fine-tuning / validation	Intel Core i9-8700 @ 3.20 GHz	NVIDIA RTX A5000
Dell Alienware R15	Fine-tuning / validation	Intel Core i7	NVIDIA RTX 4090

Vehicle Tracking Algorithm

For tracking vehicles, we adopted the ByteTrack algorithm [100], a robust and efficient MOT technique. ByteTrack utilizes KF along with a constant velocity motion model to estimate the future positions of tracked objects based on their previous states. These predicted states are then matched with detection bounding boxes from the current frame using a combination of IoU and feature embedding distances. Similar to the SORT algorithm, ByteTrack uses the Hungarian algorithm to solve the assignment problem, optimally matching predicted object states to the detected bounding boxes. The major feature of the ByteTrack algorithm is a two-step data association process that incorporates both high-confidence and low-confidence detections, resulting in improved tracking accuracy compared to many other SOTA MOT algorithms.

A virtual-line crossing-based method is implemented for the proposed turn movement counting. In this method, a virtual line is placed at each intersection leg near the lane end. When a tracked object crosses multiple virtual lines associated with a specific movement direction, the turn counter is updated.

Movement of Interest (MOI) Assignment

In a typical four-way intersection, the legs can be manually assigned as South (S), North (N), East (E), and West (W), forming the set of zone IDs $Z = \{S, N, E, W\}$. Each zone accommodates four primary vehicle movements: through (straight), right turn, left turn, and U-turn. Vehicles entering an intersection are considered inbound traffic, while those exiting are considered outbound. We define the inbound zones as $Z_{in} = \{S_{in}, N_{in}, E_{in}, W_{in}\}$ and the outbound zones as $Z_{out} = \{S_{out}, N_{out}, E_{out}, W_{out}\}$. The set of MOI is the Cartesian product of these sets:

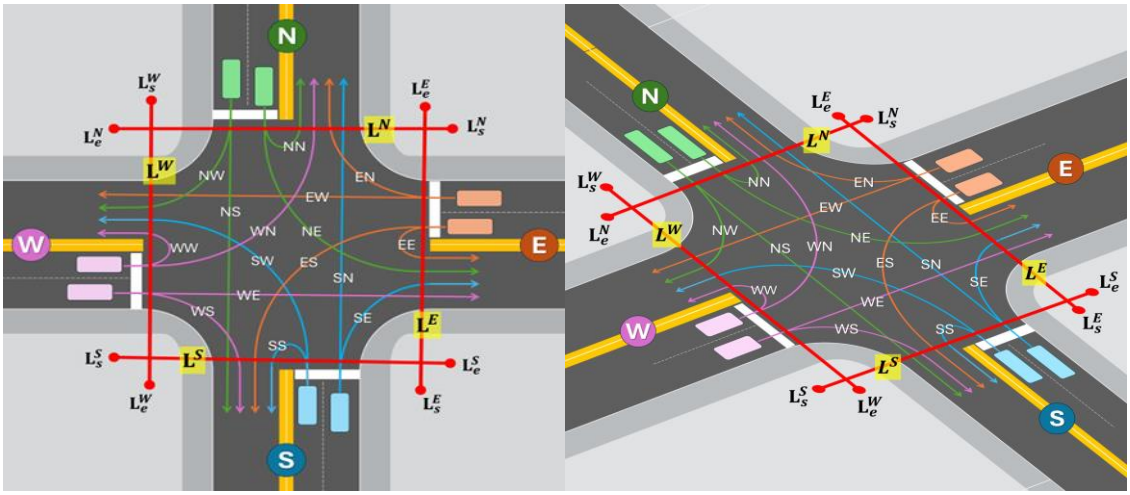
$$M = Z_{in} \times Z_{out} = \{(z_o, z_d) \mid z_o \in Z_{in}, z_d \in Z_{out}\}$$

where z_o represents the origin zone, and z_d denotes the destination zone, resulting in 16 distinct MOI-IDs, each representing a specific movement direction; see Figure 3.

To detect vehicle crossings, we use a virtual line L^Z at each zone $Z \in Z$. A complete movement count requires a vehicle's trajectory to intersect at least two zone lines: the first crossed line indicates the origin $z_o \in Z_{in}$, and the final crossed line indicates the destination $z_d \in Z_{out}$. Next, each movement is assigned an MOI-ID based on this sequence (z_o, z_d). For instance, a vehicle crossing inbound at L^S and outbound at L^W is categorized as a left turn from South under M (S_{in}, W_{out}) or is simply represented as M^{SW}.

Virtual lines L^Z should be ideally placed near stop lines, which mark the end of lanes and the beginning of the intersection area, as depicted in Figure 3. However, precise placement can be challenging if the camera's field of view does not clearly separate all vehicle routes, or if the recording camera is mounted too low or too far from the count ROI. In such cases, crossing lines must be strategically placed with minimal overlap to avoid triggering unintended counts from vehicle movement trajectories.

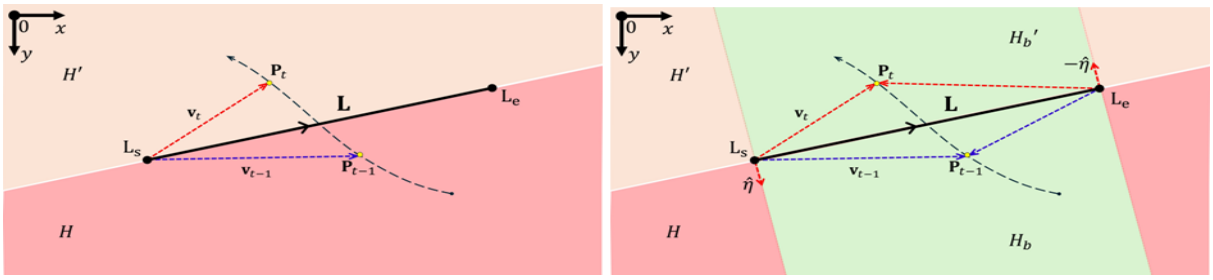
Figure 3. Illustration of a four-leg intersection, turn movements, and virtual crossing lines placement



Counting Method: Virtual Line Crossing Trigger

We utilize vector geometry principles to identify inbound and outbound vehicle crossings at each virtual line segment. Let a virtual line segment L be defined between L_s and L_e in the pixel coordinate system, as shown in Figure 4. L divides the image space into H and H' regions. To determine in which region any tracked point P is at, we compute the 2-D cross product of L and the relative position vector of P. The sign of the cross product indicates which side of the line P lies on.

Figure 4. Illustration of virtual line crossing method



To identify whether P has moved between H and H', we track the change of value of the cross product. For example, for two target points P_{t-1} and P_t in two consecutive video frames, a change in sign of the cross product indicates that the point has moved from one side to the other; see Figure 4.

Inbound-Outbound Configuration

We categorize detected crossing IDs into inbound set T_{in}^{Z} and outbound set T_{out}^{Z} based on their crossing direction. To ensure consistency between the virtual crossings described in Figure 4 and actual vehicle movements in a video frame, the virtual line L for each zone must

be configured uniformly. In our experimental setup, L_s is always placed near the outbound side, and L_e is near the inbound side of the route at each intersection leg. In other words, when viewing the intersection from any leg, L_s is always positioned on the left and L_e on the right.

Turn Count Method

The turn count method is initialized by identifying vehicles that intersect individual crossing lines, categorizing them into sets T_{in}^{Z} and T_{out}^{Z}. Due to factors such as low camera-view angles, route geometry, or closely spaced virtual crossing lines, a vehicle's bounding box may partially intersect multiple lines, while other parts may not intersect the intended crossing lines at all. To improve the robustness of line crossing detection, we enable tracking of multiple keypoints on each bounding box to trigger the crossing events. These keypoints include the four corners, midpoints of the edges, and the centroid of each bounding box. To handle unforeseen cases where a single optimal keypoint cannot be selected and intersecting multiple lines is unavoidable, we utilize the earliest inbound crossing and the latest outbound crossing data from the vehicle's tracking history to determine valid turn movements.

We use a data matrix to store the zone ID-frame index pairs, (Z_{in}, t_{in}) and (Z_{out}, t_{out}), respectively, for individual tracking IDs. From the stored data matrix, we find the smallest frame index for each tracking ID that determines the first valid inbound crossing or the origin zone z_o. Similarly, the largest frame index provides the final valid outbound crossing or the destination zone z_d. The (z_o,z_d) values against each tracking ID are then assigned to the corresponding MOI-ID set M (z_o, z_d). For U-turn cases, where $z_o = z_d$, it is only considered valid if the frame difference between crossings is greater than or equal to a threshold value f_b, which is set to 5*fps frames in the test videos. This step ensures that U-turn crossings are not triggered due to any shift in the bounding box locations when vehicles are stationary on top of the crossing lines. Finally, the turn counts C (z_o, z_d) are updated by incrementing the count for each identified MOI.

Vehicle Queue Length Estimation (VQLE)

The detection model described serves as the base module. Its outputs per frame detections and associated tracks provide the input data used to develop the full queue-length estimation algorithm.

The vehicle queue length estimation provides the count of vehicles in each lane for a given section of the intersection. The algorithm comprised different parts. The vehicle queue length estimation is implemented as a count-based approach for multiple lanes. This approach is

motivated by traffic engineers' need to know the number of vehicles in a queue. Although additional information, such as the physical queue length in meters or yards, can also be estimated, this aspect is not the primary focus of the present work. The VQLE algorithm comprises lane region definition, detection and count, and persistent enhancement.

Lane Region Definition

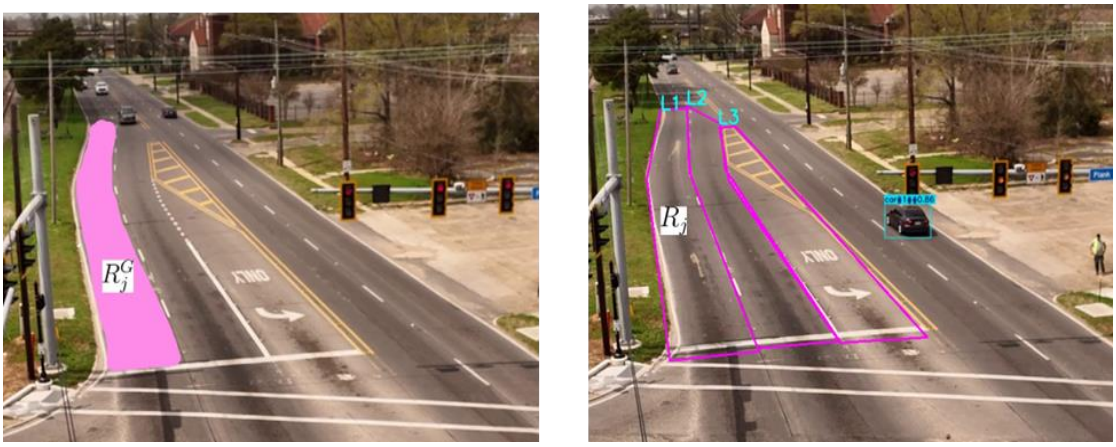
While a vehicle may be detected within a portion of the image frame, not all vehicles contribute to the queue for a given lane. To associate vehicles with a lane, the lane region definition specifies a per-lane polygonal region in the image plane. Each lane region is represented as a polygon Rj, defined by either four or six coordinate points, depending on the geometry of the lane; see Figure 5.

- For straight lanes, Rj is defined by four corner points.
- For lanes with slight curvature, six points are used to capture the shape more accurately. Formally, a lane region is defined as:

$$(R_j = \text{Polygon}\{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\})$$

where (x_i, y_i) are the pixel coordinates in the image plane, and $m \in \{4, 6\}$ represents the number of polygon vertices. Each polygon Rj defines a boundary in the image plane that corresponds to a section of the physical lane in the real world. The physical section, defined as the ground truth lane boundary, is denoted as R^G in Figure 5 and serves as the reference for accurate alignment between the image-based and real-world lane regions, such that $(R_j \approx R_j^G)$.

Figure 5. Comparison between ground truth and defined lane regions for vehicle queue estimation



(a) Ground truth: physical lane section

(b) Region in the image for queue estimation

Vehicle Queue Estimation

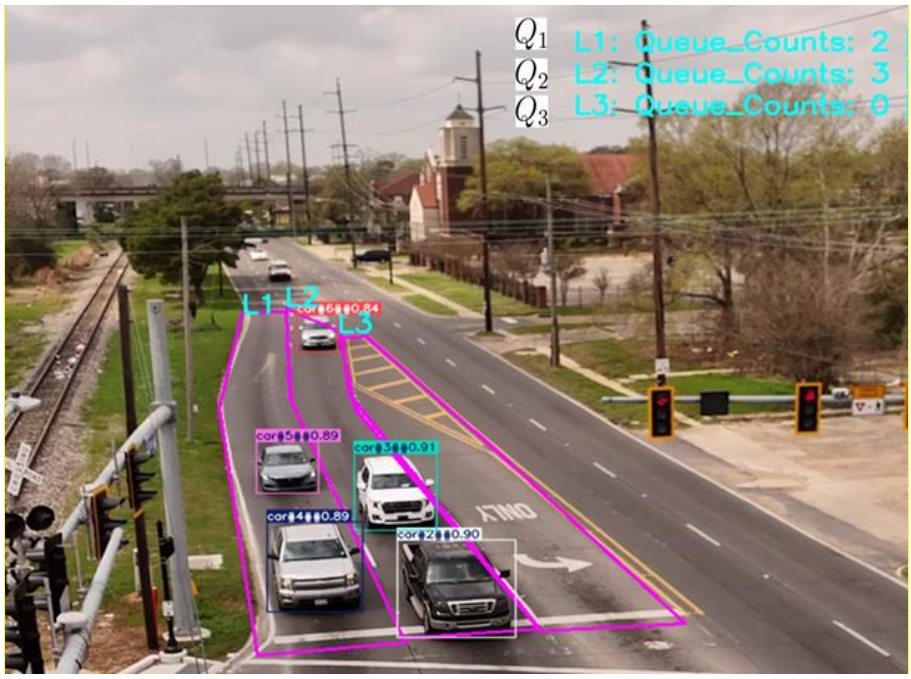
Vehicles detected in each frame are represented by bounding boxes, which provide the pixel-level dimensions of the detected objects. From these bounding boxes, the centroid coordinates (x_c, y_c) are computed to represent the approximate position of each vehicle. The queue length is then obtained through frame-wise summation of vehicles in the region.

Let N denote the total number of detected vehicles in a given frame, and let Rj represent the polygonal region corresponding to lane j. The queue length $Q_j\,(k)$ at frame k is then determined as $Q_j(k) = \sum_{i=1}^{N} 1$ for each centroid $\left(x_c^i, y_c^i\right) \in R_j$. The vehicle count for each can be seen in Figure 6.

Persistent Vehicle Count

Occasional fluctuations or misdetections may occur, affecting the stability of the queue-length estimate. To improve consistency over the duration of the video analysis, a persistence algorithm is applied. If the count for a lane remains unchanged over a window of Nv consecutive frames, the vehicle estimate is updated; otherwise, short-term fluctuations caused by temporary misdetections are ignored**.**

Figure 6. Vehicle queue lengths for a three-lane section showing $Q1 = 2$, $Q2 = 3$, and $Q3 = 0$ for Lanes 1, 2, and 3, respectively



Adaptive Lane Boundary Algorithm

Drifting Problem

A common issue in drone-captured videos is gradual camera drift between lane boundaries defined in the video frame and the ground truth position of the lanes [101]. Although the lanes remain visible, their initial pixel-wise locations slowly shift, reducing the accuracy of the queue counting; see Figure 7.

Correction

To address this problem, the Optical–Flow–Based Adaptive Multi-Lane Boundary Algorithm (OF-ALBA) is developed to stabilize and rectify the drift. The pyramidal Lucas–Kanade optical flow [51] is used to estimate the drift and update lane boundaries adaptively. The pyramidal Lucas–Kanade algorithm tracks feature-point motion in local patches under brightness-constancy and small-motion assumptions.

Correction Details: Adaptive Lane Boundary Algorithm (ALBA)

The correction procedure is implemented as follows:

1. **Initialization**: Define a reference region of interest in the first frame and detect feature points within a defined region using a corner detection method such as the Shi–Tomasi algorithm [102].
2. **Feature Tracking:** For each frame k, the feature points are tracked using pyramidal Lucas–Kanade optical flow, retaining only valid correspondence points.
3. **Displacement Estimation**: The displacement experienced by each feature point is computed, and the average displacement across all valid points is determined. At scheduled update frame intervals, the margin of lane drift is updated.
4. **Adaptive Boundary Update:** Every lane region is translated and updated to match the ground truth.
5. **Counting**: Vehicle counting is then performed within the updated region, ensuring alignment with the actual lane boundaries despite any drift from the initial ground truth. The algorithm continuously compensates for drift.

Figure 7 shows the mechanism of the algorithm on an actual video sample without ALBA. The actual traffic lane gradually drifts over time towards the left and out of alignment with the defined lane boundaries, which are represented in magenta. Initially, at the start of the video, the defined lane boundaries and the ground truth lane positions are well aligned. However, as the video progresses, a noticeable drift occurs between the static, frame-attached boundaries and the true physical lane positions. This misalignment arises because the defined

lane boundaries are fixed relative to the video frame rather than to real-world coordinates. Consequently, camera or drone motion, such as small yaw, pitch, or translation shifts, causes deviation. Over longer periods, this drift leads to cumulative misalignment, where vehicles from one lane may be incorrectly assigned to adjacent lanes. For instance, vehicles in Lane 1 may be mistakenly counted under Lane 2, and similarly, Lane 2 vehicles may drift into Lane 3, resulting in instability and inaccuracies in queue estimation. Such drift effects underscore the importance of implementing an adaptive lane-boundary correction mechanism that dynamically adjusts to maintain alignment between the visual lane regions and their true real-world counterparts throughout the video. In Figure 8, the adaptive lane boundary algorithm is applied. It can be observed that under similar traffic conditions, the defined lane boundaries now align closely with the physical lanes, effectively correcting the drift previously seen in Figure 7. Further, vehicles are clearly positioned within their respective lanes, demonstrating the improved spatial consistency achieved through the algorithm.

