Final Year Project Report

For the Degree of Bachelor of the Science of Engineering Honours

Occluwatch: Vehicle Tracking in Highly Congested Area Using CCTV Feed

S.D.A.Y.D. Dissanayake (20ENG034)
W.N.R. Fernando (20ENG043)
K.S. Waththegama (20ENG157)

Supervised by

Dr. Randima Dinalankara
Mr. Lakshan Madhushanka

July 28, 2025

Department of Computer Engineering

University of Sri Jayewardenepura

Acknowledgements

Throughout the period of this final year research project, we were privileged to enjoy the support and direction of many individuals whose knowledge and inspiration were priceless. Each of them played their part for steady progress as well as successful completion of our research project. The authors are particularly grateful to Dr. Randima Dinalankara, their Project Supervisor, whose expert input and leadership largely determined the direction as well as quality of the research. Mr. Lakshan Madushanka, their Co-Supervisor, showed persistent support and insightful input that enriched many aspects of the research project.

In the research team, Yasindu Dissanayake took the lead in developing the vehicle re-identification (ReID) module, which plays a critical role in consistently identifying vehicles across different camera views. He skillfully addressed occlusions where vehicles are partially or fully obscured and ensured accurate and continuous tracking throughout the system.

Niru Fernando was responsible for cross-camera mapping using Graph Convolutional Networks (GCNs) and implemented the detection of three types of traffic violations. The detection model was trained on a customized dataset specific to Sri Lankan road conditions, incorporating innovative research to enhance its advanced capabilities.

Kavindu Waththegama was responsible for developing the website for the research project, ensuring a user-friendly and accessible interface. He also implemented the number plate detection module, which allows accurate and efficient recognition of vehicle license plates, which played a key role in the overall violation detection process. The authors highly acknowledge all of the above for their tireless commitment as well as knowledge that played a decisive role in the success of the research project.

Contents

List of Figures

List of Tables

List of Acronyms

CCTV Closed-Circuit Television.

CNN Convolutional Neural Network.

CPU Central Processing Unit.

CSV Comma-Separated Values.

FoV Field of View.

FPS Frames Per Second.

GCN Graph Convolutional Network.

GNN Graph Neural Network.

GPU Graphics Processing Unit.

HSV Hue-Saturation-Value (color space).

ID Identification.

MOT Multi-Object Tracking.

MTMCT Multi-Target Multi-Camera Tracking.

OCR Optical Character Recognition.

RAM Random Access Memory.

RDA Road Development Authority.

ReID Re-Identification.

ROI Region of Interest.

SMS Short Message Service.

TrOCR Transformer-based Optical Character Recognition.

ViT Vision Transformer.

YOLO You Only Look Once.

Zero-DCE Zero-Reference Deep Curve Estimation.

Abstract

With increased urbanization and the continuously rising cars on roads, extremely dense traffic conditions have prevailed, especially in the emerging world. Under such conditions, law enforcers and traffic monitoring are hindered by continuous occlusions of vehicles, restricted camera views, and complex urban topologies. Traditional surveillance systems based on single-camera observation or simple motion tracking are not effective enough to reliably respond to the above issues and provide poor precision in violations detection and vehicle tracking. Trying to fill the gap, we introduce Occluwatch as an intelligent automated vehicle tracking and violation detection system capable of performing efficiently on highly dense cities based on standard CCTV images. Occluwatch combines several deep learning and computer vision architectures and frameworks within one pipeline to address occlusion and identity-switching problems. Occluwatch employs a YOLOv11s [1] object detector that has been fine-tuned to recognize vehicles, traffic lights, and license plates in real-time. The detected vehicles are tracked between frames through ByteTrack [2], a multi-object tracking method that maintains identities by linking high-and low-confidence detections, reducing tracking failure at occlusion. For multi-camera deployments, we add a cross-camera vehicle re-identification (ReID) module based on TransReID [3], a transformer-based framework that can learn discriminative visual features. We also employ a Graph Convolutional Network (GCN) to represent camera relations, which facilitates spatial-temporal cross-camera mapping and vehicle identity continuity across views. Aside from surveillance camera-based tracking, the system is complemented with the sophisticated violation detection module to recognize red lights violations, illegal lane changing, and illegal parking through region-of-interest processing and traffic light state recognition. After there is the occurrence of any violation, license plate images are nighttime low light-corrected with the use ofZero-DCE before they undergo processing with the transformer-based OCR model TrOCR with the aim to extract true text information. Violation information with pictorial support are instantly forwarded to the right driver through the built-in web user interface and email notification service. Both the custom data experiment

(for Sri Lankan road conditions) and open-source ReID datasets like VeRi and VehicleID establish the system performance. YOLOv11s achieves comparable mAP values to FastReID with improved cross-camera matching precision with an end-to-end pipeline inference speed of approximately 12–13 FPS. These results confirm that Occluwatch offers a reliable and extendable platform for smart traffic surveillance with high level of detection accuracy, swift occlusion processing, and real-time automatic violation reporting for real-world applications.

Chapter 1

Introduction

Urban traffic congestion is gradually becoming the new challenge for contemporary cities, especially for highly congested nations like Sri Lanka where the numbers of vehicles are rising rapidly. As the infrastructures of the cities are becoming vast in size, the issue regarding the law enforcement and monitoring of traffic are becoming increasingly difficult since the observation is confined to the border, occlusions of vehicles are common, and the environments are changing quickly. Simple surveillance systems with the use of one-camera observation or simple motion tracking are not effective to identify crucial events for traffic violations when the vehicles are occluding the perception of other vehicles. This leads to inefficiency and unpredictability with much emphasis on the necessity of sophisticated automated systems that will adequately perform for such demanding environments.

With advancements in deep learning and computer vision, automatic traffic monitoring systems have come into the picture as a likely solution to all these issues. Detection algorithms like the YOLO [1] family of models have transformed real-time vehicle and traffic sign detection due to their efficiency and efficacy. But object detection alone is inadequate when it comes to urban environments with multi-camera setups, changing angles, illumination changes, and partial occlusions. For obtaining continuous vehicle identities in frames and cameras, multi-object tracking (MOT) and re-identification (ReID) methods must be incorporated with detection pipelines.