Figure 7. Analysis with adaptive lane boundary algorithm not active

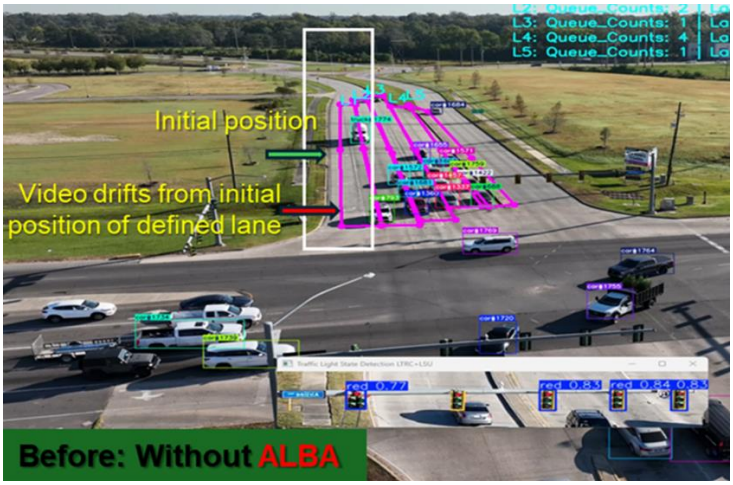
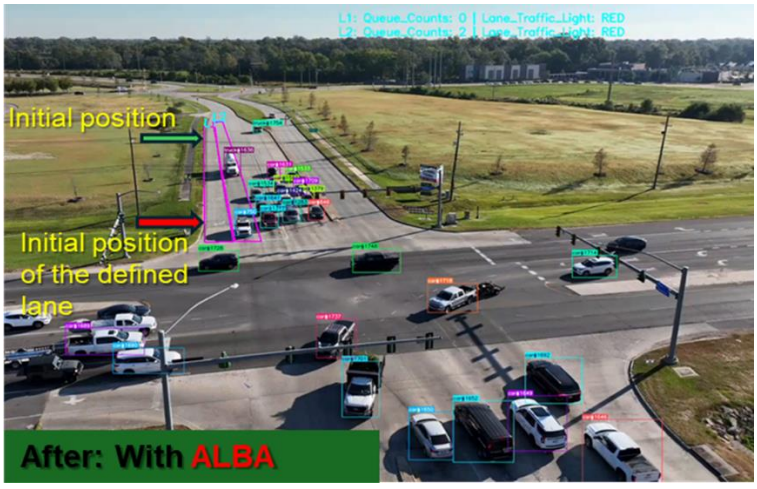


Figure 8. Analysis with adaptive lane boundary algorithm active



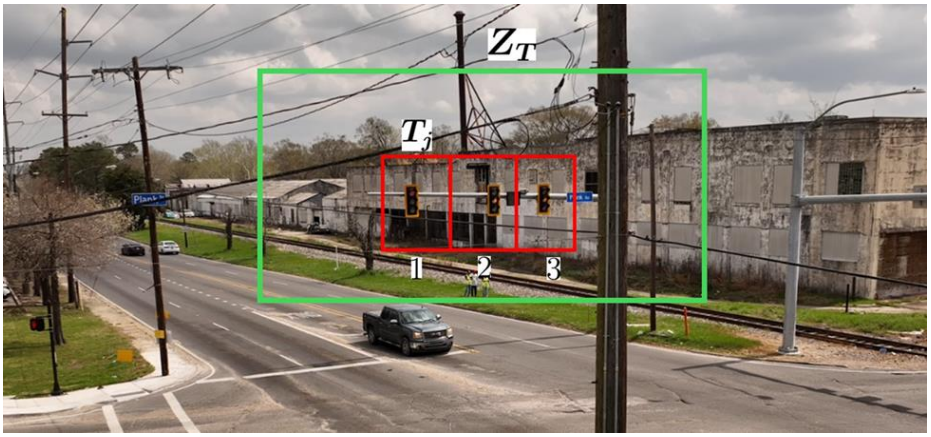
Traffic Light Signal Recognition (TLSR) Algorithm

The traffic signal recognition algorithm detects the signal states associated with each lane. The components of the algorithm are described as follows:

Traffic Light Region Definition

The complete image frame captures multiple things, including the traffic light. This can lead to multiple features not necessary for learning and inference potentially interfering with detecting the exact signal and establishing a robust traffic detection model and algorithm. To address this, an outer-and-inner region definition strategy is designed. First, a traffic light region of interest Z_T is defined sufficiently large to cover the zone of all the traffic lights corresponding to the lanes. Within this zone, the individual traffic light regions per lane are specified as, where $j = 1, \ldots, m$, and m is the total number of traffic light modules. An illustration of this is shown below in Figure 9.

Figure 9. Illustration of the proposed Traffic Light Signal Assignment strategy



Signal Detection and Persistence Filtering

The traffic signal zone of interest, denoted as Z_T, defines the area used for detecting traffic-light signals. Within this zone, the bounding boxes of detected traffic lights are analyzed, and their centroids are computed. Each centroid is then checked to determine whether it lies within a defined lane-specific region T.

Formally, if the centroid (,) lies within the region T_j, then the corresponding color label at that location is assigned as the signal state:

$$\text{colorLabel}\left((x_c^l, y_c^l)\right) = S_j(k)$$

where $S_j\,(k)$ represents the detected traffic signal state for lane j at frame k. Each detection is subsequently assigned to its corresponding lane. However, intermittent misclassifications may occur in individual frames, typically lasting only a few frames, which can introduce

noise into the TLSR output. To mitigate this, a temporal persistence filter is applied to enforce frame-to-frame consistency. A minimum frame-sequence length N_T is defined, such that a traffic signal state is registered as valid only if it persists consecutively for at least N_T frames.

In other words, the registered state $\bar{S}(k)$ is updated only when the same detection is maintained across N_T consecutive frames, effectively suppressing short-term fluctuations and spurious misclassifications.

Time-Based Filtering for Flashing Signals

The flashing phase primarily corresponds to the Flashing-Yellow (FY) state. This phase is inferred from the alternating sequence of two actual signal states: Yellow (Y) and Off (OF). To detect this behavior, the algorithm records the start and end times of consecutive Yellow and Off states, forming a repeating pattern. The Off state serves as the trigger point, while the final Off state marks the termination of the flashing sequence. For example, a typical flashing sequence can be represented as:

$$\text{FY: OF} - \text{Y} - \text{OF} - \text{Y} - \text{OF} - \text{Y} - \text{OF} - \ldots$$

In some cases, a Yellow phase may persist longer than usual, making it harder to distinguish a Flashing-Yellow sequence from a steady Yellow signal. To address this, a time-based filtering criterion is introduced. Let τY represent the duration of a continuous Yellow segment, and τd denote the threshold duration. If the duration $(\Delta\tau_Y)$ is shorter than the threshold τ_d, the segment is classified as part of a Flashing-Yellow sequence; otherwise, it is considered a steady Yellow phase.

Formally, this is defined as: $(Y_r(\tau) = \text{FY} \text{ if } \tau_Y < \tau_d, \text{ and Y otherwise})$, where FY denotes Flashing-Yellow, and Y denotes steady Yellow.

Approximate-Synchronization and Lane-to-Traffic Signal Order Alignment in Dual-Drone Video Capture Framework

The dual-drone data collection setup captures the same intersection scene from two opposite orientations, one oriented toward the vehicle queue region and the other toward the traffic light region. This configuration introduces two primary challenges.

First, despite the drone operation team's efforts to initiate both recordings nearly simultaneously, a small temporal offset typically exists between the two video streams, often within a few seconds. This can be verified from the metadata file associated with the video, which is the SRT file.

Second, because the cameras face opposite directions, the lane ordering in one video is not the same in the second video. For instance, left-to-right in the queue view does not directly correspond to that in the traffic-light view, which appears reversed.

Approximate Synchronization with Metadata (ASM)

We explore ways to synchronize the two videos, which are deemed sufficient for analysis. An approximate synchronization approach is proposed to align the two video streams based on their embedded timestamps. This synchronization does not aim for exact frame-level matching, which would require both drones to capture identical visual content, but rather seeks time-based coherence sufficient for comparative lane-to-traffic signal analysis.

Assume that the embedded timestamps in the associated SRT metadata are accurate, but the two drones were not started at the same time. Let the start times of the two recordings be denoted by τ_A and τ_B. The maximum or latter of these two start times, defined as:

$$\tau_e = \max(\tau_A^s, \tau_B^s)$$

is used to establish a common temporal reference. Sometimes, to ensure a clean and standardized reference point, this later start time can first be rounded up to the nearest whole second and then shifted forward by a small bias τ_b, to prevent sub-second jitter. The resulting reference time to trim both videos is now:

$$\tau_r = \tau_e + \tau_b$$

Using Video File Name for Time Reference

In some cases, the associated SRT files may be unavailable. When this occurs, the video file name generated by the drone can be used as an alternative source of time reference. The file names stored directly on the drone's memory typically include the recording date and time as part of the file name structure, even though they do not contain additional details such as GPS coordinates or frame-level timestamps available in the SRT files. For example, a video file named: VVV 20250314130758 0001.MP4 encodes the following information:

Date: March 14, 2025
Time: 13:07:58 (hh:mm: ss)

This method provides only a coarse time reference; it is still sufficient for identifying approximate recording periods when SRT metadata is missing.

Scene Inspection

When the drones have unsynchronized internal clocks, the recorded timestamps may differ by several seconds or even up to a minute. In such cases, relying on the SRT metadata or the file name alone can be challenging for achieving approximate time-based synchronization.

In this case, visual inspection of the videos can be used. Both recordings can be reviewed side by side to ensure they begin with approximately the same visual context. It is important to note that even when the start times differ, the captured scenes may still represent the same real-world event within a few frames of each other. For example, one video, such as the traffic-light view, may capture the signal turning Green, while the other, such as the queue-view video, records vehicles beginning to discharge. Despite timestamp differences, both are temporally aligned with the same event.

To improve confidence in alignment, the videos should be compared at moments where moving objects, such as vehicles or pedestrians, are visible in both views. By identifying at least three corresponding instances where the same event appears across both videos, a sufficient level of temporal alignment can be established. A common scene visible in both recordings can then serve as a reference point for trimming and aligning the two videos.

In practice, even when two videos differ by a few seconds in their start times, visual inspection often reveals nearly identical scenes. This level of correspondence is typically adequate for synchronized analysis with acceptable accuracy.

Experience from this project indicates that the best practice is to synchronize the internal clocks of the drones before flight and implement an on-field loud sound signal (e.g., clap or beep) during recording. This sound, captured by both drones, can later serve as a clear reference point for synchronization, as discussed further in the Recommendations section.

Lane-to-Traffic Signal Order Alignment

To maintain consistent visual interpretation and minimize post-processing overhead, the traffic-light video is horizontally flipped as a pre-processing step, thereby aligning the apparent left-to-right lane order with that of the queue-oriented video. This ensures a one-to-one correspondence between lanes and traffic lights, eliminating the need for additional coordinate transformations or inverse mapping operations in the algorithm itself (i.e., from first-to-*N*th lanes in the queue view to *N*th- to-first traffic-light positions). It is noted that order alignment is also useful where visualization is intended. The key benefit of visualization is that it allows the user to verify that the right videos are being processed and provides a direct match between the video and the algorithm result. The following figures show samples of the pre-processing done. The video can be flipped using available video processing software or a flip algorithm. Figure 10 shows the original image from the video, and Figure 11 shows the flipped version.

Figure 10. Lane-to-traffic signal without reordering

Figure 11. Lane-to-traffic signal with reordering

Data Collection and Dataset Generation

The primary data source for the project was aerial video and fixed traffic camera footage captured at various intersections in Baton Rouge, Louisiana. The videos represented different lane layouts and common types of traffic signal configurations in the city. Intersections of interest were identified by DOTD Traffic Engineers. Aerial footage was captured by drones, while fixed traffic camera footage was provided by the DOTD Traffic Engineering team. Drone flights were conducted by the staff from the DOTD Location and Survey Section, in collaboration with the LSU and LTRC research teams, with multiple field visits and data acquisition. First, an initial set of videos was collected by the drone team. This first batch served as a baseline for quality review and helped identify practical challenges and potential alternatives.

Based on the review, a joint effort among the DOTD Location and Survey drone team, the LSU research team, and the LTRC research team refined vantage points and flight altitudes to ensure clear visibility of vehicle queues and full traffic signal cycles. While the initial campaign used a single drone to capture turn movements, queue length, and signal states in one shot, the research team subsequently proposed a dual-drone video capture method to overcome the limitations of a single shot for all traffic parameters of interest.

DOTD Ground Traffic Cam Dataset

The traffic cam dataset was obtained from various intersections. Samples of these are shown in Figure 12.

Figure 12. Samples of traffic camera data set

Dual Drone Video Capture Method

The initial evaluation revealed that using a single drone to simultaneously capture the multi-lane intersection and the traffic signal state information in a single video, with sufficient clarity and quality for the AI model development, was challenging. These challenges included:

1. At certain intersections, obstructions such as poles and trees prevent capturing both scenes adequately in a single shot.
2. The traffic light appears small due to the poor proximity of the drone to the traffic lights, which is limited by spatial constraints to capture the central intersection along it. Moreover, regulation constraints prevent moving too close to the lights and flying over the intersection.
3. Capturing the full queue length across multiple lanes at the same time is challenging, making some vehicles appear smaller than usual.

To address these limitations and improve the captured videos, a dual-camera setup was adopted. One camera was positioned at a sufficient height to capture the queue length in the

primary lane while keeping the central intersection visible in the frame. This configuration enabled the camera to focus on a single lane with enhanced detail while maintaining the overall intersection context. The second camera was positioned to prioritize capturing the traffic light sections of the lane. The two drone cameras were oriented in opposite directions during video capture. The drone platforms used are the Mavic 3 and Matric300, as shown in Figure 13. A portion of the data collection team is shown in Figure 14.

Figure 13. Sample images of drone platforms used: (a) Matric300; (b) Mavic 3

Figure 14. Field operations during data collection: (a) team on site; (b) drone unpacking

Data Collection: Traffic Light Signals

The associated traffic lights for different lanes were captured by the DOTD Location and Survey drone team. Two people operated two drones in near-synchronous coordination. The following figures show sample footage of the traffic lights. These intersections included:

- Plank Road at 72nd Avenue; Plank Road at Choctaw Drive
- Airline Highway at Stumberg Lane; Plank Road at Evangeline Street

Sample images of the various intersection and traffic light types are shown in Figure 15 to Figure 19.

Figure 15. Traffic light at Airline Highway at Stumberg Lane; from left: R/Y/G and Y/G-Protected

Figure 16. Traffic light at Plank Road at 72nd Avenue; from left: Y/G-Protected and R/Y/G

Figure 17. Traffic light at Plank Road at 72nd Avenue South; from left: FY and R/Y/G

Figure 18. Traffic light at Plank Road at Choctaw Drive: All R/Y/G

Figure 19. Traffic light at Plank Road at Evangeline Street; from left: R/Y/G and Y/G-Protected

Data Collection: Vehicle Turning Movements and Queue Length

The following figures show sample footage of vehicle lanes, queues, and the central intersection for turning movements collected from the intersection, including:

- Plank Road at 72nd Avenue
- Plank Road at Choctaw Drive
- Airline Highway at Stumberg Lane
- Plank Road at Evangeline Street

Sample images of the various intersections and the nature of vehicles and queues are shown in Figure 20 to Figure 23.

Figure 20. Vehicle data from Airline Highway South

Figure 21. Vehicle data from Plank Road at 72nd Avenue

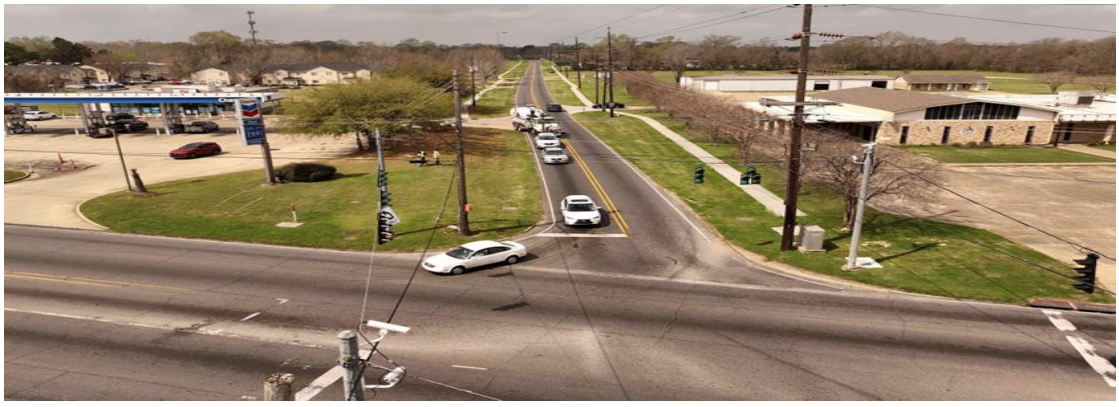


Figure 22. Vehicle data from Plank Road at Choctaw Drive

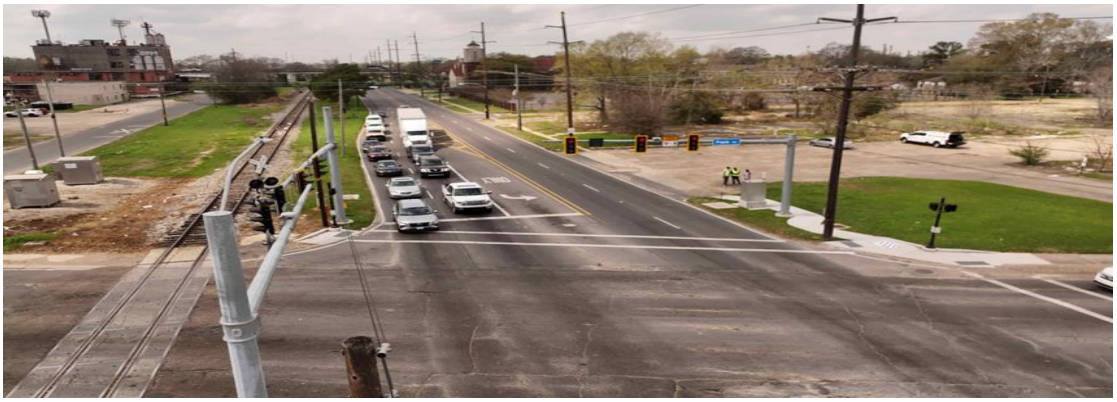


Figure 23. Vehicle data from Plank Road at 72nd Avenue

Dataset Generation

The dataset here refers to a structured collection of images and defined class labels (i.e., annotations) for training computer vision models, which are subsequently used to develop AI-based traffic analysis algorithms. The dataset was generated from the video data collected during field operations. A series of pre-processing stages was included in this process.

Data Extraction and Cropping

To generate the dataset, frames were extracted from selected videos and sampled sequences, with a portion reserved for verifying and testing the developed algorithms.

In this process, rather than extracting every frame sequentially, frames were extracted at defined intervals (e.g., every 25th, 30th, or 70th frame). This approach introduced variability into the dataset and minimized redundancy caused by consecutive, visually similar frames. An automated extraction procedure was implemented to process the video stream frame-by-frame, selecting and saving only those frames that satisfy the predefined sampling condition.