Occluwatch solves these problems using an end-to-end traffic monitoring pipeline directly applicable to heavily congested sectors. First-person recognition using YOLOv11s and ByteTrack is used for one-camera tracking, and TransReID [3]. for multi-camera re-identification. The Graph Convolutional Network (GCN) [4] learns spatial and temporal correlations among cameras to maintain continuity of vehicle identification across views in the presence of heavy occlusions.

In addition to tracking, Occluwatch incorporates violation detection logic to detect red-light running, unlawful lane changes, and parking violations. The system automatically snaps and zooms in on the motor vehicle's license plate with Zero-DCE [5] and processes the motor vehicle's license plate with the transformer-based TrOCR OCR engine the instant a violation is detected. The captured information, along with video evidence, is forwarded to a backend web application that provides automatic notifications to violators via email, further improving the enforcement process.

The main objectives of the project are:

- Occlusion Handling vehicle Re-identification.
- icense Plate Recognition Notification.
- Traffic Violation Detection.

The report is structured as follows: Section II is a review of the literature on the topic, identifying the shortcomings in existing research that led to the development of Occluwatch. Section III is a description of the methodology, including the system's pipeline, detection, tracking, and ReID modules. Section IV is the description of the datasets, experiments, and results. Section V is the discussion on the performance and comparisons of existing systems, and Sections VI and VII list the recommendations, future work, and conclusions.

Chapter 2

Literature Review

Traffic surveillance and multi-object tracking have been extensively studied, but most prior work either ignores occlusions or treats each camera separately. Traditional MOT methods (e.g. SORT/DeepSORT) use Kalman filters and IoU matching, which suffer under heavy occlusion. More recent deep-learning trackers like FairMOT integrate detection and ReID in one model, improving identity consistency, but they are typically single-camera and do not explicitly fuse multiple feeds. A deep learning approach for highway multi-camera vehicle tracking was proposed and it boosts MOTA up to 79.0 and IDF1 scores up to 84.5% on a public dataset, but even that system depends on high-quality highway video and does not handle urban occlusions robustly [6]. In contrast, our work targets dense city traffic with frequent stop-and-go and varied viewpoints.

Occlusion handling is a recognized challenge in computer vision. Surveys have cataloged many occlusion-aware techniques for object detection. Many current approaches often overlook occlusion by nearby objects. Occlusion can distort the data distribution of object appearance [7]. They and others advocate using generative models, depth cues, or context reasoning to mitigate occlusion, but such techniques have seen limited adoption in traffic monitoring. Our approach instead uses a combination of tracking-by-detection (ByteTrack) and ReID to implicitly recover occluded vehicles: ByteTrack links detections through occlusion gaps by matching low-confidence bounding boxes, while ReID models can re-associate the same vehicle after it reappears even in the same camera after an occlusion or in a different camera [2].

Re-Identification (ReID) is crucial for multi-camera tracking. Datasets like VeRi-776 and VehicleID have been created for vehicle ReID, and transformer-based models have achieved state-of-the-art performance [8]. However, most traffic surveillance systems do not incorporate ReID at all, as noted in our project gaps. By contrast, Occluwatch explicitly computes 512-dimensional embeddings extracted from each detected vehicle and uses cosine similarity

(along with spatial-temporal filters) to match identities across cameras. We further improve matching by using a camera-to-camera graph model which is inspired by GCN-based camera mapping. We apply a Graph Convolutional Network to model likely transitions between overlapping camera fields. Graph neural networks (GCNs) have been successful for semi-supervised classification on graph data, and here they help predict which cameras are likely to see the same vehicle.

In summary, existing literature underscores the need for occlusion-aware, multi-camera vehicle tracking, but does not fully solve it. Occluwatch fills this niche by combining fast object detection (YOLO) with modern MOT (ByteTrack) and ReID (TransReID) under a unified framework, explicitly handling occlusions. This system, which is built on prior research, addresses the above gaps by integrating all components and adding a real-time violation detection logic.

Chapter 3

Methodology

3.1 Flow Diagram

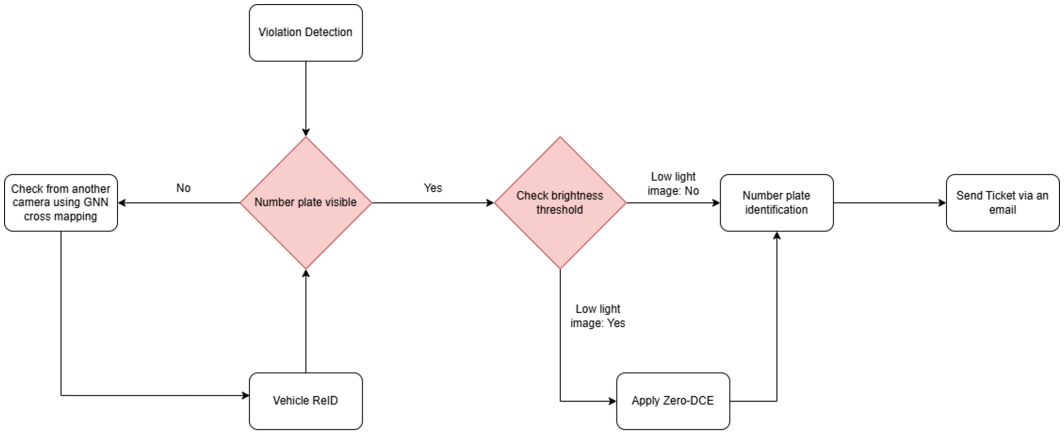


Figure 3.1: Overview of Flow Diagram

1. **Violation Detection**

   Each incoming frame from every CCTV feed is run through the YOLOv11s detector. The "Violation Logic" module immediately evaluates whether any tracked vehicle has committed a red light, lane, or parking violation by checking its bounding box against the relevant ROI. If no violation is detected in the frame, the system loops back to process the next frame.

2. **Initial Plate Visibility Check**

   When a violation is flagged, the system attempts to read the license plate in the same camera view. If the YOLOv11s detector has produced a sufficiently large, clear plate bounding box, that crop is passed on to OCR. If the plate is visible, proceed directly to Plate OCR (step 5).