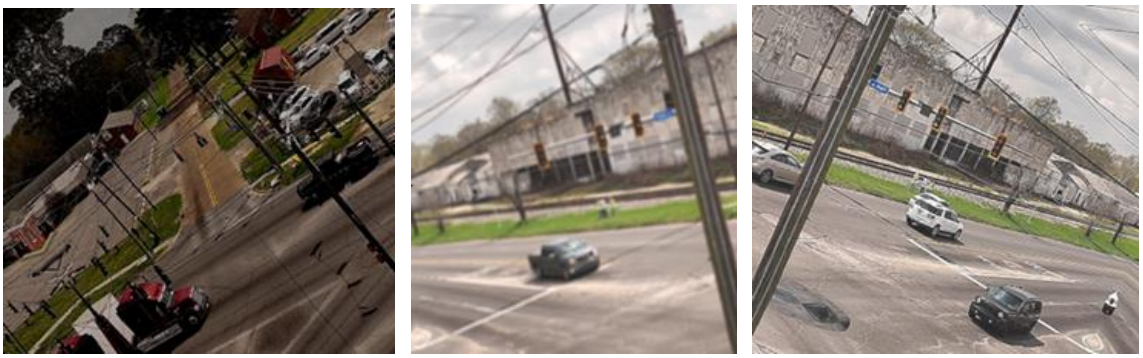
This strategy also ensures that the resulting dataset captures diverse visual content across different time segments of the video while avoiding unnecessary repetition and storage overhead.

The next key step involved cropping the large frames into manageable image sizes, constrained to a minimum resolution (e.g., 640×640 pixels) and focusing on sections containing the object of interest for feature extraction. This step was particularly applied to the traffic signal dataset generation. This is relevant because of the regulatory constraints on drones, which prevent them from being flown in close proximity to provide an ideal close-up view of the lights. Moreover, the presence of electrical poles, trees, and other structural elements at certain intersections obstructs the preferred viewpoints. Even with the video captured, unwanted areas such as roadways and vehicles in lanes can appear alongside the traffic video and interfere with the deep learning model's feature extraction.

Data Augmentation

To enhance dataset diversity and improve model robustness, various augmentation techniques were applied to the extracted and cropped images. Data augmentation effectively increases the dataset size by generating transformed variants of existing samples, thereby improving generalization during model training. Transformations such as rotation, horizontal flipping, brightness adjustment, and contrast variation were applied to introduce appearance diversity and simulate different environmental conditions. Figure 24 presents examples of the augmented images generated from the original dataset. In addition to these standalone augmentations performed during dataset preparation, an internal augmentation pipeline was also applied during model training using the Albumentations library. This combination improves both dataset richness and learning stability.

Figure 24. Examples of augmented performance on the data:
(a) rotation and contrast; (b) rotation and blurring; (c) rotation and sharpening



Data Labeling

A detailed analysis was completed to define the object classes, namely, traffic-light states and vehicle types, to capture the variations observed in the collected video data. Taking a cue from the Federal Highway Administration (FHWA) guidelines on automatic vehicle classification systems [103], it is recommended that visually similar objects be grouped into broader, representative categories to improve recognition accuracy in AI-based models. Accordingly, vehicle types corresponding to FHWA Classes 2 and 3 were merged into a single category labeled as “Car,” representing standard passenger vehicles.

Traffic Light States

Table 7. Defined traffic signal states

ID	Label	Description
0	R	Red
1	Y	Yellow (Amber)
2	G	Green
3	YP	Yellow-Protected
4	GP	Green-Protected
5	OF	Off-State
–	FY*	Flashing-Yellow

*Flashing-Yellow is indirectly classified in the algorithm based on temporal sequencing. Thus, while the direct classification includes six primary labels (IDs 0–5), one additional state (FY) is inferred.

Table 8. Vehicle classification

ID	Label
0	Car
1	Truck
2	Bus

After the pre-processing stage, the data were labeled according to the defined classification scheme. The labeling process was performed manually, requiring substantial man-hours to

annotate, review, and correct mislabeling. Several existing auto-labeling algorithms were explored; however, experience showed that they often led to various forms of misclassification because the pretrained models did not align with the specific characteristics and requirements of the research project dataset. Consequently, many samples required the relabeling and readjustment of bounding boxes to ensure accuracy and consistency with the project specifications.

Figure 25 and Figure 26 show the vehicle labeling process, showing examples of the defined classes and annotation procedures. For each image containing one or more objects, the annotator manually draws a bounding box around the object and assigns the corresponding class label. The annotation data are stored in text files that include information such as the class ID, bounding box coordinates, and dimensions for each annotated object.

Figure 25. Manual labeling of vehicle samples 1 and 2

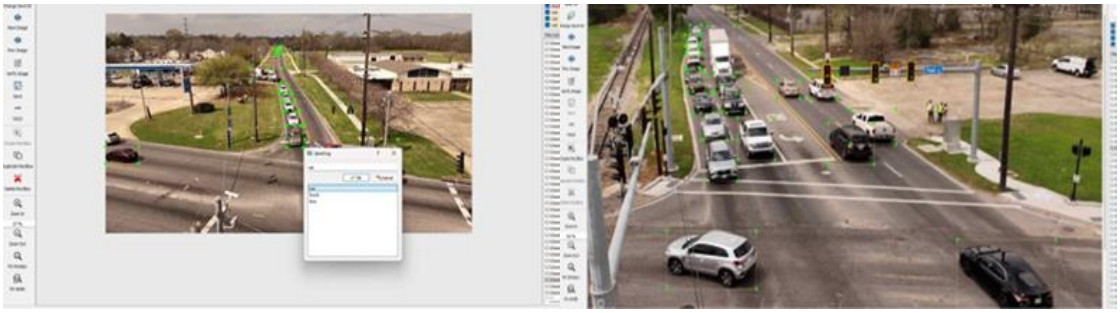


Figure 26. Manual labeling of vehicle samples 3 and 4

Similarly, the labeling of traffic lights was performed as shown in Figure 27 and Figure 28; these are examples of the defined classes and the corresponding labeling process.

Figure 27. Manual labeling of traffic light signal samples 1 and 2

Figure 28. Manual labeling of traffic light signal samples 3 and 4

Open-Source Traffic Intersection Datasets

Effective TMC methods require optimal camera placement to capture all vehicle entry and exit points, as poor placement can lead to significant occlusions and degrade accuracy.

AICity2021 Dataset: This dataset provides 31 unannotated traffic video clips, totaling approximately five hours of footage captured from 20 unique camera views in Iowa. The dataset comprises high-resolution video data sourced from two distinct camera types: long-range Pan-Tilt-Zoom (PTZ) cameras with a resolution of 1280x720, and fisheye cameras with a resolution of 1280x960, recorded between 10 and 15 fps. The dataset presents a variety of traffic environments, including signalized intersections, highways, and ramps, captured under diverse conditions such as daylight, dawn, rain, and snow.

Bellevue Traffic Video Dataset: The Bellevue traffic dataset consists of roughly 101 hours of video footage collected from five traffic intersections within Bellevue, Washington. The videos are captured at 30 fps with a 1280x720 resolution from fisheye traffic cameras mounted on roadside poles. For each intersection, the dataset provides videos recorded 24 hours a day, enabling analysis across different traffic conditions and lighting scenarios. Samples of the open-source traffic data are shown in Figure 29.

Figure 29. Samples from open-source traffic intersection datasets

A list of the datasets used for the counting performance evaluation is included in Table 9.

Table 9. Intersection datasets used for TMC evaluation

Dataset	Scene	Resolution	Duration	Frames
AICity2021 Counting	Cam 6	1280×960	30 min	˜18k
	Cam 9	1920×1080	5 min	˜3k
Bellevue 2018	116th–NE12th–20170911120833	1280×720	˜1 hr	˜108k
	150th–Newport–20170911080831	1280×720	˜1 hr	˜108k
LADOTD-Drone 2025	Airline Cue	1920×1080	3 min	—
	Airline West-001	1920×1080	4:11 min	—
	Airline West-002	1920×1080	4:38 min	—
	Plank 72nd West	1920×1080	3 min	—
	Plank Evangeline	1920×1080	30 sec	—
	Plank Choctaw	3840×2160	30 sec	—

Pedestrian and Vehicle Interaction

The interaction between vehicles and pedestrians at intersections plays a critical role in optimizing the overall performance and safety of urban traffic systems. Intersections with heavy vehicular traffic are generally more prone to pedestrian crossings and potential conflicts than those with lower traffic volumes. The complexity of these interactions depends not only on traffic density but also on the behavioral dynamics of both vehicles and pedestrians during the queue formation and discharge phases.

The goal is to analyze the trajectories of both pedestrians and vehicles over time to determine these interactions and pedestrian behavior. Two components of particular interest are:

- **Pedestrian Crossing Behavior:** Pedestrians, particularly vulnerable groups such as the elderly or individuals with mobility impairments, require more time to cross intersections

safely. The duration of pedestrian crossing events significantly affects vehicle waiting times and the efficiency of signal phases.

- **Vehicle Compliance and Reaction Behavior:** Vehicles are expected to comply with stop conditions when pedestrians are present within designated crossings. However, the degree of compliance and the drivers' reaction time can vary depending on traffic conditions, visibility, and behavioral tendencies.

Data Collection: Pedestrian Videos

In this project, traffic surveillance videos installed at various intersections were used to capture possible pedestrian movements. These cameras are typically mounted on traffic light pole structures positioned near pedestrian crosswalks to capture crossing events with adequate visibility and coverage.

Camera Installation

Dedicated cameras were installed at six locations to capture pedestrian activity levels, particularly near crosswalks. Each camera was mounted on a supporting pole secured to existing traffic light pylons to ensure stability and optimal viewing angles. The installation process was carried out through a collaborative effort between the LTRC engineers and LSU research team.

Figure 30 shows the six locations where cameras were installed. These include points along Nicholson Drive, Nicholson Stadium, near Gulf Hall, Highland Road at LSU, Highland Road at Chimes, and the Highland West/East intersection.

Figure 30. Pedestrian camera installed at different intersections

a) Nicholson Stadium Intersection

b) Gulf Hall

c) Highland Road at LSU Intersection

d) Highland Road at LSU Intersection


e) Highland Road at Chimes


f) Highland Road at W/E State Street

Discussion of Results

Performance Evaluation: Turn Movement Count

In our turn movement count (TMC) evaluation method, True Positives (TP) are defined as instances of correctly counted turning vehicles. False Positives (FP) refer to non-vehicle objects that are incorrectly counted as vehicles, while False Negatives (FN) indicate vehicles that are not detected or counted. True Negatives (TN) are excluded from the analysis, as they do not correspond to any events of interest within the context of this study. Using the TP, FP, and FN counts from the test videos, we calculated the micro-averaged F1 score.

$$\mathrm{F1}_{\mathrm{micro}} = \frac{2 \times \sum_{j=1}^{M} \mathrm{TP}_j}{2 \times \sum_{j=1}^{M} \mathrm{TP}_j + \sum_{j=1}^{M} \mathrm{FP}_j + \sum_{j=1}^{M} \mathrm{FN}_j}$$

where M represents the total number of MOIs. TP_j, FP_j, and FN_j are the True Positives, False Positives, and False Negatives, respectively, for each MOI j.

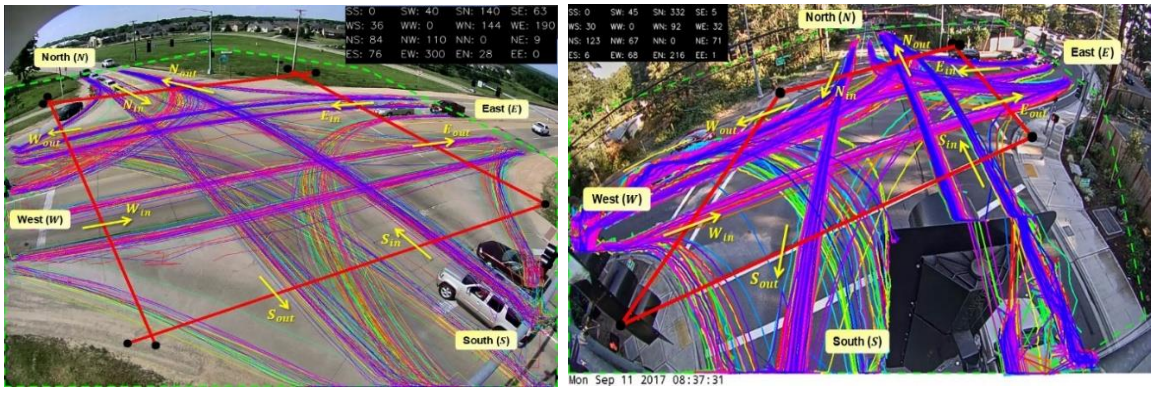
Table 10. Summary of turn movement count results across datasets and scenes

Dataset	Scene	Duration	GT	TP	FP	FN	F1(%)	Speed (fps)
AICity 2021	Cam 6	30 min	1262	1216	4	46	97.99%	17.16
	Cam 9	5 min	90	90	0	0	100.00%	15.69
Bellevue	116th at NE8h	15 min	650	625	9	25	97.35%	22.08
	150th at Newport	30 min	1122	1086	6	36	98.10%	22.10
LADOTD-Drone	Airline West	4:38 min	1189	1160	1	28	99.57%	17.50
	Airline East	3:48 min	1282	1228	0	54	99.62%	17.56
	Plank at 72nd	3 min	95	92	0	3	98.39%	20.25
	Plank at Evangeline	3 min	42	42	0	0	100.00%	20.3
	Plank at Choctaw	3 min	51	50	2	0	98.03%	11.82

The evaluation results in Table 10 demonstrate that the proposed TMC framework achieves consistently high accuracy across diverse intersection scenes. On the AICity2021 dataset, the system attains near-perfect detection on shorter sequences (Cam 9, 100% F1-micro) and maintains robust performance on longer sequences (Cam 6, 97.99%). The Bellevue dataset shows similarly strong outcomes, with F1-micro above 97% even under different traffic and lighting conditions. For the LADOTD-Drone dataset, which contains higher-resolution aerial views and shorter clips, the framework achieves F1-micro values above 98% in all scenes, including cases with perfect or near-perfect precision and recall (Airline East and Plank at Evangeline). Processing speeds range from 11.8 to 22.1 fps, depending on resolution and scene complexity, confirming that the approach is capable of real-time performance.

The turn-count algorithm itself is designed to be event-driven, focusing on key line-crossing events at entry and exit points rather than relying on continuous trajectories or complex reidentification networks. This design ensures that turn counts remain accurate as long as the vehicle is successfully detected either before or after crossing the virtual line. By leveraging this logic, the framework minimizes dependency on uninterrupted tracking, making it robust to short-term tracking inconsistencies. While detection and tracking performance may degrade when vehicles appear very small due to their distance from the camera, the accuracy of turn counts within the defined ROI remains unaffected. This robustness is attributed to the fact that vehicles naturally become more prominent in the frame as they approach the intersection, improving the detection model's accuracy and the tracking algorithm's association confidence.

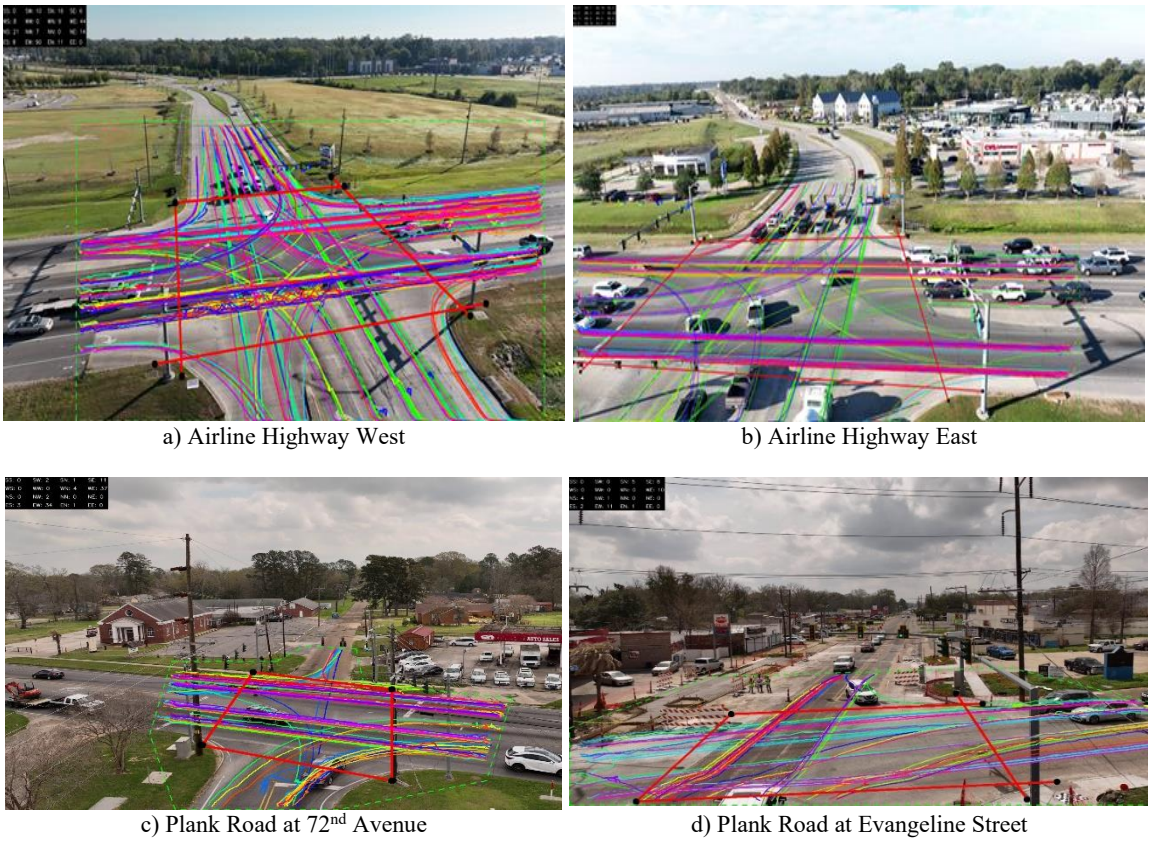
Figure 31. Examples of turn-movement trajectory visualization for fixed-camera intersections used in the TMC evaluation



AICity2021 (Cam 6) Bellevue (150th at Newport)

An inherent limitation of the proposed method occurs when objects are occluded on one side of the line and re-associated only after appearing on the opposite side, potentially missing valid line-crossing events. This challenge, which is more common in prolonged occlusions or congestion, can be mitigated with predictive tracking models or logic-based inference to approximate crossings from reassociated tracks. Despite this limitation, ByteTrack's ability to preserve tracking IDs ensures that critical entry and exit events are detected in most cases, maintaining the framework's robustness in complex scenarios. Additionally, higher processing speeds can be achieved with lighter detection models, lower input resolutions, or skipped frames, without significantly affecting turn-count accuracy, areas to be explored in future improvements.