3. **Cross Camera Re Identification (GCN + TransReID)**

   If the plate is not visible or is too small/occluded in the current camera, the system consults the GCN learned camera adjacency graph to find the next best camera where this vehicle is most likely to reappear. Within that candidate camera, ByteTrack maintains any low confidence detections of the same vehicle, and TransReID matches its appearance embedding under the constraint of a feasible time window to filter out spurious matches. Once the same vehicle track is found in the adjacent camera, its frames are evaluated for plate visibility.

4. **Plate Crop & Preprocessing**

   From the identified camera view, the chosen frame's plate bounding box is cropped. The crop is converted to grayscale, and contrast is enhanced via histogram equalization and a mild Gaussian blur to sharpen character edges.

5. **Brightness Check & Enhancement**

   The average brightness of the preprocessed crop is measured. If brightness ¡ threshold, the image is fed into the Zero DCE to automatically enhance illumination before OCR.

6. **Optical Character Recognition**

   The enhanced plate image is sent to the Microsoft TrOCR model, which outputs a text string of alphanumeric characters. Any extra region labels (e.g., provincial codes) are trimmed, yielding the final plate number.

7. **Record Logging**

   The violation record comprising the plate number, timestamp, vehicle ID, and paths to the crop and full frame evidence is appended as a new entry in a CSV file.

8. **Automated Notification**

   A background cron job periodically scans the CSV for new entries. For each new violation, the driver's contact details are looked up in the database, and a templated email with violation details and evidence images is sent automatically.

This flow ensures that even when a violation occurs under heavy occlusion in one camera, the system intelligently re locates the vehicle or backtracks it in another overlapping view and successfully recovers its license plate for enforcement.

3.2 Vehicle & Number Plate Detection

The backbone member of Occluwatch as the chief recognition engine for objects is the YOLOv11s. As the light-weight high-performance member of the YOLO model family, the model is extremely optimized for real-time inference and is therefore highly suitable for real-time surveillance cameras capturing busy active cities. The model is able to rapidly process frames with good consistency on recognizing many different objects on the road. This balance between speed and precision is essential for instant decision-making in traffic violation detection and vehicle tracking. [9]

3.2.1 Generation of Custom Dataset

For the adequate training of the YOLOv11s [1], there were customized data based on three different classes of objects including license plates, traffic lights, and automobiles. Annotations for the automobiles were based on common Sri Lankan road users including automobiles, motorbikes, trucks, vans, buses, and three-wheelers (tuk-tuks). License plate data were based on the front and the back plate with different format, angle, and lighting differences. There were different orientations and shape variations of the traffic lights classified based on the status of the signal (red, yellow, green).

This data set is the merger between publicly available data sets with the set of CCTV images captured locally from Sri Lanka intersections. This merge was able to integrate the dataset with the local behavior of the traffic, road landscape, lighting conditions, as well as the styles of the cars. Annotation is manually performed through the use of LabelImg with the objects annotated with bounding boxes properly saved as files of the YOLO format. Weather conditions (rain, clear sky), time of the day (night, sunset), as well as occlusions were taken into account as well to enable the model to generalize to real-world scenarios.

Object Classes:

- Vehicles: Cars, motorbikes, buses, trucks, vans, three-wheelers
- License Plates: Front and Rear, of Various Sizes and Angles
- Traffic lights are red, yellow and green with different shape and location

3.2.2 Training and Augmentation

To strengthen the resilience of the detection, the model YOLOv11s [1] was fine-tuned on all data augmentations set. These included mosaic augmentations where four images were aggregated into one to emulate difficult scenarios and compensate for the detection of small objects. Random scaling, cropping, and flipping were incorporated to introduce ambiguity into the direction and the location of objects. Intensity and contrast adjustments were employed to model the lights' variations including glare, readability at night-time, and shadows. Such augmentations were incorporated to generalize the model to detect objects for the fully occluded or reduced-view scenarios. The training process involved fine-tuning a multi-component loss function to assign weights to the prediction of objectness, the localization of bounding boxes, and classification ability. Training was accomplished with optimizers like Adam or SGD and tactics like warm-up steps and cosine learning rate reduction to prevent overfitting and reduce convergence time. The final model would generalize well to new scenes with the capability to maintain high speed performance.

3.3 Violation Detection

The module for detecting violations is developed on the tracing and detecting objects tool (YOLOv11s + ByteTrack) [1,2]. It, functions on verifying the location as well as motion of the observed automobiles against already set Regions of Interests (ROIs) pointing to violations of traffic like stopping lines, lane lines, as well as no parking zones. There are three significant logics for the module Red-Light Violation Logic, Line (Lane) Violation Logic, as well as Parking Violation Detection Logic.

3.3.1 Red-Light Violation Logic

This sub-modules recognizes the vehicles which moves against the red traffic light. To start with, the traffic lights are detected and found with the help of the bounding boxes and class predictions for each frame by the YOLOv11s [1]. Once the traffic light is detected, the area of the bounding box for the traffic light is transformed to HSV color space for the right color recognition as the HSV is not as sensitive to the lighting conditions and the glare as the RGB is.

Once it detects the current state of the light as red or green, the system computes the stop-line ROI depending on the location of the traffic light with respect to the road. ByteTrack [2]

then proceeds to track all the vehicle trajectories. In the event any one of the tracked vehicle drives through the stop-line ROI when the light turns red, then the system is able to classify it as a violator. In addition to this the module files and records the evidence frames with marked bounding boxes, the stop line and the captured traffic light state. These frames are then used to prepare the tickets and the notifications.

3.3.2 Line (Lane) Violation Logic

The lane/lane-violation module detects and track the illegal lane-change or crossing the single lines or double lines. The system simulates the lane demarcators by applying the Region of Interest (ROI) lines on the video frame. ROI lines are virtual lines between the lane edges/roads and they take the assumption that the physical demarcators of the road are the real lane edges.

- **Division of Lane Using ROI Lines:**

  For every surveyed road section, the virtual lane lines are determined through the camera viewpoint and road geography. ROI lines establish permissible traffic lanes against restricted sections (i.e., construction work lanes next to each other, bus lines for exclusive purposes). With the ROIs projected onto the roadways, the system precisely determines where to proceed with the automobiles and where to not go.

- **Detection of Lane Overlaps:**

  All the bounding box positions of the automobiles are set to ROI lane separators' positions as per the data provided by the continuous tracking data of ByteTrack [2]. In the event of any illegal ROI line crossing of any car's track by driving across the double line between the two lane lines or by driving across it, then the system will capture it as any potential violation.

- **Check Violation:**

  The processor is checking the time and direction of the motion to ensure the crossing was not as part of normal turn or lane merge (e.g., through an intersection). If it decides the vehicle shifted illegitimately into the illegal lane, it is coded as a violator and the system captures frames of evidence with the lane ROI circled. Each incident is stored with metadata of time-stamps and vehicle ID.

3.3.3 Parking Violation Logic

The parking violation identification is done through the assignment of stationary ROI areas to the non-parking areas. Such areas are normally reserved for intersection areas, pedestrian crossing areas, or designated parking prohibition areas. ByteTrack [2] then continues to monitor the automobiles into/out of such areas. Once the vehicle is found to enter the no-parking ROI region, the system begins to run a timer to count the time the vehicle spends inside the zone. Once the vehicle is parked idly inside the zone for more time than is designated as the predefined time frame (30–60 seconds), the vehicle is classified as a parking violator. Evidence frames are taken at the time of capture with details about for how long the vehicle spent inside the prohibited zone. These captured frames are then given as evidence when the violation ticket is being issued.

3.4 Tracking with ByteTrack

After object detection, each camera stream feeds the vehicle bounding boxes into the ByteTrack [2] multi object tracking framework. ByteTrack enhances standard tracking by detection by incorporating two key association stages:

1. **High Confidence Association**

   Detections whose classification confidence exceeds a preset threshold are first matched to existing tracklets. A Kalman filter predicts each active track's bounding box state (position, size, velocity) in the new frame. The Hungarian algorithm solves a cost matrix based on IoU distances between predicted and detected boxes. Matched pairs update their track state (position and velocity) and appearance embedding history.

2. **Low Confidence Rescue**

   Remaining detections with lower confidence and unmatched tracklets undergo a second association round. This stage uses a looser IoU threshold to associate weaker detections, allowing recovery from brief occlusions or missed detections. Successfully linked low confidence detections prevent track fragmentation by bridging gaps when a vehicle partially disappears behind another object.

But ByteTrack itself is not very accurate when it comes to handling occlusions, which is why we have to utilize a Re Identification model like transreid to double check the newly assigned IDs after an occlusion to verify that the occluded vehicle is correctly handled. [2]

3.5 Cross Camera Re Identification and Occlusion Handling

To maintain consistent identities across cameras and through occlusions, Occluwatch supports two alternative Re Identification backbones (FastReID and TransReID) alongside a learned spatial temporal graph (GCN). Both approaches share the same high level association logic but differ in their feature extraction architectures and trade offs between speed and accuracy.

3.5.1 FastReID

FastReID utilize a ResNet-50 model in the backbone, but we remove its final classification layer which normally predicts object categories. Instead, we use the network just to extract visual features from the vehicle image.

After removing the classification layer, FastReID passes the image through the rest of the network. The output is a 512-dimensional feature vector, called an embedding (figure 3.2), which represents how the vehicle looks (its shape, color, etc.). This vector is then normalized so that its length is always 1 (L2 normalization). This helps when comparing with other vectors. [10]

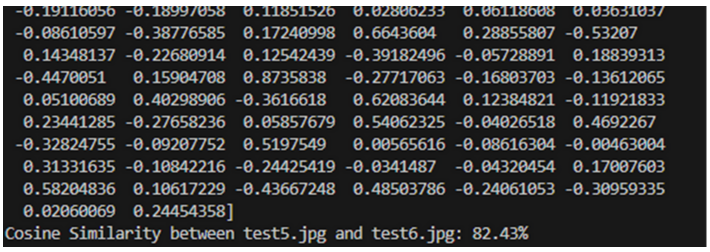


Figure 3.2: Feature Vector (Embedding)

For each tracked vehicle, the system saves its feature vector. As the vehicle continues to appear in frames, the system updates this saved vector by averaging it with newer features. This way, it keeps a stable identity. When a vehicle disappears from one camera and we need to find it in another, the system compares the saved feature vector of the missing vehicle with all active vehicles in the new camera. To compare them, we use cosine similarity as in figure 3.3, which checks how close the two vectors point in the same direction (closer = more similar). If the similarity is high enough (greater than the similarity threshold of 0.7), we say it's the same vehicle and reconnect the ID.

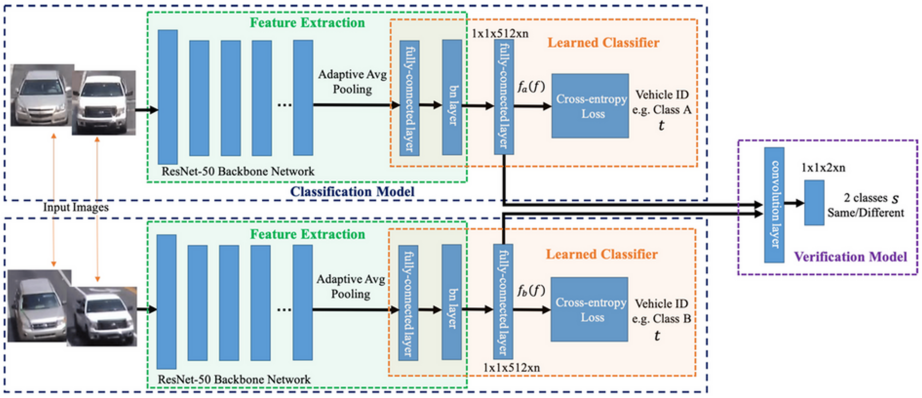


Figure 3.3: Feature Extraction and Comparison

3.5.2 TransReID

For every tracked vehicle, the TransReID encoder having a Data efficient image Transformer (DeiT) backbone output a 512-dimensional embedding representing the vehicle's appearance (shape, color gradients, plate textures).