Figure 32. Drone-based scenes of intersections used for validation of automated turn-movement counting

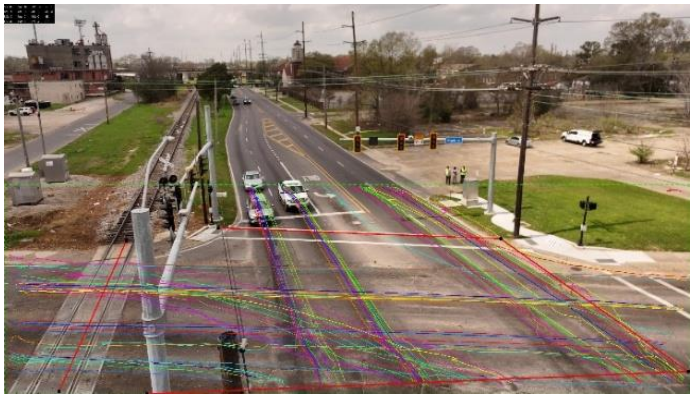


a) Airline Highway West

b) Airline Highway East

c) Plank Road at 72nd Avenue

d) Plank Road at Evangeline Street



e) Plank Road at Choctaw Drive

Performance Evaluation: Vehicle Queue Length Estimation

To evaluate the performance sampling interval of τ_e, vehicle queue lengths estimated by the algorithm were compared with ground truth values across three test cases. The performance for each lane j is defined as:

$$P_j = \left(1 - \frac{\mathrm{Err}_j}{\mathrm{GT}_j}\right) \times 100$$

where Err_j is the total absolute error between the AI algorithm developed and Ground Truth_j of the total number of vehicles observed in lane j. The overall average performance is then given by:

$$P = \frac{1}{m}\sum_{j=1}^{m} P_j,$$

where m is the number of lanes.

Performance Evaluation: Traffic Light State Recognition

Using the same evaluation interval of te, the traffic light states registered by the algorithm were compared with the ground truth report across multiple signal phases. For each lane j, the performance Bj is defined as:

$$B_j = \frac{T_j}{T_j + F_j} \times 100$$

where T_j is the number of correct classifications (i.e., true matches between the AI algorithm and ground truth) and F_j is the number of incorrect classifications (i.e., false matches) in lane j. The overall average performance across all lanes is given by

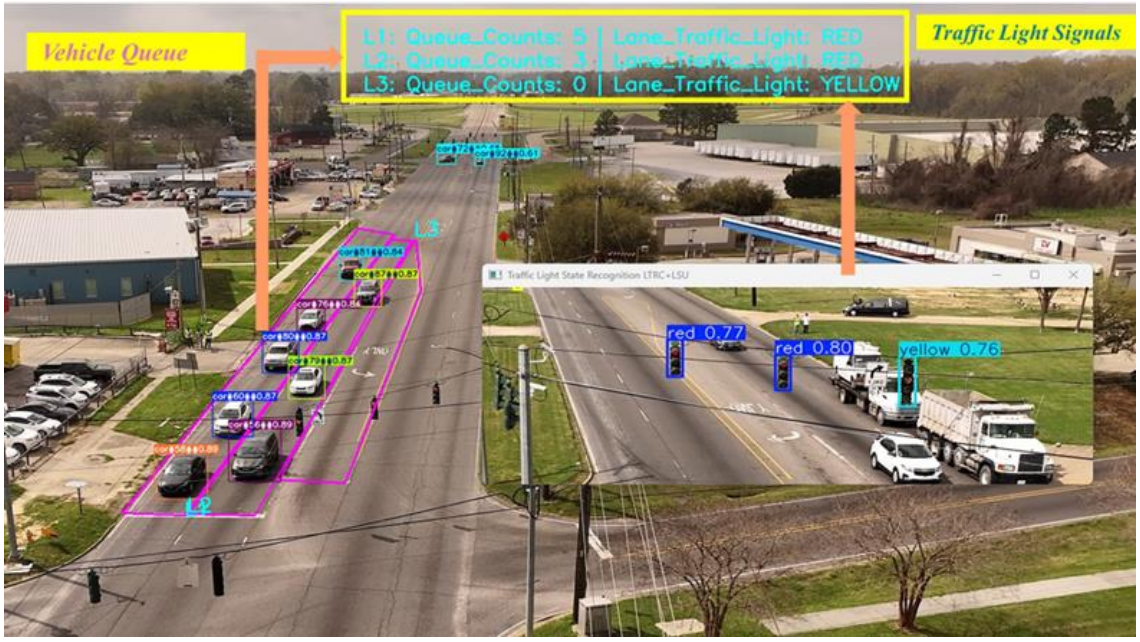
$$(B = \frac{1}{m}\sum_{j=1}^{m} B_j)$$

where m is the number of lanes considered. In the following cases, the sample video is analyzed, and the results and performance metrics defined are presented.

Plank Road at 72nd Avenue

Figure 33 provides a visualization of the video analysis. The insert shows the traffic signals, while the outer frame displays the estimated vehicle queues. From the figure, the current queue values and traffic states are highlighted within the yellow-bounded area. For the three-lane section, the estimates are $\widehat{Q_1} = 5,\ \widehat{Q_2} = 3,$ and $\widehat{Q_3} = 0$. The corresponding signal states are $\overline{S_1} = \text{Red},\ \overline{S_2} = \text{Red},\ \overline{S_3} = \text{Yellow}.$ respectively, for the three lanes. When compared with the ground truth, the estimated results show agreement, confirming the algorithm's ability to capture both the queue length and traffic signal states under multi-lane conditions.

Figure 33. Visualization of queue length estimates and traffic light states at Plank Road at 72nd Avenue

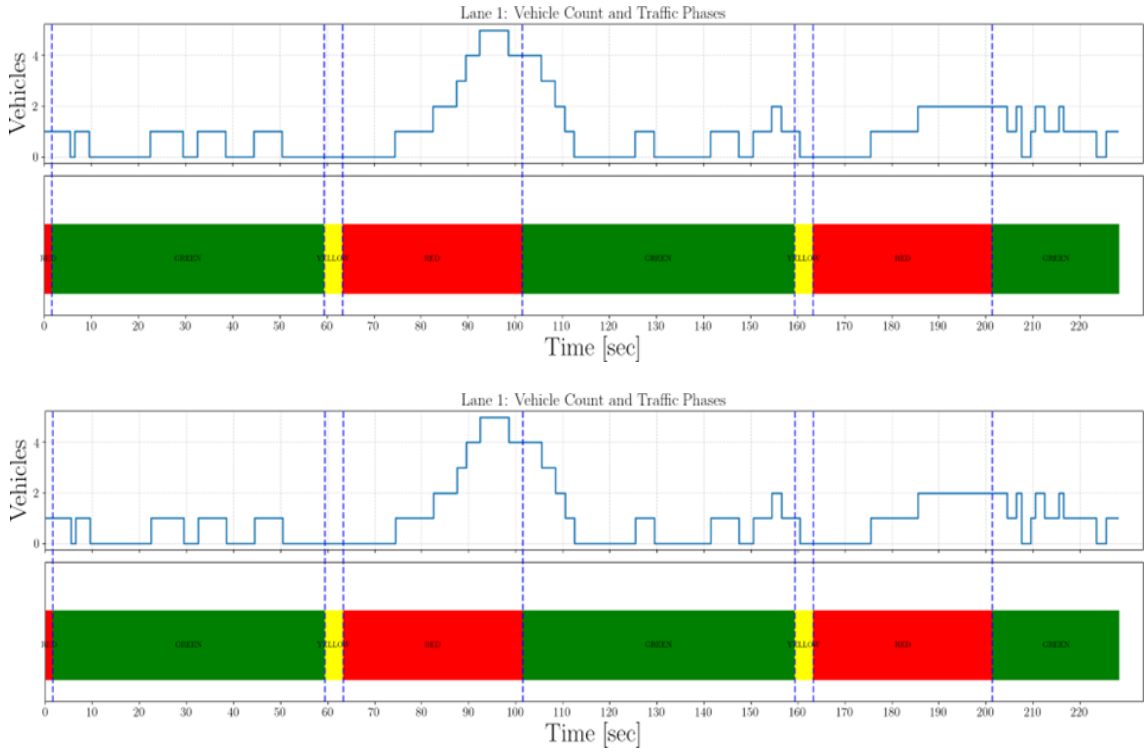


Data Analysis

Analytics on the queue and traffic light states are results from the algorithm. An example of this is Lane 1 and Lane 2, as shown in Figure 34. Figure 34(a) shows the results for Lane 1, including the variation in vehicle queue estimates alongside the corresponding traffic signal phases throughout the video. During the initial Red phase, vehicles gradually accumulate, reaching a maximum queue length of 5 before the signal transitions to Green. A slight drop in the queue count is observed, attributed to a lane change toward Lane 2, as shown in Figure 34(b).

As the signal switches to Green, vehicles begin to discharge smoothly, resulting in a sharp decrease in the queue. Overall, the results confirm that the algorithm reliably captures the dynamic buildup and dissipation of queues along with the traffic signal phases.

Figure 34. Analysis of vehicle queue length estimates and traffic light states at Plank Road at 72nd Avenue (Lanes 1 and 2)

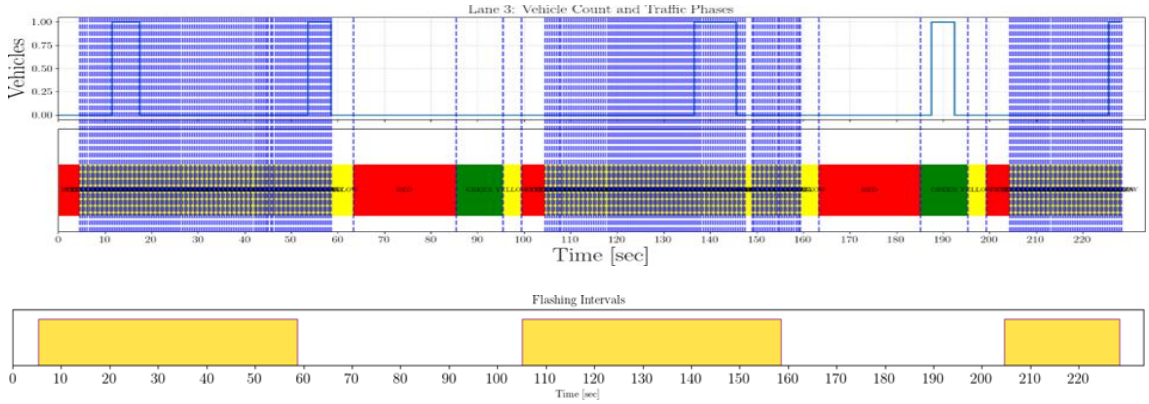


Flashing Interval

In Lane 3, a Flashing-Yellow phase is observed. This phase consists of a repeating sequence of Yellow and Off states, representing a dedicated control mode for vehicles in that lane. The algorithm accurately captured the alternating pattern of Yellow–Off–Yellow–Off transitions, confirming its sensitivity to fine-grained temporal variations in

the signal states. Additionally, a built-in time-filtering step allowed the algorithm to record the duration of each flashing phase. Figure 35 shows the detected flashing intervals within the signal timeline, while Figure 34 highlights the instantaneous state changes across consecutive frames. The region where multiple lanes show overlapping or rapidly alternating states corresponds to the flashing phase interval.

Figure 35. Flashing interval



Overall, the inclusion of this additional traffic phase, derived from a sequence of annotated Yellow and Off states, provides valuable information for intersection performance assessment. It enables more detailed decision-making regarding signal timing, safety control, and congestion impacts during transitional or cautionary phases.

Performance

The performance of the developed algorithms was quantitatively evaluated to validate their accuracy and robustness. An evaluation interval of τ_e = 10 s was chosen for both the queue-length estimation and traffic light signal recognition (TLSR) algorithms. For each interval, the algorithm's outputs were compared with manually collected ground truth data extracted directly from the video by the research team. This manual count provides the reference data for assessing performance across all three lanes.

Table 11 presents the queue-length estimation results for the three-lane section at Plank Road at 72nd Avenue, where the ground truth vehicle counts (GT) are compared with the algorithm's estimated values (Est) and their corresponding absolute errors (Err). The results show that the queue estimation algorithm achieved an overall accuracy of 98.3%, demonstrating strong reliability in identifying the dynamic buildup and dissipation of vehicle queues during different traffic signal phases. To evaluate the traffic light signal recognition (TLSR) algorithm, results at 10-second intervals were compared with ground truth labels derived from the video data. For clarity, in Table 12, labels 1–3 correspond to

the states: Red, Yellow, Green. The accuracy of each classification is indicated by T (true) or F (false), reflecting agreement or disagreement with the ground truth.

As shown in Table 12, the TLSR algorithm demonstrated strong performance across all lanes at the Plank Road at 72nd Avenue intersection. Overall, the traffic light state closely matched the manually verified ground truth, achieving an overall accuracy of 98.6%, with only one minor misclassification observed in Lane 3. This high accuracy highlights the algorithm's reliability and potential for transportation system decision-making and optimization.

Table 11. Plank Road at 72nd Avenue: vehicle queue length estimation results at 10-second intervals for a 3-lane system

Time (sec)	Time (min:sec)	Lane 1			Lane 2			Lane 3		
		GT	Est	Err	GT	Est	Err	GT	Est	Err
0	00:00	1	1	0	1	1	0	0	0	0
10	00:10	0	0	0	1	1	0	0	0	0
20	00:20	0	0	0	1	1	0	0	0	0
30	00:30	0	0	0	0	0	0	0	0	0
40	00:40	0	0	0	0	0	0	0	0	0
50	00:50	1	1	0	0	0	0	0	0	0
60	01:00	0	0	0	0	0	0	0	0	0
70	01:10	0	0	0	0	0	0	0	0	0
80	01:20	0	0	0	1	1	0	0	0	0
90	01:30	4	4	0	2	2	0	0	0	0
100	01:40	4	4	0	4	4	0	0	0	0
110	01:50	2	2	0	2	2	0	0	0	0
120	02:00	0	0	0	0	0	0	0	0	0

Time (sec)	Time (min:sec)	Lane 1			Lane 2			Lane 3		
		GT	Est	Err	GT	Est	Err	GT	Est	Err
130	02:10	0	0	0	1	1	0	0	0	0
140	02:20	0	0	0	0	0	0	1	1	0
150	02:30	0	0	0	0	0	0	0	0	0
160	02:40	1	1	0	0	0	0	0	0	0
170	02:50	0	0	0	1	1	0	0	0	0
180	03:00	1	1	0	1	1	0	0	0	0
190	03:10	2	2	0	1	1	0	1	1	0
200	03:20	2	2	0	1	1	0	0	0	0
210	03:30	1	1	0	2	2	0	0	0	0
220	03:40	1	1	0	1	0	1	0	0	0
Performance Pj (%)		100.0			95.0			100.0		
Average P (%)		98.3								

Table 12. Plank Road at 72nd Avenue: traffic light signal recognition results at 10-second intervals across three lanes (L1, L2, L3)

Time (sec)	Time (min:sec)	Lane 1			Lane 2			Lane 3		
		GT	Reg	Class	GT	Reg	Class	GT	Reg	Class
0	00:00	1	1	T	1	1	T	1	1	T
10	00:10	3	3	T	3	3	T	2	2	T
20	00:20	3	3	T	3	3	T	6	6	T

Time (sec)	Time (min:sec)	Lane 1			Lane 2			Lane 3		
		GT	Reg	Class	GT	Reg	Class	GT	Reg	Class
30	00:30	3	3	T	3	3	T	2	2	T
40	00:40	3	3	T	3	3	T	2	2	T
50	00:50	3	3	T	3	3	T	2	2	T
60	01:00	2	2	T	2	2	T	2	2	T
70	01:10	1	1	T	1	1	T	1	1	T
80	01:20	1	1	T	1	1	T	1	1	T
90	01:30	1	1	T	1	1	T	3	3	T
100	01:40	1	1	T	1	1	T	1	1	T
110	01:50	3	3	T	3	3	T	2	2	T
120	02:00	3	3	T	3	3	T	2	2	T
130	02:10	3	3	T	3	3	T	6	6	T
140	02:20	3	3	T	3	3	T	2	2	T
150	02:30	3	3	T	3	3	T	6	6	T
160	02:40	2	2	T	2	2	T	2	2	T
170	02:50	1	1	T	1	1	T	1	1	T
180	03:00	1	1	T	1	1	T	1	1	T
190	03:10	1	1	T	1	1	T	3	3	T
200	03:20	1	1	T	1	1	T	1	1	T
210	03:30	3	3	T	3	3	T	2	2	T

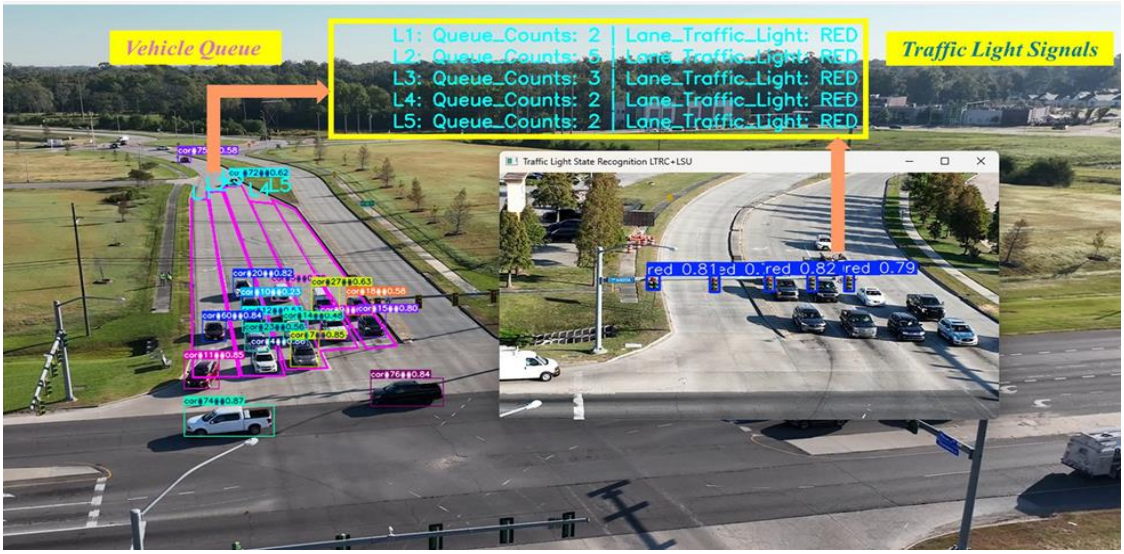
Time (sec)	Time (min:sec)	Lane 1			Lane 2			Lane 3		
		GT	Reg	Class	GT	Reg	Class	GT	Reg	Class
220	03:40	3	3	T	3	3	T	6	2	F
Accuracy Bj (%)		100.0			100.0			95.7		
Average B (%)		98.6								

State labels: 1 = Red, 2 = Yellow, 3 = Green, 6 = Off

Airline Highway at Stumberg Lane

Figure 36 provides a visualization of the algorithm run on videos from the intersection of Airline Highway at Stumberg Lane. From Figure 36, the current queue values and traffic signal states are highlighted within the yellow-bounded area. For the five-lane section, the estimated queue lengths are $\widehat{Q_1} = 2,\ \widehat{Q_2} = 5,\ \widehat{Q_3} = 3,\ \widehat{Q_4} = 2,$ and $\widehat{Q_5} = 2$. The corresponding signal states are $\overline{S_1} = \text{Red},\ \overline{S_2} = \text{Red},\ \overline{S_3} = \text{Red},\ \overline{S_4} = \text{Red},$ and $\overline{S_5} = \text{Red}$ respectively. When compared with the ground truth data, the estimated results show agreement, confirming the algorithm's ability to capture both the queue length and traffic signal states under multi-lane conditions.