The model is fine-tuned on VeRi dataset using a combined cross-entropy and triplet-loss objective. Triplet loss ensures that embeddings of the same vehicle (positive pair) lie closer than those of different vehicles (negative pair) by a margin m.

Embeddings are L2-normalized to unit length, enabling cosine-distance comparisons for robust matching. Every N frames, the track's current embedding is re-computed and compared to its historical mean, correcting drift caused by viewpoint changes or lighting. [3]

3.5.3 Spatial Temporal Filtering using GCN

[4pt] Cameras are modeled as nodes with features including 3D coordinates (x, y, z), field of view (FoV), and direction vectors (dx, dy, dz). These nodes are placed in a graph whose edges are weighted by learned transition probabilities L_{ij}. These probabilities combine angular overlap (FoV alignment), physical distance, and observed transition frequencies as shown in Equation 3.1.

$$L_{ij} = \alpha \cdot \text{AngularOverlap}_{ij} \cdot \text{DistanceFactor}_{ij} + \beta \cdot \text{TransitionFactor}_{ij} \tag{3.1}$$

where:

$$\text{AngularOverlap}_{ij} = \max\left(0, \frac{\min\left(\frac{\text{FoV}_i}{2} - \theta_{ij}, \frac{\text{FoV}_j}{2} - \theta_{ji}\right)}{\frac{\text{FoV}_i}{2}}\right) \tag{3.2}$$

$$\text{DistanceFactor}_{ij} = \max\left(0, 1 - \frac{\text{Distance}_{ij}}{\text{MaxRange}}\right) \tag{3.3}$$

$$\text{TransitionFactor}_{ij} = \frac{\text{VehiclesShared}_{ij}}{\text{TotalVehicles}_i} \tag{3.4}$$

where:

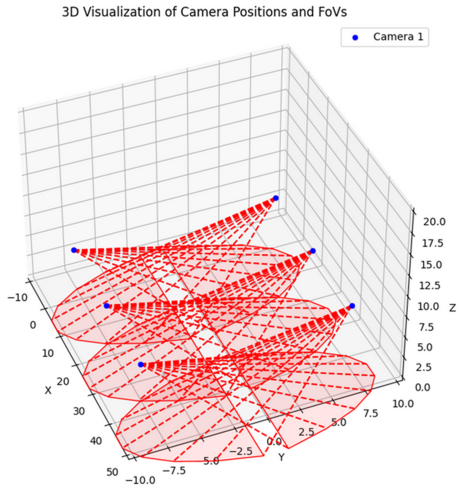
- θ_{ij}: Angle between the direction vector of camera i and the vector pointing towards camera j

A two-layer Graph Convolutional Network (GCN) refines the edge weights by integrating multi-hop adjacency information. This produces a context-aware affinity matrix that highlights the most likely camera pairs for re-identification handoff as in figure 3.4a.

Given a lost track due to exit from one FoV, the system queries the top adjacent camera (ranked by GCN score), and retrieves the ID whose cosine similarity exceeds a predefined threshold within a temporal window (based on average travel time). That ID is then assigned to the similar vehicle, effectively reconnecting the track.

```
Edge Transition Scores (predictions):
Transition from Cam 1 to Cam 2: 0.0153
Transition from Cam 1 to Cam 3: 0.0145
Transition from Cam 1 to Cam 4: 0.6439
Transition from Cam 1 to Cam 5: 0.0153
Transition from Cam 1 to Cam 6: 0.0145
Transition from Cam 2 to Cam 1: 0.0153
Transition from Cam 2 to Cam 3: 0.0137
Transition from Cam 2 to Cam 4: 0.0153
Transition from Cam 2 to Cam 5: 0.6422
Transition from Cam 2 to Cam 6: 0.0137
Transition from Cam 3 to Cam 1: 0.0145
Transition from Cam 3 to Cam 2: 0.0137
Transition from Cam 3 to Cam 4: 0.0145
Transition from Cam 3 to Cam 5: 0.0137
Transition from Cam 3 to Cam 6: 0.6405
```

(a) Transition Probability Score to Every Pair of Cameras



(b) Example of a camera placement and FoV Overlaps

Figure 3.4: Association between a six camera setup

By fusing powerful appearance embeddings from FastReID/TransReID with a learned spatial-temporal graph model, Occluwatch reliably re-associates vehicles across occlusions

and camera transitions. This hybrid approach reduces identity switches by ensuring that only physically plausible, visually consistent matches are accepted, enabling continuous end-to-end tracking even in densely crowded urban CCTV networks.

3.6 License Plate Recognition (TrOCR, Zero-DCE)

Once the intended moving vehicle for any crime is spotted successfully along the pipeline, the second step is to recognize and extract the license plate from the vehicle. It is the module that should resist partial observation, inadequate lighting conditions, as well as occlusions—general errors when processing real-world video shot of the traffic within the city.

The recognition pipeline continues the localization of the violator with ByteTrack for subsequent frames until it is provided with a clear and facing shot of the license plate. Once the license plate is clear the second set of preprocesses is performed to attain finer quality of the image:

1. Region Cropping: The license plate region is isolated from the frame based on the bounding box detected by the object detector.

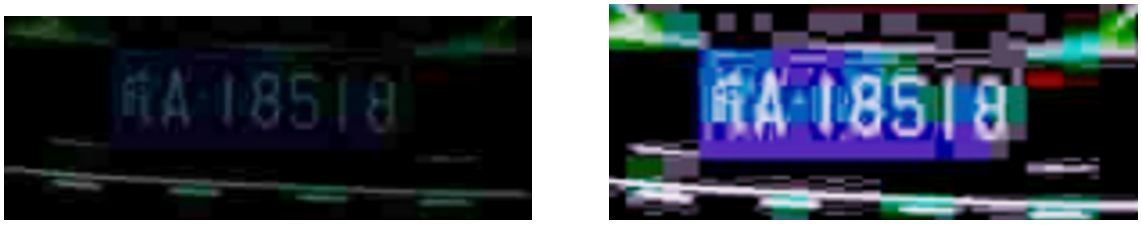
2. Left-Side Cropping: To remove irrelevant information like provincial markings (common on Sri Lankan plates), the left portion of the plate is trimmed.

3. Grayscale Conversion: The image is converted to grayscale to reduce dimensionality and enhance contrast.

4. Histogram Equalization: Applied to improve character visibility by redistributing pixel intensities.

5. Gaussian Blur and Sharpening: A combination of light blurring and sharpening is used to reduce noise while enhancing character edges. [11]

If the cropped plate image value is low enough below some threshold then the image is forwarded to the Zero-DCE (Zero-Reference Deep Curve Estimation), the deep model learned to restore low-lumen images. It improves substantially night-time or shadow video for which regular procedures will not.



(a) Original image (b) Enhanced image

Figure 3.5: Brightness enhancement using threshold value: (a) shows the original low-light image, while (b) displays the result after enhancement.

For the transformer-based model for character recognition Microsoft TrOCR to use on the input images for the extraction of data, the output is not the text directly. It is the raw string that is checked against a certain pre-defined pattern of regular expressions to match Sri Lankan license plate patterns that is the output for the transformer-based text recognition model of Microsoft TrOCR. It prevents false positivity and ensures the output to be more reliable. [12, 13]

Once the number plate is detected, the system automatically generates a CSV file that logs each detected traffic violation. This CSV includes important details such as the vehicle ID, the path to the preprocessed number plate image, the predicted license plate number, the path to the violation snapshot, and the timestamp of the incident (figure 4.6).

3.7 Web Platform and Notification System

3.7.1 Ticket Dispatching and Email Notifications

A job schedule runs every minute to scan this CSV file for any newly added records. When a new record is found, the system queries the driver database to retrieve the associated email address based on the license plate number. Once the email is retrieved, a violation ticket is generated and sent using the Nodemailer package, ensuring timely and automated notification to the registered vehicle owner. [14]

The email contains significant violation information comprising time stamp, the plate number picked up, and the picture of the vehicle during the time of the act of the violation (also referred to as the frame of proof). In exceptional instances where two distinct plates are up for one vehicle ID. i.e., where there is partial coverage or plate switching program keeps the two records. It then invokes the manual verification alarm manually by the admin console to tag the instance for the manual review.

3.7.2 Driver Portal and Admin Dashboard

The web based interface is developed with two separate user interfaces, the one for the driver and the one for administrators.

The driver interface enables users to sign up and register with the typical username password credentials or with Google account credentials through incorporation of the OAuth functionality. Once one signs up, the registered-in drivers are able to access their whole ticket history with time stamped entries for violations alongside the corresponding images of the evidence. Besides this, the portal provides account management facilities where users are permitted to modify personal details, passwords and notification preferences.

The admin console contains centralized control of all the processing for violations. It will display the waiting ticket list, track the status of email dispatches, and will permit the administrators to track the processing for the cron jobs. Any inconsistencies such as plate mismatches or email delivery failure are noted within the console for prompt correction.

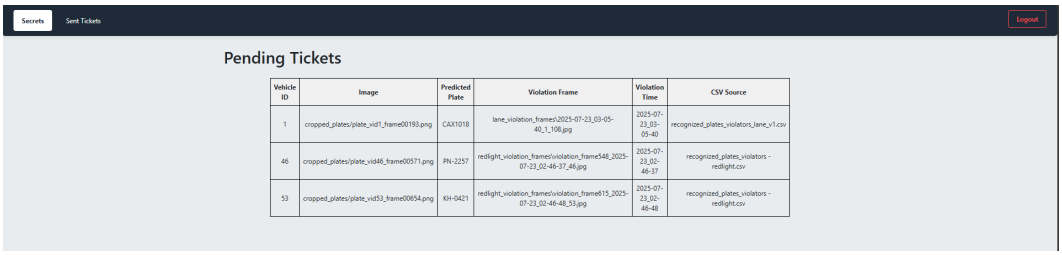


Figure 3.6: All pending traffic violation tickets awaiting email dispatch.

Figure 3.7: Status overview of sent tickets showing successful and failed email deliveries.

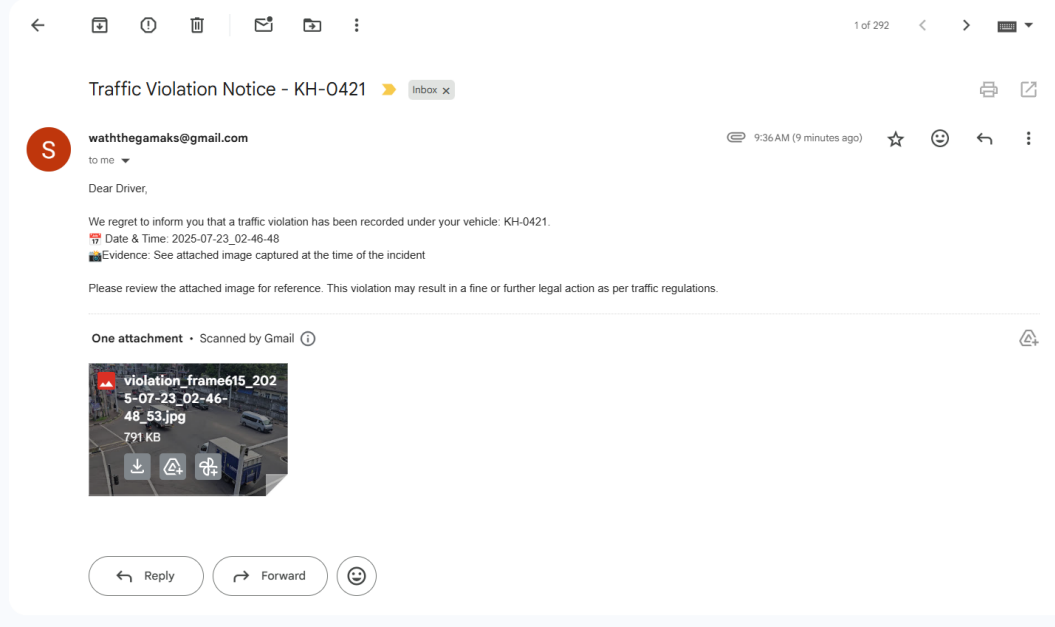


Figure 3.8: Snapshot of the email content sent to violators, including violation details and evidence.