Figure 36. Visualization of queue length estimates and traffic light states at Airline Highway at Stumberg Lane



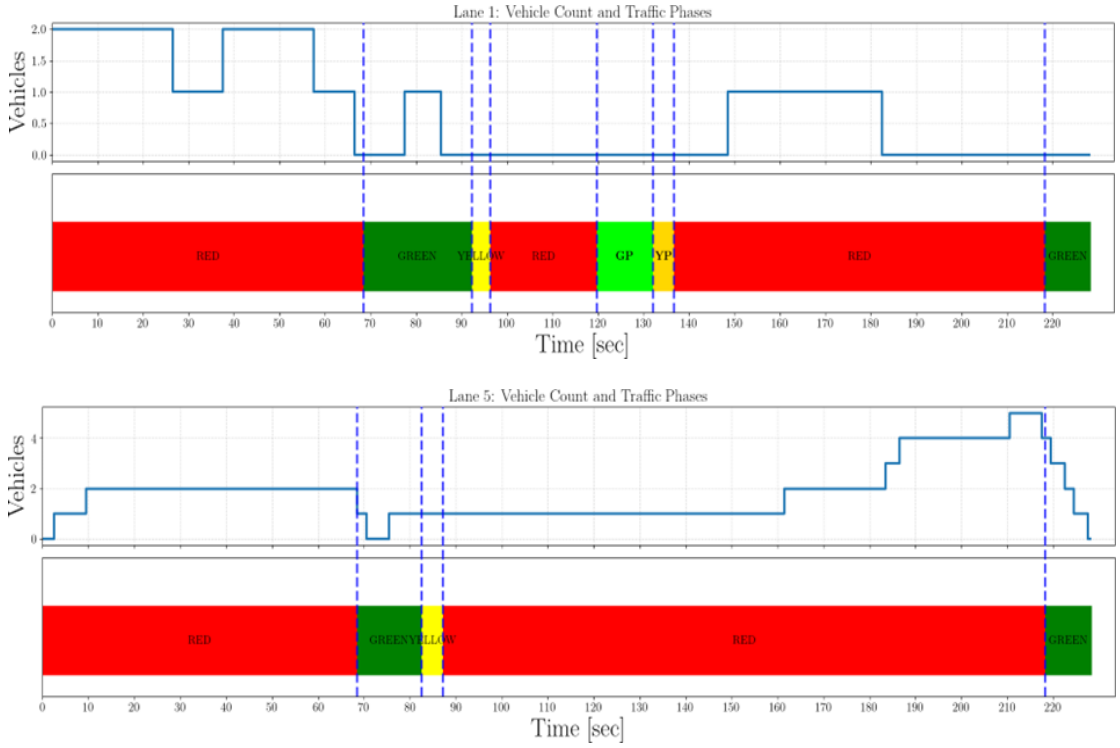
Data Analysis

The data from the algorithm provides quantitative information on both vehicle queue lengths and traffic signal states. For example, consider two lanes of varied traffic situations. Figure 37 presents the analyzed results for Lane 1 and Lane 5, showing the variation in vehicle counts over time corresponding signal phases. The queue patterns are more pronounced in Lane 5 than Lane 1.

In Lane 1, fewer vehicles are observed to accumulate, primarily due to the presence of permitted turn movements enabled by the Green-Protected (GP) and Yellow-Protected (YP) signal phases, which allow vehicles to proceed while others remain stationary in their respective lanes. During the first Red phase, vehicles begin to accumulate, reaching a maximum queue length of two. In the subsequent phase, only a single vehicle remains in the lane.

In contrast, Lane 5 experiences a larger buildup. During the initial Red phase, the queue reaches two vehicles before the signal transitions to Green, allowing for smooth discharge. During the following Red phase, vehicles accumulate more gradually, reaching a total of seven before being cleared during the next Green phase, where the queue length returns to zero. The data shown in these figures support informed evaluation of lane-specific traffic patterns. They provide insight into how signal duration, particularly during the Red phases, influences queue formation and dissipation rates across multiple lanes. Overall, the results indicate that the algorithm can effectively capture the temporal dynamics of traffic queues and signal changes, offering valuable information for traffic and transportation engineering assessments.

Figure 37. Analysis of vehicle queue length estimates and traffic light states at Airline Highway at Stumberg Lane (Lane 1 and Lane 5)



Performance

Performance evaluation was conducted using the same time interval of τ_e = 10 s for both the queue-length estimation and traffic signal state recognition algorithms. As in the previous case, the ground truth data were manually determined, with observations recorded across all five lanes. Details of the evaluation are provided in Table 14, located in the Appendix, where labels 1–6 correspond to the signal states: Red, Yellow, Green, Green-Protected, Yellow-Protected, and Off. The comparison between the ground truth

states and the algorithm's recognition (Reg) is represented as T (true) or F (false), indicating agreement or disagreement, respectively.

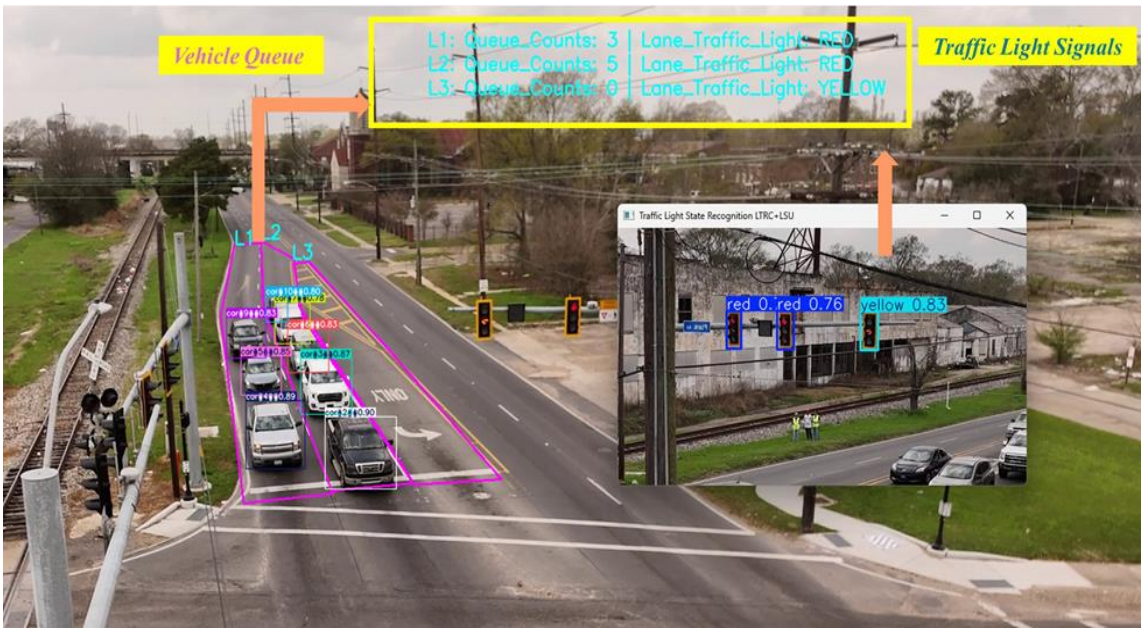
Table 13, also located in the Appendix, summarizes the performance results for the intersection of Airline Highway at Stumberg Lane. As shown, the queue-length estimation algorithm demonstrated an overall high accuracy of 99.04%. This shows strong agreement between the algorithm and the manually verified ground truth data. Also, for traffic light state recognition, the algorithm's accuracy of 100% across all lanes demonstrates strong agreement with the ground truth data.

Plank Road at Choctaw Drive

Figure 38 provides a visualization of the algorithm run on videos from the intersection of Plank Road at Choctaw Drive. The larger image shows the queue, and the inset shows the traffic light states.

The yellow-bounded region in Figure 38 shows the current queue values and traffic signal states. For the three-lane section, the estimated queue lengths are $\widehat{Q_1} = 3$, $\widehat{Q_2} = 5$, $\widehat{Q_3} = 0$, and the corresponding signal states are $\bar{S_1} = \text{Red}$, $\bar{S_2} = \text{Red}$, $\bar{S_3} = \text{Yellow}$, respectively. When compared with the ground truth data, the values and states agree with the algorithm output.

Figure 38. Visualization of queue length estimates and traffic light states at Plank Road at Choctaw Drive

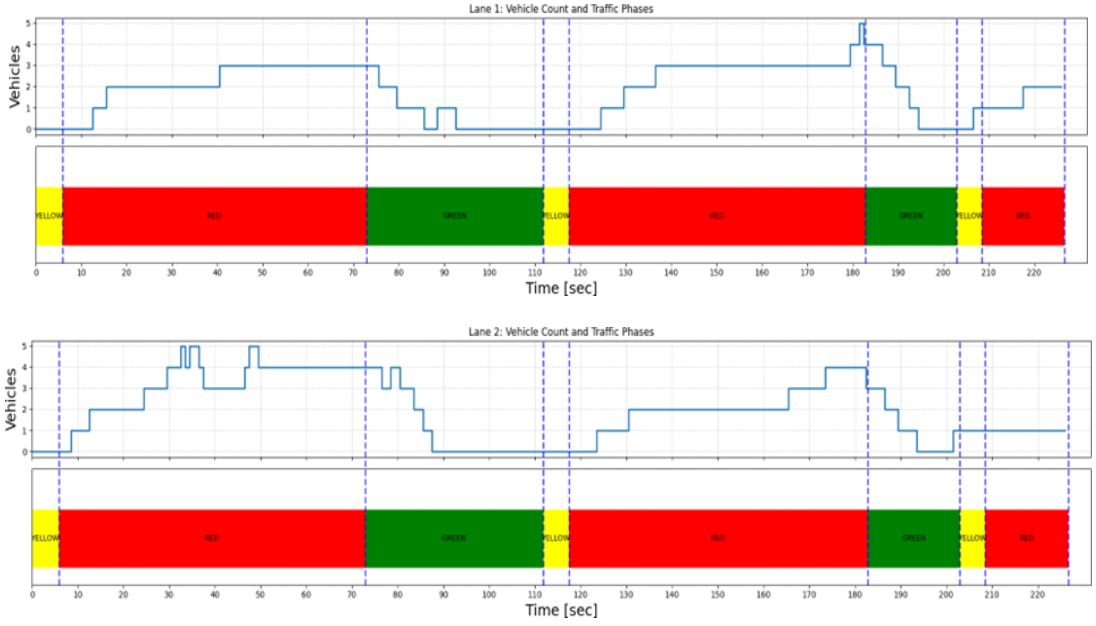


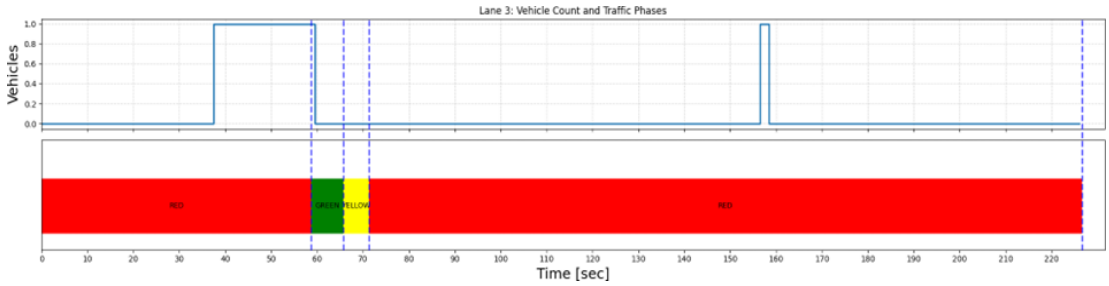
Data Analysis

Analytics on the queue and traffic light states are results from the algorithm. For instance, Lane 1 and Lane 2, as shown in Figure 39, show the results for Lane 1, including the variation in vehicle queue estimates alongside the corresponding traffic signal phases throughout the video. After the Yellow phase, traffic transitions to Red. During this phase, vehicles queue gradually accumulate, reaching a maximum length of three before the signal transitions to Green. Similar queues build up in Lane 2 to a maximum of five and finally four. In Lane 3, the queue length is experienced intermittently, with only one vehicle observed.

At signal Green, vehicles discharge smoothly, resulting in a decrease in the total queue discharge to zero. After this phase, the queue accumulates in the next Red cycle, repeating the discharge in the next Green cycle, as expected. Overall, the results confirm that the algorithm captures the dynamic buildup and discharge of queues, along with traffic signal phases. This provides information essential for augmenting existing knowledge and data to support better-informed decisions in traffic and transportation engineering.

Figure 39. Analysis of vehicle queue length estimates and traffic light states at Choctaw Drive intersection (Lanes 1–3)





Performance

Similarly, performance evaluation was conducted using the same time interval of τ_e = 10s for both the queue length estimation and traffic signal state recognition algorithms. As in the previous case, the ground truth data were manually determined, with observations recorded across three lanes. For clarity, in Table 16, located in the Appendix, labels 1–3 correspond to the signal states: Red, Yellow, and Green. The comparison between the ground truth and algorithm outputs is represented as T (true) or F (false), indicating agreement or disagreement, respectively. Table 15, also located in the Appendix, summarizes the performance for the queue length estimation algorithm. The results demonstrate an overall accuracy of 98.4%. This shows strong agreement between the algorithm and the manually verified ground truth data. Also, for traffic light state recognition, the algorithm's 100% accuracy across all lanes demonstrates strong agreement with the ground truth data.

Average Performance Across the Three Cases

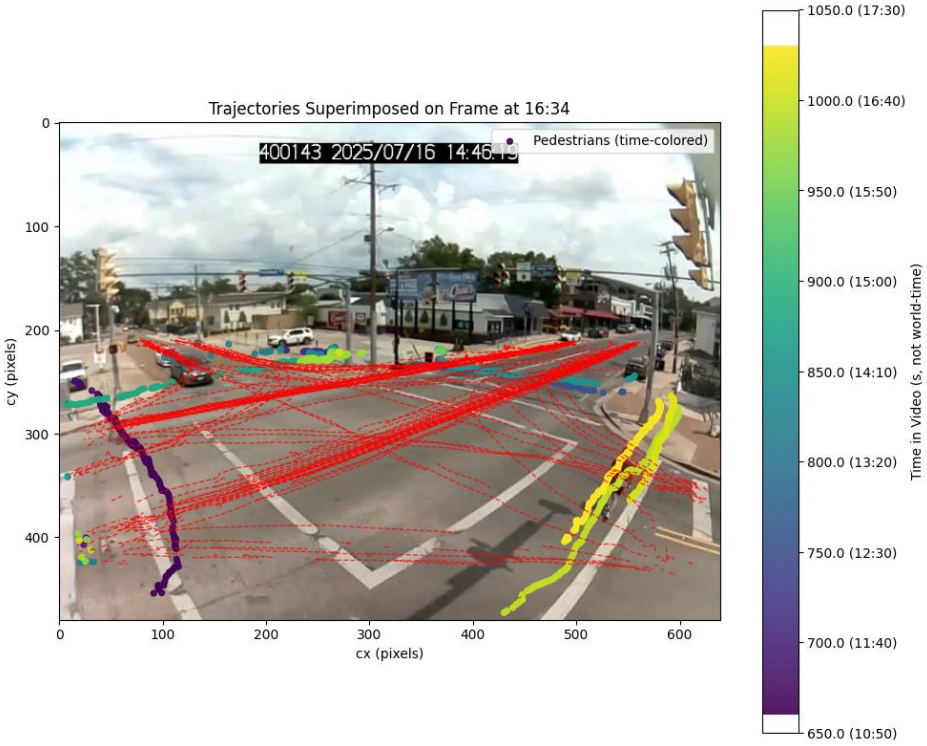
Based on the results presented in Table 11, Table 13, and Table 15, the Vehicle Queue Length Estimation algorithm achieved an average accuracy of 98.6%. This demonstrates the algorithm's ability to provide reliable information for traffic engineering decision-making while minimizing manual effort from engineers, technicians, and operators. The results in Table 12, Table 14, and Table 16 show that the traffic light signal recognition algorithms achieved an average accuracy of 99.5%. This demonstrates the effectiveness of the algorithm in providing important traffic information to improve intersection performance.

Analysis of Pedestrian-Vehicle Interaction

By capturing and analyzing the trajectories of vehicles and pedestrians at the intersection, we obtained a detailed visualization of their spatial and temporal proximity. This information forms a foundation for understanding and evaluating pedestrian behavior and interaction with vehicle motion at the intersection. Figure 40 shows the results of the interaction analysis. It

can be seen that pedestrians traverse four primary regions while multiple vehicles are in motion, highlighting key areas where interactions are more likely to occur.

Figure 40. Pedestrian-vehicle interaction analysis



Conclusions

This study demonstrates the feasibility and effectiveness of applying computer vision and artificial intelligence (AI) to the performance evaluation of signalized intersections. By leveraging video data from drones, fixed cameras, and pedestrian cameras, three methods were developed and tested: (1) a vehicle counting framework for turn movement analysis, (2) a queue detection method integrated with traffic light state recognition, and (3) pedestrian behavior and interaction analysis. Together, these tools provide a robust, data-driven basis for evaluating intersection operations.

The vehicle counting framework employed a detection-tracking-counting pipeline, combining the YOLOv11 detection model with the ByteTrack multi-object tracking algorithm. By implementing a virtual line-crossing strategy, the system was able to generate accurate turn movements across multiple lanes and approaches. The evaluation results confirmed that this approach consistently achieved F1-micro scores exceeding 97% across varied datasets, demonstrating resilience to occlusions, partial trajectory loss, and different lighting conditions.

The queue detection method extended the analysis by estimating lane-specific vehicle demand for each signal state. By associating vehicle presence within predefined regions of interest with recognized traffic light phases, the system provided detailed insights into queuing behavior and congestion dynamics. Results indicated an overall accuracy above 98% in estimating queue counts across multiple test intersections, including scenarios with complex lane configurations. Importantly, the inclusion of traffic light recognition ensured that queue estimates could be directly tied to signal timing, enabling meaningful performance evaluation and identifying potential issues such as queue spillback and signal inefficiency.

Beyond the core methods, the study also implemented supporting modules to improve robustness. An adaptive lane boundary algorithm mitigated issues caused by drone drift, keeping lane ROIs properly aligned. Trajectory extraction and analysis provided additional context for vehicle behaviors, enabling future extensions to behavior analysis and safety assessment. A graphical user interface (GUI) was developed to integrate all functionalities into a practitioner-friendly tool. This interface allows engineers to load videos, define ROIs, perform detection and counting, and export results without requiring specialized programming expertise.

Overall, the findings confirm that AI-based video analytics can deliver accurate, scalable, and non-intrusive alternatives to conventional traffic monitoring methods such as manual counts and inductive loop detectors. The proposed framework not only reduces costs and human

effort but also provides richer temporal and spatial detail, empowering engineers to make more informed decisions about intersection design, traffic signal operations, and congestion management.