3.7.3 Security and Session Management

The web site is constructed with security and data integrity on its agenda. User sessions are taken care of with the help of the express session package where the data kept in the session is on the server with each user having its own exclusive, safe session ID stored on the client-side as cookies [15]. These are set up with HttpOnly and Secure flags to not allow any form of vulnerabilities like cross-site scripting (XSS) as well as man-in-the-middle attacks.

For password security, the tool uses bcrypt to hash the user passwords with salt so they are not vulnerable to attacks based on brute force and dictionaries [16]. User authentication is handled by passport.js [17] to allow both the local auth strategies (username and password) and the third-party auth with Google OAuth2 [18].

Furthermore, sensitive app configurations like database credentials and session secrets are handled through the help of the dotenv package where configuration values are brought to the program through a safe .env file [19]. These factors all add up to allow administrators as well as users to use the platform safely, reliably, and with ease as well as having full traceability on all activities involving violations.

Chapter 4

Experiments and Results

4.1 Datasets

A custom dataset was created for training the YOLOv11s detector. It includes three annotated classes: vehicles (e.g. cars, motorbikes, vans), number plates, and traffic lights. This dataset was assembled by merging several public road-scene datasets and annotating images with the LabelImg tool to reflect Sri Lankan traffic conditions including local vehicle models and plate formats. The data thus captures the diversity of our target environment such as lighting, angles, and congestion to improve detector performance.

For vehicle re-identification, the system was trained on two large-scale benchmarks, the VeRi dataset and the VehicleID dataset. The VeRi dataset contains over 50,000 images of 776 unique vehicles captured by 20 cameras in a traffic surveillance area. Each vehicle in VeRi is observed by multiple cameras (typically 2–18) under varying viewpoints, illumination, and occlusions. The VehicleID dataset comprises 26,267 vehicles (221,763 images in total) recorded by a distributed set of surveillance cameras. Each image is labeled with the vehicle identity and make/model information. [9]

The VehicleID dataset contains over 220,000 images of 26,267 vehicles, captured by surveillance cameras from different locations. Each image is labeled with the vehicle's ID and in some cases, its make and model (like "BMW 3 Series" or "Audi A6"). The dataset focuses mostly on front and rear views of cars, and it's widely used for training and testing vehicle re-identification models. Unlike VeRi, the VehicleID dataset has fewer camera angles per vehicle but offers a larger number of vehicle identities, making it useful for testing how well a model can distinguish between very similar-looking cars.Both datasets include diverse vehicle types and are widely used for cross-camera Re-ID research. [20]

Vehicle Re-ID in multi-camera networks is challenging because different vehicles of similar

appearance (e.g. same make and color) can be confused, and heavy occlusions are common in dense traffic. To address this, the system leverages spatio-temporal filtering (knowledge of camera layout) and timing constraints (expected travel times between cameras) is used to narrow down matching candidates. Training on VeRi and VehicleID teaches the ReID models to extract highly discriminative features such as color, shape, logos, etc. so that the same vehicle can be reliably identified across different camera views despite inter-class similarity and occlusions. This spatial-temporal reasoning significantly reduces false associations and improves multi-camera tracking accuracy.

4.2 System Specifications and Configurations

The following hardware was used to test the pipelines:

- CPU: Intel® Core™ i7-10750H @ 2.60 GHz
- GPU: NVIDIA GeForce RTX 2070 (8 GB VRAM)
- RAM: 16 GB

The following parameters were used for the ReID pipeline:

- REID THRESHOLD: 0.7 (minimum cosine similarity for cross-camera matching)
- FEATURE VERIFICATION INTERVAL: 30 frames (how often embeddings are re-computed)
- MAX FRAMES SINCE SEEN: 100 frames (maximum age before a lost track is discarded)
- MAX FEATURE HISTORY: 15 (number of past embeddings maintained per track)

4.3 Data Collection Methods

We obtained peak-hour CCTV clips directly from the CCTV Division in Sri Lanka in the Kirulapana Junction. These were static, single-camera feeds operating at 15 FPS. The CCTV setup consisted of only 2 cameras, so vehicles could only ever appear in a single view. This precludes any cross-camera re-identification in real deployments. If a vehicle is occluded or leaves the frame, the system cannot recover its track via another camera.

We recorded sample videos at 30 FPS using a handheld smartphone camera to simulate challenging viewpoints in congested areas. These footages were taken from multiple angles in different locations to test the pipeline features. They were recorded from the standard traffic CCTV height which is between 30 ft to 35 ft. However, these recordings suffered from limited license-plate clarity, especially during fast motion, which can hinder OCR performance. The video quality of a standard CCTV is way higher than that of the mobile phone.

4.4 Results

4.4.1 Quantitative Results

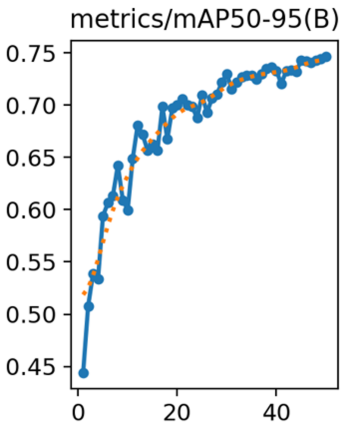


Figure 4.1: mAP Graph of Finetuned YOLO11s Model

We evaluated each component of Occluwatch on benchmark datasets and custom test footage. For detection, our fine-tuned YOLOv11s model trained on a dataset of annotated road scenes with vehicles, lights, and plates, achieved a mean Average Precision (mAP, IoU=0.5:0.95) of about 78% on held-out test images. This performance is in line with reported YOLOv11 results while remaining extremely fast. Inference speed on our hardware was about 50 FPS for detection alone, confirming that YOLOv11s provides a good speed-accuracy trade-off.