In conclusion, this project establishes a solid foundation for the integration of computer vision and AI into intersection performance evaluation. By automating vehicle counting and queue detection with traffic light recognition, the developed system provides transportation agencies with actionable insights to improve the safety, efficiency, and reliability of traffic operations. With further refinement, particularly through expanded datasets, enhanced robustness under challenging conditions, and integration with adaptive traffic control, the framework has the potential to become a core component of next-generation traffic management systems in Louisiana and beyond.

Recommendations

The results of this study confirm that artificial intelligence (AI)-based vehicle counting and queue detection with traffic light recognition can be effectively applied to intersection performance evaluation. To strengthen these tools and ensure their long-term utility, the following recommendations are proposed:

For Practice

- **Deploy AI-based tools for routine monitoring**: The developed methods should be used in operational studies to replace manual counts and supplement traditional sensors. Their ability to deliver high accuracy with reduced labor makes them practical for widespread adoption.
- **Utilize the GUI for day-to-day applications:** The graphical interface should be provided to DOTD engineers and technicians to simplify adoption. Training workshops will ensure users can efficiently define ROIs, configure zones, and generate automated reports.
- **Expand video data collection campaigns:** We recommend collecting more video data across a wide range of intersections, weather scenarios, and lighting environments for future fine-tuning of the detection model.
- **Ground camera placement:** For accurate counting from ground camera videos, the field of view must cover the whole intersection area, including the point(s) at which all approaches converge. This ensures accurate capture of both entry and exit events for inbound and outbound traffic streams. Additionally, the camera must be installed at a sufficient elevation to minimize occlusions between the vehicle flows. Empirical observations indicate that very wide-angle or fisheye lenses can effectively satisfy these coverage requirements. However, videos captured with ultra-wide lenses must undergo major distortion correction prior to use by the vehicle detection and tracking models. Alternatively, cameras with a narrower field of view (e.g., 55 to 70 degrees) should be mounted at least 25 ft above the ground. Manual calibration of the cameras is needed to ensure that all the approach-intersection junctions appear in the frame.
- **Optimal drone operation:** Maintaining consistent video quality of the capture scene for the intended application is challenging due to variations in intersection geometry, environmental conditions, and operational constraints at certain locations. A near-optimal

capture can be determined by following several recommendations. Based on field experiences and video review, a good relative altitude for drone video capture is 22 m (approximately 72 ft) with a 5% margin, providing a balance between scene coverage and vehicle visibility. Additionally, it is recommended that operators maintain clearance from overhead cables and orient the camera so it does not capture trees that may obscure vehicles for queue or count analysis. Also, where the intersection layout permits, the video can be captured from either side of the roadway, and the viewpoint that yields the clearest lane and vehicle visibility can be selected for analysis.

- **Synchronization at data collection point:** During the project, videos were captured nearly simultaneously by drone operators or engineers using a manual start signal. To improve performance and ensure consistency, two synchronization approaches are recommended. First, the drones should be verified to share a synchronized external "master" clock, as this facilitates accurate algorithm-based alignment of the video streams. Care must be taken to ensure the synchronization precision is achieved at the second level at a minimum, preferably at the millisecond level, since minute-level synchronization is insufficient for precise frame alignment. Second, a sound-based synchronization method can be adopted as an additional safeguard or as an alternative when high-precision clock synchronization is not feasible. In this approach, a common audio signal (e.g., a loud clap, beep, or whistle) is generated at the start or during both recordings. An optimal position should be chosen to ensure the sound is clearly captured by both drones. On-board microphones should be verified for functionality, and if unavailable, external microphones can be added. The recorded audio serves as a reliable in-video marker during post-processing; by detecting the common frequency peak corresponding to the sound event, the two videos can be precisely synchronized. This dual-video alignment approach greatly improves the accuracy of the integrated algorithm and ensures consistent timing across both video sources.

Future Research

- **Create larger, well-annotated datasets:** Current detection models, such as YOLOv11, benefit greatly from diverse training samples. Future work should prioritize building larger datasets of Louisiana-specific traffic scenes, including variations in camera angles, queue formations, and signal types. High-quality manual annotation will remain essential for improving detection accuracy.

- **Enrich dataset diversity:** Additional footage should be captured at night, in adverse weather, and in both rural and urban contexts. These conditions are underrepresented in the current dataset but are critical for achieving generalizable models.
- **Develop semi-automated annotation pipelines:** To reduce the time-intensive burden of manual labeling, future work should investigate semi-supervised or AI-assisted labeling tools, with human verification to maintain accuracy.
- **Improve queue measurement fidelity:** While this study emphasized queue counts, future research should extend methods to estimate physical queue lengths (in meters or feet) using improved calibration and homography-based mapping.
- **Integrate with adaptive signal systems:** Ultimately, data from automated vehicle counts and queue detection should feed directly into adaptive traffic signal controllers, creating a closed-loop system for real-time intersection management.

Follow-Up Activities

- Conduct a systematic evaluation of tethered-drone hardware options, including regulatory considerations, commercial and open-source systems, and performance factors such as endurance, camera quality, and operating height, to identify the most suitable platform for enhancing video capture in future deployments**.**
- Enhance the software's recognition capabilities by developing automated tools that identify roadway features, such as rutting, true vehicle paths, damaged infrastructure, and vulnerable road users, to help engineers evaluate intersections more quickly and make better-informed design and maintenance decisions.
- Develop an AI-driven workflow to calibrate highway VISSIM models, automating the estimation of key parameters such as vehicle speeds, heavy-vehicle percentages, and traffic volumes, to improve simulation accuracy and reduce manual engineering effort.

Acronyms, Abbreviations, and Symbols

Term	Description
AI	Artificial Intelligence
CNN	Convolutional Neural Network
DeepSORT	Deep Simple Online and Realtime Tracking
DOTD	Louisiana Department of Transportation and Development
FHWA	Federal Highway Administration
FFCA	Feature-Enhancement, Fusion, and Context-Aware
FPS	Frames per Second
GPU	Graphics Processing Unit
GT	Ground Truth
HOG	Histogram of Oriented Gradients
HOTA	Higher Order Tracking Accuracy
ICE	Intersection Control Evaluation
Reg	Recognized State
IoU	Intersection over Union
KLT	Kanade–Lucas–Tomasi (Feature Tracker)
LBP	Local Binary Patterns
LOS	Level of Service
LTRC	Louisiana Transportation Research Center
MOE	Measure of Effectiveness
MOI	Movement of Interest
MOTA	Multiple Object Tracking Accuracy
MOTP	Multiple Object Tracking Precision
ORB	Oriented FAST and Rotated BRIEF
PHF	Peak Hour Factor
Re-ID	Re-Identification
ROI	Region of Interest
RT-DETR	Real-Time Detection Transformer
RTS	Rauch–Tung–Striebel (Smoothing Algorithm)
SP L-K	Sparse Pyramid Lucas–Kanade (Optical Flow)
SSD	Single Shot Multibox Detector
SVM	Support Vector Machine
TLSR	Traffic Light Signal Recognition
TMC	Turn Movement Counts

TN	True Negative
TP	True Positive
FP	False Positive
FN	False Negative
VEDAI	Vehicle Detection in Aerial Imagery
VQLE	Vehicle Queue Length Estimation
YOLO	You Only Look Once

References

[1] N. Seenouvong, U. Watchareeruetai, C. Nuthong, K. Khongsomboon and N. Ohnishi, "A computer vision based vehicle detection and counting system," in *2016 8th International Conference on Knowledge and Smart Technology (KST)*, 2016.

[2] Z. Qi, M. Li, C. Liu, M. Zhao and M. Long, "A measurement method for vehicle queue length of intersection based on image processing," in *2018 Eighth International Conference on Image Processing Theory, Tools and Applications (IPTA)*, 2018.

[3] M. Zanin, S. Messelodi and C. M. Modena, "An efficient vehicle queue detection system based on image processing," in *Proceedings of the 12th International Conference on Image Analysis and Processing*, 2003.

[4] T. Jiang, M. Cai, Y. Zhang and X. Jia, "Fast video-based queue length detection approach for self-organising traffic control," *IET Intelligent Transport Systems,* vol. 13, p. 670–676, 2019.

[5] C. Yang, J. Zhang, H. Li, H. Yu and Y. Xu, "Extraction of traffic parameters at urban intersections using the low-altitude unmanned aerial vehicle image," in *CICTP 2020*, ASCE, 2020, pp. 838-847.

[6] Z. Al-Ariny, M. A. Abdelwahab, M. Fakhry and E.-S. Hasaneen, "An efficient vehicle counting method using mask R-CNN," in *2020 International Conference on Innovative Trends in Communication and Computer Engineering (ITCE)*, 2020.

[7] M. Zinanyuca and D. Arce, "Traffic parameters acquisition system using faster R-CNN deep learning based algorithm," in *2020 IEEE ANDESCON*, 2020.

[8] W. Al Okaishi, A. Zaarane, I. Slimani, I. Atouf and M. Benrabh, "A vehicular queue length measurement system in real-time based on SSD network," *Transport and Telecommunication Journal,* vol. 22, p. 29–38, 2021.

[9] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu and A. C. Berg, "SSD: Single shot multibox detector," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[10] Z. Luo, F. Branchaud-Charron, C. Lemaire, J. Konrad, S. Li, A. Mishra, A. Achkar, J. Eichel and P.-M. Jodoin, "MIO-TCD: A new benchmark dataset for vehicle classification and localization," *IEEE Transactions on Image Processing,* vol. 27, p. 5129–5141, 2018.

[11] J. Zhu, K. Sun, S. Jia, Q. Li, X. Hou, W. Lin, B. Liu and G. Qiu, "Urban traffic density estimation based on ultrahigh-resolution UAV video and deep neural network," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing,* vol. 11, p. 4968–4981, 2018.

[12] H. Song, H. Liang, H. Li, Z. Dai and X. Yun, "Vision-based vehicle detection and counting system using deep learning in highway scenes," *European Transport Research Review,* vol. 11, p. 1–16, 2019.

[13] G. Amato, L. Ciampi, F. Falchi and C. Gennaro, "Counting vehicles with deep learning in onboard UAV imagery," in *2019 IEEE Symposium on Computers and Communications (ISCC)*, 2019.

[14] K.-H. N. Bui, H. Yi, H. Jung and J. Cho, "Video-based traffic flow analysis for turning volume estimation at signalized intersections," in *Asian Conference on Intelligent Information and Database Systems*, 2020.

[15] B. A. Mansour, "Traffic data extraction from drones using deep learning-based computer vision methods," *The University of Auckland, Department of Civil and Environmental Engineering,* 2020.

[16] A. Holla, U. Verma and R. M. Pai et al, "Efficient vehicle counting by eliminating identical vehicles in uav aerial videos," in *2020 IEEE International Conference on Distributed Computing, VLSI, Electrical Circuits and Robotics (DISCOVER)*, 2020.

[17] M. Umair, M. U. Farooq, R. H. Raza, Q. Chen and B. Abdulhai, "Efficient video-based vehicle queue length estimation using computer vision and deep learning for an urban traffic scenario," *Processes,* vol. 9, p. 1786, 2021.

[18] Y. Zhang, M. Ye, G. Zhu, Y. Liu, P. Guo and J. Yan, "FFCA-YOLO for small object detection in remote sensing images," *IEEE Transactions on Geoscience and Remote Sensing,* 2024.

[19] S. Razakarivony and F. Jurie, "Vehicle detection in aerial imagery: A small target detection benchmark," *Journal of Visual Communication and Image Representation,* vol. 34, p. 187–203, 2016.

[20] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov and S. Zagoruyko, "End-to-end object detection with transformers," in *European Conference on Computer Vision*, 2020.

[21] W. Lv, S. Xu, Y. Zhao, G. Wang, J. Wei, C. Cui, Y. Du, Q. Dang and Y. Liu, "DETRs beat YOLOs on real-time object detection," *arXiv e-prints,* p. arXiv2304, 2023.

[22] T. Ye, W. Qin, Z. Zhao, X. Gao, X. Deng and Y. Ouyang, "Real-time object detection network in UAV-vision based on CNN and transformer," *IEEE Transactions on Instrumentation and Measurement,* vol. 72, p. 1–13, 2023.

[23] W. Lu, C. Lan, C. Niu, W. Liu, L. Lyu, Q. Shi and S. Wang, "A CNN-transformer hybrid model based on CSWin transformer for UAV image object detection," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing,* vol. 16, p. 1211–1231, 2023.

[24] C.-Y. Wang, I.-H. Yeh and H.-Y. M. Liao, "YOLOv9: Learning what you want to learn using programmable gradient information," *arXiv preprint arXiv: 2402.13616,* 2024.

[25] A. Clausse, S. Benslimane and A. de La Fortelle, "Large-scale extraction of accurate vehicle trajectories for driving behavior learning," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, 2019.

[26] J.-W. Hsieh, S.-H. Yu, Y.-S. Chen and W.-F. Hu, "Automatic traffic surveillance system for vehicle tracking and classification," *IEEE Transactions on Intelligent Transportation Systems,* vol. 7, p. 175–187, 2006.

[27] D. Notz, F. Becker, T. Kühbeck and D. Watzenig, "Extraction and assessment of naturalistic human driving trajectories from infrastructure camera and radar sensors," in *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, 2020.

[28] F. Yu, H. Yan, R. Chen, G. Zhang, Y. Liu, M. Chen and Y. Li, "City-scale vehicle trajectory data from traffic camera videos," *Scientific Data,* vol. 10, p. 711, 2023.

[29] J. Heo and Y. Kwon, "3-D vehicle trajectory extraction using dcnn in an overlapping multi-camera crossroad scene," *Sensors,* vol. 21, p. 7879, 2021.

[30] O. Abdeljaber, A. Younis and W. Alhajyaseen, "Extraction of vehicle turning trajectories at signalized intersections using convolutional neural networks," *Arabian Journal for Science and Engineering,* vol. 45, p. 8011–8025, 2020.

[31] M. Shokrolah Shirazi and B. T. Morris, "Trajectory prediction of vehicles turning at intersections using deep neural networks," *Machine Vision and Applications,* vol. 30, p. 1097–1109, 2019.

[32] Y. Shen, U. Ozguner, K. Redmill and J. Liu, "A robust video based traffic light detection algorithm for intelligent vehicles," in *2009 IEEE Intelligent Vehicles Symposium*, 2009.

[33] C. Wang, T. Jin, M. Yang and B. Wang, "Robust and real-time traffic lights recognition in complex urban environments," *International Journal of Computational Intelligence Systems,* vol. 4, p. 1383–1390, 2011.

[34] Y. Ji, M. Yang, Z. Lu and C. Wang, "Integrating visual selective attention model with HOG features for traffic light detection and recognition," in *2015 IEEE Intelligent Vehicles Symposium (IV)*, 2015.

[35] Z. Shi, Z. Zou and C. Zhang, "Real-time traffic light detection with adaptive background suppression filter," *IEEE Transactions on Intelligent Transportation Systems,* vol. 17, p. 690–700, 2015.

[36] H.-K. Kim, J. H. Park and H.-Y. Jung, "Effective traffic lights recognition method for real time driving assistance systemin the daytime," *International Journal of Electrical and Computer Engineering,* vol. 5, p. 1429–1432, 2011.

[37] Z. Cai, Y. Li and M. Gu, "Real-time recognition system of traffic light in urban environment," in *2012 IEEE Symposium on Computational Intelligence for Security and Defence Applications*, 2012.

[38] X. Li, H. Ma, X. Wang and X. Zhang, "Traffic light recognition for complex scene with fusion detections," *IEEE Transactions on Intelligent Transportation Systems,* vol. 19, p. 199–208, 2017.

[39] M. P. Philipsen, M. B. Jensen, A. Møgelmose, T. B. Moeslund and M. M. Trivedi, "Traffic light detection: A learning algorithm and evaluations on challenging dataset," in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, 2015.

[40] A. Almagambetov, S. Velipasalar and A. Baitassova, "Mobile standards-based traffic light detection in assistive devices for individuals with color-vision deficiency," *IEEE Transactions on Intelligent Transportation Systems,* vol. 16, p. 1305–1320, 2014.

[41] G. Siogkas, E. Skodras and E. Dermatas, "Traffic lights detection in adverse conditions using color, symmetry, and spatiotemporal information," in *Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP)*, 2012.

[42] M. Diaz-Cabrera, P. Cerri and P. Medici, "Robust real-time traffic light detection and distance estimation using a single camera," *Expert Systems with Applications,* vol. 42, p. 3911–3923, 2015.

[43] C.-C. Chiang, M.-C. Ho, H.-S. Liao, A. Pratama and W.-C. Syu, "Detecting and recognizing traffic lights by genetic approximate ellipse detection and spatial texture layouts," *International Journal of Innovative Computing, Information, and Control,* vol. 7, p. 6919–6934, 2011.

[44] M. Omachi and S. Omachi, "Traffic light detection with color and edge information," in *2009 2nd IEEE International Conference on Computer Science and Information Technology*, 2009.

[45] M. Omachi and S. Omachi, "Detection of traffic light using structural information," in *IEEE 10th International Conference on Signal Processing Proceedings*, 2010.

[46] A. E. Gomez, F. A. Alencar, P. V. Prado, F. S. Osorio and D. F. Wolf, "Traffic lights detection and state estimation using hidden markov models," in *2014 IEEE Intelligent Vehicles Symposium Proceedings*, 2014.

[47] R. De Charette and F. Nashashibi, "Traffic light recognition using image processing compared to learning processes," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009.

[48] R. De Charette and F. Nashashibi, "Real time visual traffic lights recognition based on spot light detection and adaptive traffic lights templates," in *2009 IEEE Intelligent Vehicles Symposium*, 2009.

[49] G. Trehard, E. Pollard, B. Bradai and F. Nashashibi, "Tracking both pose and status of a traffic light via an interacting multiple model filter," in *17th International Conference on Information Fusion (FUSION)*, 2014.

[50] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 2005.

[51] R. Szeliski, Computer Vision: Algorithms and Applications, Springer Nature, 2022.

[52] H.-K. Kim, Y.-N. Shin, S.-G. Kuk, J. H. Park and H.-Y. Jung, "Night-time traffic light detection based on svm with geometric moment features," *International Journal of Computer and Information Engineering,* vol. 7, p. 472–475, 2013.

[53] J. Gong, Y. Jiang, G. Xiong, C. Guan, G. Tao and H. Chen, "The recognition and tracking of traffic lights based on color segmentation and camshift for intelligent vehicles," in *2010 IEEE Intelligent Vehicles Symposium*, 2010.

[54] V. Haltakov, J. Mayr, C. Unger and S. Ilic, "Semantic segmentation based traffic light detection at day and at night," in *German Conference on Pattern Recognition*, 2015.

[55] H. Naimi, T. Akilan and M. A. Khalid, "Fast traffic sign and light detection using deep learning for automotive applications," in *2021 IEEE Western New York Image and Signal Processing Workshop (WNYISPW)*, 2021.

[56] N. Xiang, Z. Cao, Y. Wang and Q. Jia, "A real-time vehicle traffic light detection algorithm based on modified YOLOv3," in *2021 IEEE 4th International Conference on Electronics Technology (ICET)*, 2021.

[57] T. Sarker and X. Meng, "Traffic signal recognition using end-to-end deep learning," in *Tran-SET 2022*, ASCE, 2022, p. 182–191.

[58] A. Krizhevsky, I. Sutskever and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, 2012.

[59] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár and C. L. Zitnick, "Microsoft Coco: Common objects in context," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, 2014.

[60] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv: 1409.1556,* 2014.

[61] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[62] K. He, X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[63] L. Jiao, F. Zhang, F. Liu, S. Yang, L. Li, Z. Feng and R. Qu, "A survey of deep learning-based object detection," *IEEE Access,* vol. 7, p. 128837–128868, 2019.

[64] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu and M. Pietikäinen, "Deep learning for generic object detection: A survey," *International Journal of Computer Vision,* vol. 128, p. 261–318, 2020.

[65] H.-K. Kim, J. H. Park and H.-Y. Jung, "An efficient color space for deep-learning based traffic light recognition," *Journal of Advanced Transportation,* vol. 2018, 2018.

[66] T. V. Janahiraman and M. S. M. Subuhan, "Traffic light detection using tensorflow object detection framework," in *2019 IEEE 9th International Conference on System Engineering and Technology (ICSET)*, 2019.

[67] R. Kulkarni, S. Dhavalikar and S. Bangar, "Traffic light detection and recognition for self driving cars using deep learning," in *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, 2018.

[68] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu and A. C. Berg, "SSD: Single shot multibox detector," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, 2016.

[69] J. Müller and K. Dietmayer, "Detecting traffic lights by single shot detection," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018.

[70] J. Kim, H. Cho, M. Hwangbo, J. Choi, J. Canny and Y. P. Kwon, "Deep traffic light detection for self-driving cars from a large-scale dataset," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018.

[71] Q. Wang, Q. Zhang, X. Liang, Y. Wang, C. Zhou and V. I. Mikulovich, "Traffic lights detection and recognition method based on the improved YOLOv4 algorithm," *Sensors,* vol. 22, p. 200, 2021.

[72] S. Yan, X. Liu, W. Qian and Q. Chen, "An end-to-end traffic light detection algorithm based on deep learning," in *2021 International Conference on Security, Pattern Analysis, and Cybernetics (SPAC)*, 2021.

[73] K. Wang, X. Tang, S. Zhao and Y. Zhou, "Simultaneous detection and tracking using deep learning and integrated channel feature for ambient traffic light recognition," *Journal of Ambient Intelligence and Humanized Computing,* vol. 13, p. 271–281, 2022.

[74] Z. Ouyang, J. Niu, Y. Liu and M. Guizani, "Deep CNN-based real-time traffic light detector for self-driving vehicles," *IEEE Transactions on Mobile Computing,* vol. 19, p. 300–313, 2019.

[75] F. Gao and C. Wang, "Hybrid strategy for traffic light detection by combining classical and self-learning detectors," *IET Intelligent Transport Systems,* vol. 14, p. 735–741, 2020.

[76] L. Du, W. Chen, S. Fu, H. Kong, C. Li and Z. Pei, "Real-time detection of vehicle and traffic light for intelligent and connected vehicles based on YOLOv3 network," in *2019 5th International Conference on Transportation Information and Safety (ICTIS)*, 2019.

[77] J.-G. Wang and L.-B. Zhou, "Traffic light recognition with high dynamic range imaging and deep learning," *IEEE Transactions on Intelligent Transportation Systems,* vol. 20, p. 1341–1352, 2018.

[78] M. Weber, M. Huber and J. M. Zöllner, "HDTLR: A CNN based hierarchical detector for traffic lights," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018.

[79] S. Saini, S. Nikhil, K. R. Konda, H. S. Bharadwaj and N. Ganeshan, "An efficient vision-based traffic light detection and state recognition for autonomous vehicles," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, 2017.

[80] M. B. Jensen, K. Nasrollahi and T. B. Moeslund, "Evaluating state-of-the-art object detector on challenging traffic light data," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017.

[81] K. Behrendt, L. Novak and R. Botros, "A deep learning approach to traffic lights: Detection, tracking, and classification," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[82] M. Weber, P. Wolf and J. M. Zöllner, "DeepTLR: A single deep convolutional network for detection and classification of traffic lights," in *2016 IEEE Intelligent Vehicles Symposium (IV)*, 2016.

[83] Z. Wang, Z. Deng and Z. Huang, "Traffic light detection and tracking based on Euclidean distance transform and local contour pattern," in *Proceedings of 2013 Chinese Intelligent Automation Conference*, 2013.

[84] J. Levinson, J. Askeland, J. Dolson and S. Thrun, "Traffic light mapping, localization, and state detection for autonomous vehicles," in *2011 IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[85] N. Fairfield and C. Urmson, "Traffic light mapping and detection," in *2011 IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[86] L. C. Possatti, R. Guidolini, V. B. Cardoso, R. F. Berriel, T. M. Paixão, C. Badue, A. F. De Souza and T. Oliveira-Santos, "Traffic light recognition using deep learning and prior maps for autonomous cars," in *2019 International Joint Conference on Neural Networks (IJCNN)*, 2019.

[87] M. Hirabayashi, A. Sujiwo, A. Monrroy, S. Kato and M. Edahiro, "Traffic light recognition using high-definition map features," *Robotics and Autonomous Systems,* vol. 111, p. 62–72, 2019.

[88] Z. Li, Q. Zeng, Y. Liu, J. Liu and L. Li, "An improved traffic lights recognition algorithm for autonomous driving in complex scenarios," *International Journal of Distributed Sensor Networks,* vol. 17, p. 15501477211018374, 2021.

[89] C. Jang, S. Cho, S. Jeong, J. K. Suhr, H. G. Jung and M. Sunwoo, "Traffic light recognition exploiting map and localization at every stage," *Expert Systems with Applications,* vol. 88, p. 290–304, 2017.

[90] V. John, K. Yoneda, B. Qi, Z. Liu and S. Mita, "Traffic light recognition in varying illumination using deep learning and saliency map," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2014.

[91] A. Geiger, P. Lenz and R. Urtasun, "Are we ready for autonomous driving? The kitti vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[92] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[93] X. Huang, X. Cheng, Q. Geng, B. Cao, D. Zhou, P. Wang, Y. Lin and R. Yang, "The apolloscape dataset for autonomous driving," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018.

[94] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan and O. Beijbom, "nuScenes: A multimodal dataset for autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[95] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan and T. Darrell, "Bdd100k: A diverse driving dataset for heterogeneous multitask learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[96] A. Fregin, J. Muller, U. Krebel and K. Dietmayer, "The DriveU traffic light dataset: Introduction and comparison with existing datasets," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[97] M. B. Jensen, M. P. Philipsen, A. Møgelmose, T. B. Moeslund and M. M. Trivedi, "Vision for looking at traffic lights: Issues, survey, and perspectives," *IEEE Transactions on Intelligent Transportation Systems,* vol. 17, p. 1800–1815, 2016.

[98] J. Apeltauer, A. Babinec, D. Herman and T. Apeltauer, "Automatic vehicle trajectory extraction for traffic analysis from aerial video data," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences,* vol. 40, p. 9–15, 2015.

[99] R. Khanam and M. Hussain, "YOLOv11: An overview of the key architectural enhancements," *arXiv preprint arXiv: 2410.17725,* 2024.

[100] Y. Zhang, P. Sun, Y. Jiang, D. Yu, F. Weng, Z. Yuan, P. Luo, W. Liu and X. Wang, "Bytetrack: Multi-object tracking by associating every detection box," in *European Conference on Computer Vision*, 2022.

[101] A. Telikani, A. Sarkar, B. Du, F. Santoso, J. Shen, J. Yan, J. Yong and E. Yap, "Unmanned aerial vehicle-aided intelligent transportation systems: Vision, challenges, and opportunities," *IEEE Communications Surveys & Tutorials,* pp. 1-1, 2025.

[102] J. Shi et al, "Good features to track," in *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1994.

[103] Federal Highway Administration, Traffic Monitoring Guide, Appendix C. Vehicle Types, 2014.

Appendix A

Airline Highway at Stumberg Lane Performance Evaluation

Table 13. Airline Highway at Stumberg Lane:
Vehicle queue length estimation results at 10-second intervals

Time (sec)	Time (min:sec)	Lane 1			Lane 2			Lane 3			Lane 4			Lane 5		
		GT	Est	Err	GT	Est	Err	GT	Est	Err	GT	Est	Err	GT	Est	Err
0	00:00	2	2	0	4	4	0	4	4	0	0	0	0	5	5	0
10	00:10	2	2	0	5	5	0	5	5	0	1	1	0	2	2	0
20	00:20	2	2	0	5	5	0	4	4	0	2	2	0	2	2	0
30	00:30	1	1	0	5	5	0	4	4	0	2	2	0	2	2	0
40	00:40	2	2	0	5	5	0	4	4	0	2	2	0	2	2	0
50	00:50	2	2	0	5	5	0	5	5	0	2	2	0	2	2	0
60	01:00	1	1	0	7	7	0	6	6	0	2	2	0	2	2	0
70	01:10	0	0	0	5	5	0	5	5	0	1	1	0	1	1	0
80	01:20	1	1	0	2	2	0	0	0	0	0	0	0	1	1	0
90	01:30	0	0	0	1	1	0	0	0	0	0	0	0	1	1	0
100	01:40	0	0	0	1	1	0	0	0	0	0	0	0	1	1	0
110	01:50	0	0	0	1	1	0	0	0	0	0	0	0	1	1	0
120	02:00	0	0	0	1	1	0	0	0	0	0	0	0	1	1	0
130	02:10	0	0	0	1	1	0	0	0	0	0	0	0	1	1	0
140	02:20	0	0	0	1	1	0	0	0	0	0	0	0	1	1	0
150	02:30	1	1	0	1	1	0	0	0	0	0	0	0	1	1	0
160	02:40	1	1	0	1	1	0	1	1	0	2	2	0	1	1	0
170	02:50	1	1	0	1	1	0	2	1	1	1	1	0	2	2	0
180	03:00	1	1	0	1	1	0	2	2	0	2	2	0	2	2	0
190	03:10	0	0	0	1	1	0	3	3	0	4	3	1	4	4	0
200	03:20	0	0	0	1	1	0	3	3	0	5	5	0	4	4	0
210	03:30	0	0	0	2	2	0	3	3	0	5	5	0	5	5	0
220	03:40	0	0	0	1	1	0	1	1	0	4	4	0	3	3	0
Performance *Pj* (%)		**100.0**			**100.0**			**98.1**			**97.1**			**100.0**		
Average *P* (%)		**99.04**														

Ground truth counts (GT) are compared with algorithm estimates (Est) and absolute error (Err) across five lanes.

Traffic light state recognition results at 10-second intervals across five lanes (L1–L5). Ground truth states (GT) are compared with AI results recognized (Reg), and correctness is indicated as classification outcome (Class.), with T = True and F = False. State labels: 1 = Red, 2 = Yellow, 3 = Green, 4 = Green-Protected, 5 = Yellow-Protected, 6 = Off.

Table 14. Airline Highway and Stumberg Lane:
Traffic light state recognition results at 10-second intervals across five lanes (L1–L5)

Time (s)	Time (min:s)	Lane 1			Lane 2			Lane 3			Lane 4			Lane 5		
		GT	Reg	Class.	GT	Reg	Class.	GT	Reg	Class.	GT	Reg	Class.	GT	Reg	Class.
0	00:00	1	1	T	1	1	T	1	1	T	1	1	T	1	1	T
10	00:10	1	1	T	1	1	T	1	1	T	1	1	T	1	1	T
20	00:20	1	1	T	1	1	T	1	1	T	1	1	T	1	1	T
30	00:30	1	1	T	1	1	T	1	1	T	1	1	T	1	1	T
40	00:40	1	1	T	1	1	T	1	1	T	1	1	T	1	1	T
50	00:50	1	1	T	1	1	T	1	1	T	1	1	T	1	1	T
60	01:00	1	1	T	1	1	T	1	1	T	3	3	T	3	3	T
70	01:10	3	3	T	3	3	T	3	3	T	3	3	T	3	3	T
80	01:20	3	3	T	3	3	T	3	3	T	3	3	T	3	3	T
90	01:30	3	3	T	3	3	T	3	3	T	1	1	T	1	1	T
100	01:40	1	1	T	1	1	T	1	1	T	1	1	T	1	1	T
110	01:50	1	1	T	1	1	T	1	1	T	1	1	T	1	1	T
120	02:00	1	1	T	1	1	T	1	1	T	1	1	T	1	1	T
130	02:10	4	4	T	1	1	T	1	1	T	1	1	T	1	1	T
140	02:20	4	4	T	1	1	T	1	1	T	1	1	T	1	1	T
150	02:30	1	1	T	1	1	T	1	1	T	1	1	T	1	1	T
160	02:40	1	1	T	1	1	T	1	1	T	1	1	T	1	1	T
170	02:50	1	1	T	1	1	T	1	1	T	1	1	T	1	1	T
180	03:00	1	1	T	1	1	T	1	1	T	1	1	T	1	1	T
190	03:10	1	1	T	1	1	T	1	1	T	1	1	T	1	1	T
200	03:20	1	1	T	1	1	T	1	1	T	1	1	T	1	1	T
210	03:30	1	1	T	1	1	T	1	1	T	1	1	T	1	1	T
220	03:40	3	3	T	3	3	T	3	3	T	3	3	T	3	3	T
Accuracy *Bj* (%)		**100.0**			**100.0**			**100.0**			**100.0**			**100.0**		
Average *B* (%)		**100.0**														

Plank Road at Choctaw Drive Performance Evaluation

Plank at Choctaw: Vehicle queue length estimation results at 10-second intervals for a three-lane system (Case 2). Ground truth counts (GT) are compared with algorithm estimates (Est) and absolute error (Err).

Table 15. Plank Rd at Choctaw Drive: Vehicle queue length estimation results at 10-second intervals for a three-lane system

Time (s)	Time (min:s)	Lane 1			Lane 2			Lane 3		
		GT	Est	Err	GT	Est	Err	GT	Est	Err
0	00:00	0	0	0	0	0	0	0	0	0
10	00:10	0	0	0	0	0	0	0	0	0
20	00:20	2	2	0	2	2	0	0	0	0
30	00:30	2	2	0	3	3	0	0	0	0
40	00:40	2	2	0	4	4	0	1	1	0
50	00:50	3	3	0	5	5	0	1	1	0
60	01:00	3	3	0	5	5	0	1	1	0
70	01:10	3	3	0	5	5	0	0	0	0
80	01:20	2	2	0	4	4	0	0	0	0
90	01:30	1	0	1	0	0	0	0	0	0
100	01:40	0	0	0	0	0	0	0	0	0
110	01:50	0	0	0	0	0	0	0	0	0
120	02:00	0	0	0	0	0	0	0	0	0
130	02:10	1	1	0	1	1	0	0	0	0
140	02:20	3	3	0	2	2	0	0	0	0
150	02:30	3	3	0	2	2	0	0	0	0
160	02:40	3	3	0	2	2	0	1	1	0
170	02:50	3	3	0	3	3	0	0	0	0
180	03:00	3	3	0	4	4	0	0	0	0
190	03:10	3	3	0	2	2	0	0	0	0
200	03:20	0	0	0	0	0	0	0	0	0
210	03:30	1	1	0	1	1	0	0	0	0
220	03:40	2	2	0	1	1	0	0	0	0
Performance Pj (%)		**95.2**			**100.0**			**100.0**		
Average P (%)		**98.4**								

Plank at Choctaw: Traffic light state recognition results at 10-second intervals across three lanes (L1, L2, L3). Ground truth states (GT) are compared with AI results recognized (Reg), and correctness is indicated as classification outcome (Class.), with T = True and F = False. State labels: 1 = Red, 2 = Yellow, 3 = Green.

Table 16. Plank Road and Choctaw Drive: Traffic light state recognition results at 10-second intervals across three lanes (L1, L2, L3)

Time (s)	Time (min:s)	Lane 1			Lane 2			Lane 3		
		GT	Reg	Class.	GT	Reg	Class.	GT	Reg	Class.
0	00:00	2	2	T	1	1	T	1	1	T
10	00:10	1	1	T	2	2	T	1	1	T
20	00:20	1	1	T	1	1	T	1	1	T
30	00:30	1	1	T	1	1	T	1	1	T
40	00:40	1	1	T	1	1	T	1	1	T
50	00:50	1	1	T	1	1	T	1	1	T
60	01:00	1	1	T	1	1	T	3	3	T
70	01:10	1	1	T	1	1	T	2	2	T
80	01:20	3	3	T	1	1	T	1	1	T
90	01:30	3	3	T	3	3	T	1	1	T
100	01:40	3	3	T	3	3	T	1	1	T
110	01:50	3	3	T	3	3	T	1	1	T
120	02:00	1	1	T	3	3	T	1	1	T
130	02:10	1	1	T	1	1	T	1	1	T
140	02:20	1	1	T	1	1	T	1	1	T
150	02:30	1	1	T	1	1	T	1	1	T
160	02:40	1	1	T	1	1	T	1	1	T
170	02:50	1	1	T	1	1	T	1	1	T
180	03:00	1	1	T	1	1	T	1	1	T
190	03:10	3	3	T	1	1	T	1	1	T
200	03:20	3	3	T	3	3	T	1	1	T
210	03:30	1	1	T	3	3	T	1	1	T
220	03:40	1	1	T	1	1	T	1	1	T
Accuracy Bj (%)		**100.0**			**100.0**			**100.0**		
Average B (%)		**100.0**								

Appendix B

Graphic User Interfaces

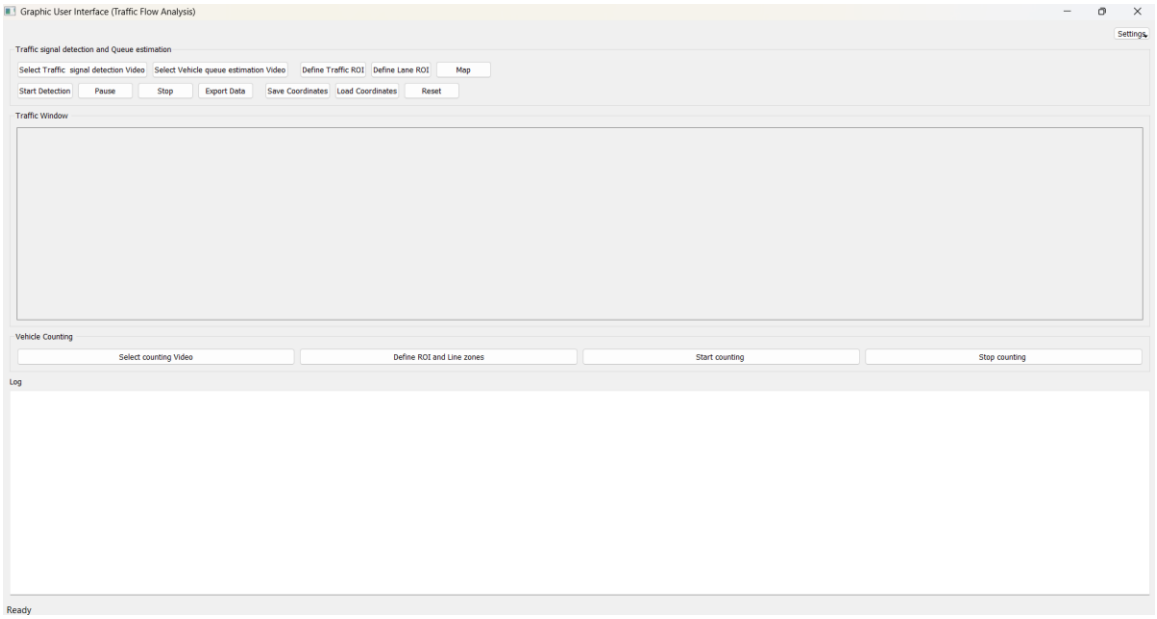
This section details the development of the Graphic User Interfaces (GUIs), allowing the operator to navigate the developed algorithm and apply it to videos collected. The GUIs provide an interactive platform for automated traffic flow and pedestrian-interaction analysis.

Traffic Flow Analysis GUI

The traffic flow analysis GUI integrates video input, region of interest (ROI) specification, cardinal directional line-zone definition, and real-time vehicle counting, trajectory tracking, and queue length estimation into a unified framework.

Each control button corresponds to a specific stage in the vehicle counting and tracking workflow; see Figure 41.

Figure 41. Overview of the traffic flow analysis GUI panel



The traffic flow analysis system is organized around two primary objectives:

- **Traffic signal detection and queue length estimation:** A dedicated traffic-light model operates on user-defined regions-of-interest (ROIs) to recognize signal phases. In addition, a vehicle-detection model estimates queue length over user-specified lane ROIs.
- **Vehicle counting and trajectory analysis:** Vehicle counting uses line-zone counting with cardinal directions (South, East, North, West). Vehicles are tracked across line

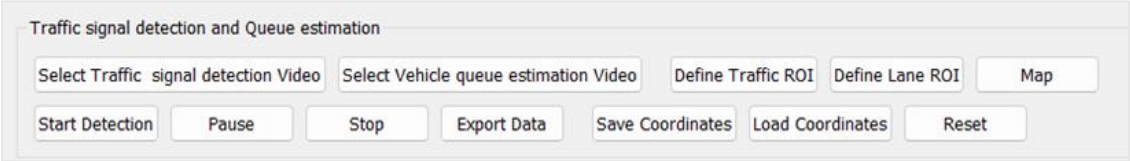
zones, and entry/exit events are logged to compute flows and turning movements. The framework also extracts trajectories and supports multi-line origin–destination (OD) analysis.

These modules allow users to load video(s), define ROIs and line zones interactively, adjust detection parameters to meet desired specifications, and control execution.

Traffic Signal Recognition and Lane Estimation Panel

The interface, pictured in Figure 42, centralizes the tools to (1) load input videos, (2) define regions-of-interest (ROIs), and (3) map each lane ROI to its corresponding traffic signal ROI.

Figure 42. Image section of the GUI panel for traffic signal recognition and lane estimation



- **Load traffic/vehicle videos:** File selection: opens the operating system's file dialog to choose videos (.mp4, .avi) for analysis.
- **Traffic-light processing:** The traffic-light video is passed to the signal-detection model, which identifies phases (Red, Yellow, Green, Protected, Off).
- **Vehicle processing:** The vehicle video is passed to the detection/tracking model to estimate queue lengths and trajectories.
- **Logging:** After loading, absolute file paths are printed to the log panel for confirmation.
- **Define Regions-of-Interest (ROIs):** Users interactively draw polygonal ROIs on selected video frames to constrain detection to meaningful areas. The ROI is drawn by sequentially placing vertices with mouse clicks to constrain detection to meaningful areas. The polygon is finalized with the Enter key. Examples of this can be seen in Figure 43 and Figure 44.
- **Traffic ROIs**: Definition occurs in two layers: (1) an outer polygon enclosing the signal panel, and (2) multiple inner polygons for individual signal lamps. Arbitrary polygon shapes are allowed, but care must be taken to minimize overlap between inner ROIs to avoid cross-detections.

- **Lane ROIs**: Lane polygons designate vehicle approach areas for accurate queue-length estimation. There is a one-to-one mapping between lane ROIs and traffic signal ROIs; each lane ROI is annotated with the same index as its corresponding signal ROI.
- **Interaction controls**: The ROI tool supports:
  - Zoom: Mouse wheel to zoom in/out for precise vertex placement
  - Undo last point: Press u; start new region: Press n.
  - Complete region: Press Enter

Figure 43. ROI examples for traffic signal analysis: (a) outer ROI and (b) individual traffic light

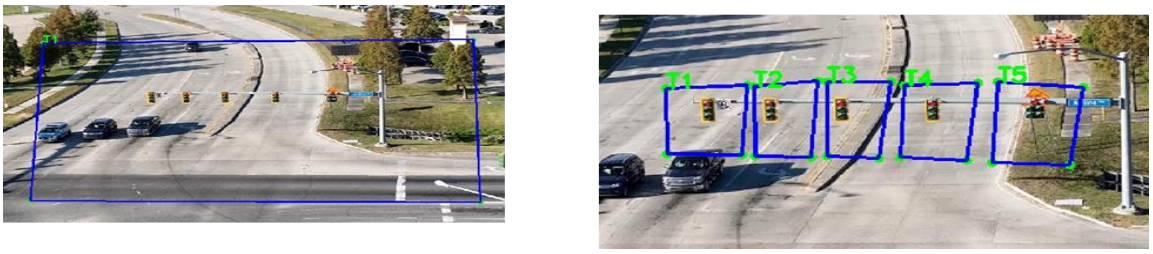
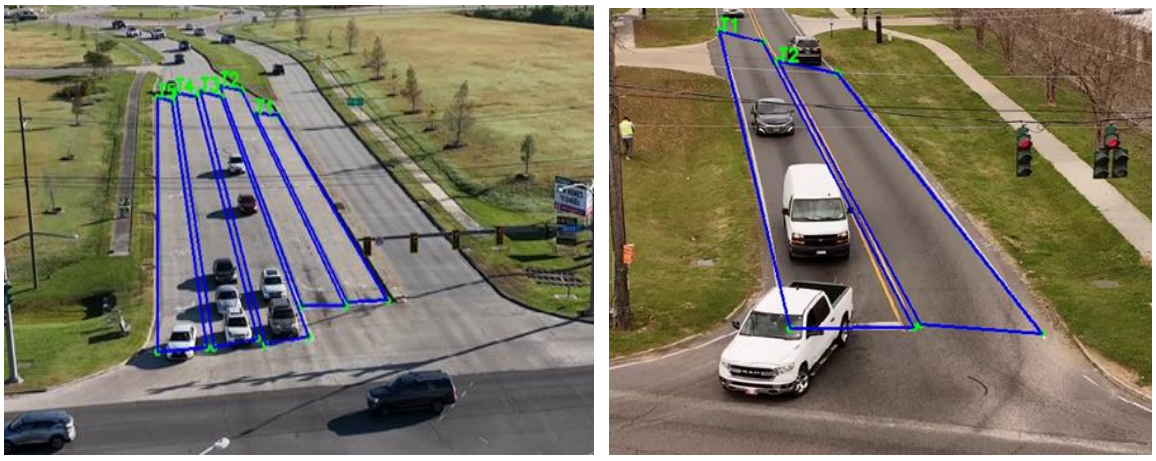


Figure 44. Selected vehicle lane regions in a cropped frame



Control Buttons

The control panel provides additional buttons to manage the processing workflow:

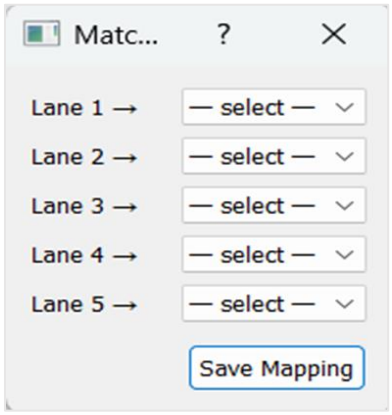
- **Start detection (shortcut Ctrl + R key):** Initializes processing and, if visualization is enabled, displays the detection and queue-length estimation views.

- **Pause/resume (shortcut = space key):** Temporarily halts video processing without releasing resources, allowing inspection of the current frame and adjustment of GUI settings. The same button is also used to resume processing without restarting.
- **Stop (shortcut Esc key):** Terminates the detection loop and releases all active video-capture streams. The GUI remains open for new runs without restarting the application.
- **Reset (shortcut R key):** Clears all loaded data (file paths, ROIs, lane–signal mappings, cached detections) and resets interface elements (log panel, video preview, button states) to their initial conditions for a fresh run.
- **Export (shortcut E key):** Generates PDF reports containing vehicle queue-length timelines and traffic signal phases formatted for transportation-engineering workflows.

MAP Button

Intersections often have multiple lanes associated with different traffic signals. The mapping feature ensures each lane region is correctly paired with its corresponding traffic signal region; this step is essential for aligning detected vehicle behavior with the correct signal phase. The mapping process is performed through a GUI dialog box where the user assigns each lane ROI to one of the traffic ROIs. The resulting mapping is used downstream in the queue duration analysis and reporting stage. Figure 45 shows a drop-down list of the typical mapping on the GUI.

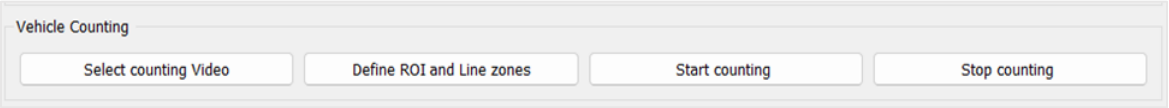
Figure 45. Mapped coordinates of selected regions



Vehicle Counting Panel

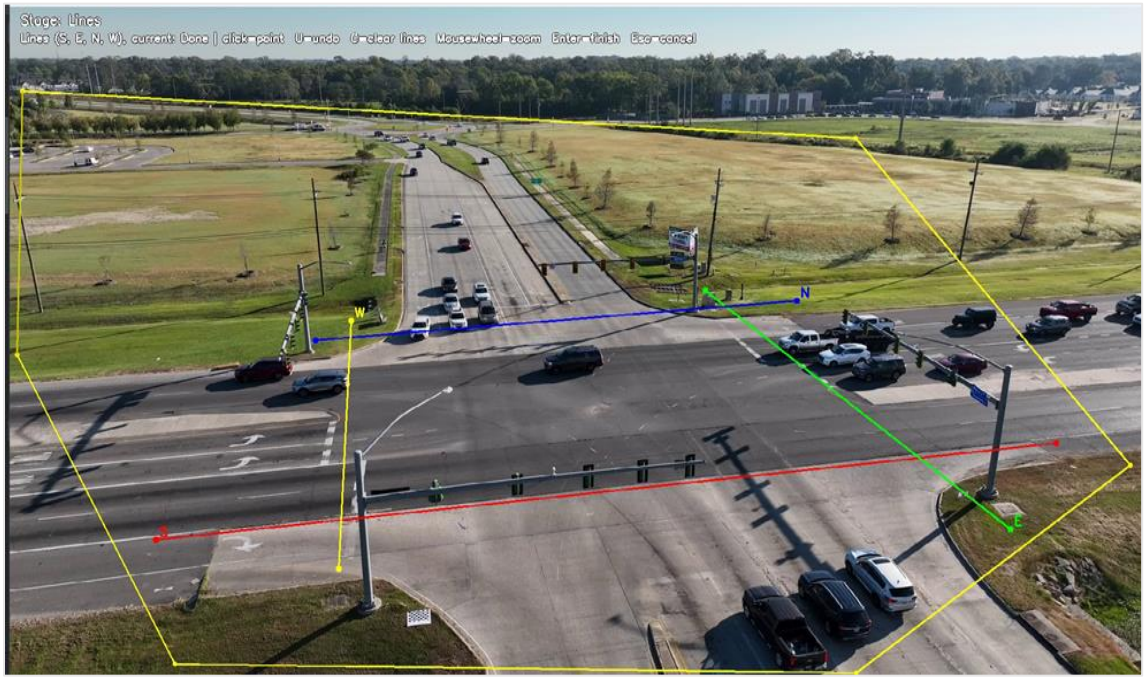
The Vehicle Counting Panel is a dedicated interface that concentrates all functionality required for vehicle detection, counting, and origin-destination tracking. The workflow is organized into sub-components, as shown in Figure 46:

Figure 46. Overview of the GUI panel for vehicle counting and tracking



- **Load Video:** Similar to the traffic signal recognition and lane estimation panel, the Load Video button opens the system file dialog to select input files in supported formats (.mp4, .avi,.mov,.mkv).
- **ROI and S/E/N/W Line Zones:** The ROI and line zones button establishes entry/exit boundaries for the counting in the cardinal directions (South, East, North, West). The system enables the user to define both the region of interest (ROI) and the directional line zones directly from the first video frame. The ROI is specified by sequentially selecting vertices with mouse clicks; once all vertices are placed, the polygon is finalized by pressing the Enter key. The resulting zone can be seen in Figure 47.
- For directional line zones, each zone is tied to a direction key and created with two mouse clicks that define start and end points, updating the counters.
- **Start Counting:** Initializes the detection and object-tracking pipeline, then processes frames with on-screen annotations (e.g., bounding boxes, IDs, and line-crossing counts).
- **Stop Counting:** Signals the processing loop to terminate via a stop event, ensuring that video handles are released and logs are finalized. Analysis outputs for vehicle counting and trajectory tracking are extracted and saved automatically in the same directory as the vehicle input video(s).

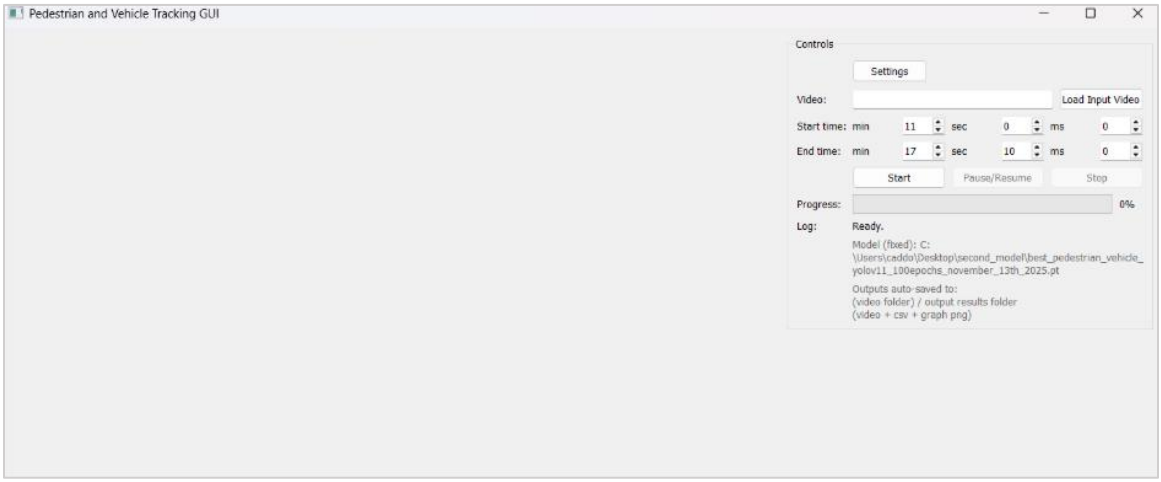
Figure 47. An image of line zones defined for vehicle tracking and counting



Pedestrian-Vehicle Interaction GUI

In the pedestrian-vehicle tracking interface shown in Figure 48 below, the left side displays the real-time view panel, while the right side provides the video selection, time-window inputs, and other control features.

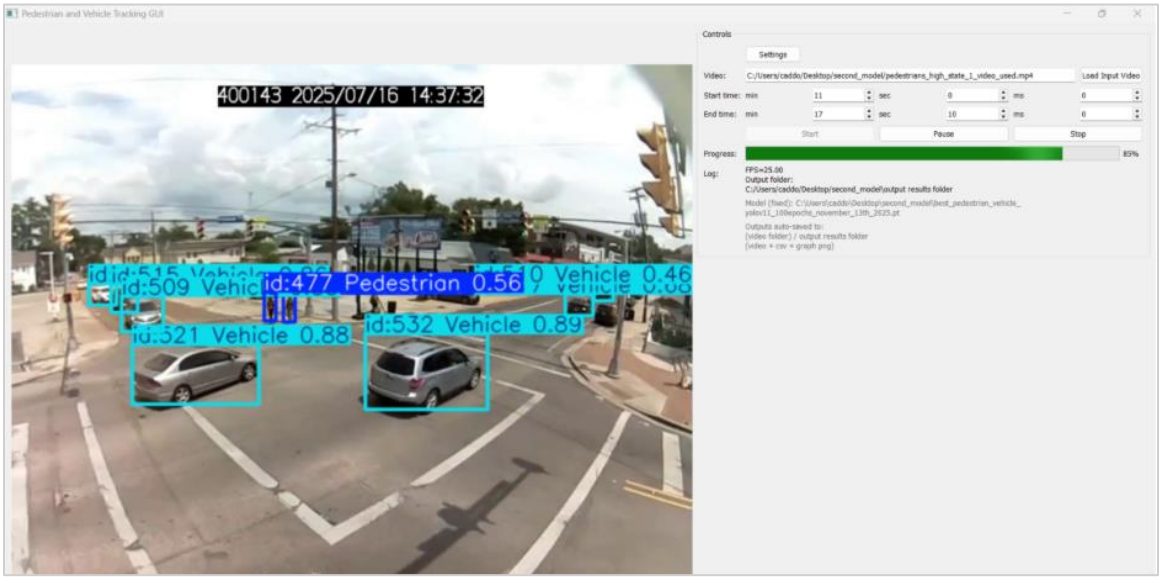
Figure 48. Overview of the pedestrian-vehicle interaction GUI panel



Steps

- For pedestrian–vehicle tracking analysis, the user selects the input video by clicking “Load Input Video.” The selected video is then used as the source for detection and tracking.
- If the user does not intend to process the entire input video, they may define a specific analysis window using the “start time” and “end time” inputs. This enables processing of only the selected segment rather than the full video duration. The GUI validates the selected time window to ensure that the end time is not earlier than the start time.
- The user initiates processing by clicking the “start” button. The GUI then analyzes the input video within the specified time window and displays the annotated results in the embedded view window. Processing progress is updated continuously, while the Log panel provides key status messages and system feedback throughout execution, as shown in Figure 49 below.

Figure 49. Pedestrian-vehicle interaction processing in the GUI panel



After processing completes, the GUI automatically saves all results to an “output results folder” created in the same directory as the input video. This folder contains the annotated output video, the exported trajectories CSV file, and the generated trajectory plot graph.