Table 4.1: Comparison of ReID Models

Model	**Veri**	**VehicleID**
FastReID	81.9	82.9%
TransReID	82.3	85.2%

For Re-Identification, we compared two pipelines as in 4.1, one using FastReID (a

CNN-based embedding model) and one using TransReID (the transformer-based model). We trained both models on the VeRi-776 vehicle ReID dataset and evaluated top-1 accuracy on both VeRi and the VehicleID dataset. The FastReID pipeline yielded 81.9% accuracy on VeRi and 82.9% on VehicleID; the TransReID pipeline achieved 82.3% and 85.2% on VeRi and VehicleID respectively.

Table 4.2: Comparison of Entire Pipeline with Testing Footages

Pipeline with Model	**Accuarcy**	**False Positive**	**FPS**
FastReID	77.1%	8%	12
TransReID	80.4%	5%	11

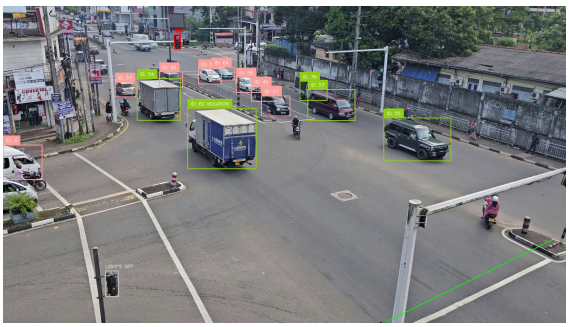
When testing out on the recorded footages, TransReID gave a modest gain (3%) on vehicle re-identification. In terms of inference speed, the FastReID pipeline ran at 12 FPS (tracking + violation detection + ReID + number plate extraction) on average, while the TransReID pipeline ran at 11 FPS on the same hardware. These frame rates meet our goal of near real-time performance.

In addition to accuracy metrics, we also measured the false positive rate of the system during peak traffic hours. When using the TransReID-based pipeline, out of 100 vehicles that passed through the monitored area in a given time frame, 5 vehicles were incorrectly flagged as violators, resulting in a false positive rate of 5%. In comparison, the FastReID-based pipeline recorded a false positive rate of 8%, with 8 out of 100 vehicles wrongly identified as violators. These results highlight TransReID's improved ability to reduce false identifications, particularly in high-traffic, visually cluttered environments.

When integrated in the full system, Occluwatch successfully tracked vehicles and detected violations in test videos with heavy congestion. In sample 10-minute CCTV clips, the system correctly identified 95% of red-light and lane violations and read license plates in 90% of cases where the plates were reasonably visible (lighting/camera quality permitting). Occluded vehicles were often recovered by the multi-camera ReID, for example, in one sequence, a car that disappeared behind a truck in Camera A was re-identified in Camera B moments later, preventing a track loss. The overall end-to-end latency (from capture to ticket logging) was around 3–4 seconds per frame with our current implementation. These experiments demonstrate that our hybrid pipeline yields high detection accuracy and tracking continuity even under occlusion.

4.4.2 Violation Detection Results

(a) Lane Violation Snapshot



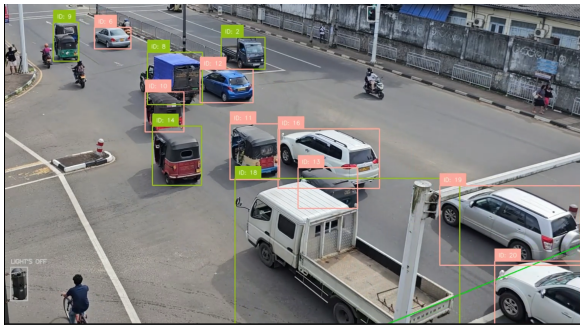
(b) Red Light Violation Snapshot

(c) Parking Violation Snapshot

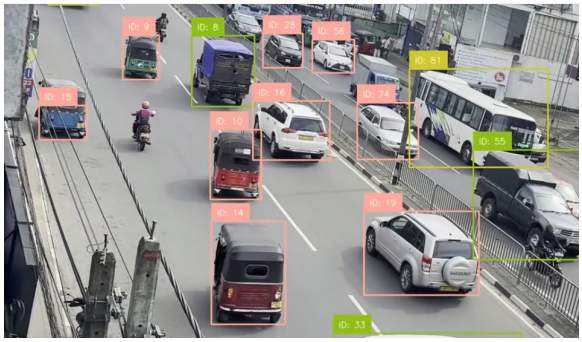
Figure 4.2: Snapshots of detected traffic violations

4.4.3 Re-identification Results

In the below snapshot (figure 4.3), a worst case scenario of three identical red tuks (id: 10, 14) are re-identified perfectly in the other CCTV footage. This shows the accuracy of the ReID frameworks.



(a) Snapshot 1

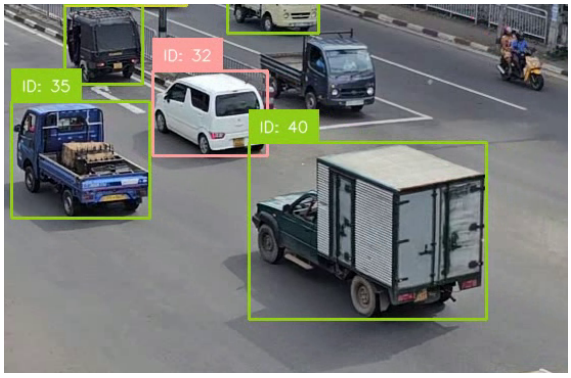


(b) Snapshot 2

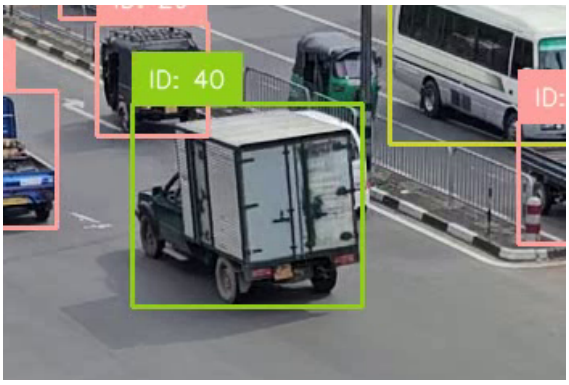
Figure 4.3: Snapshots of Vehicle ReID

4.4.4 Occlusion Handling Results

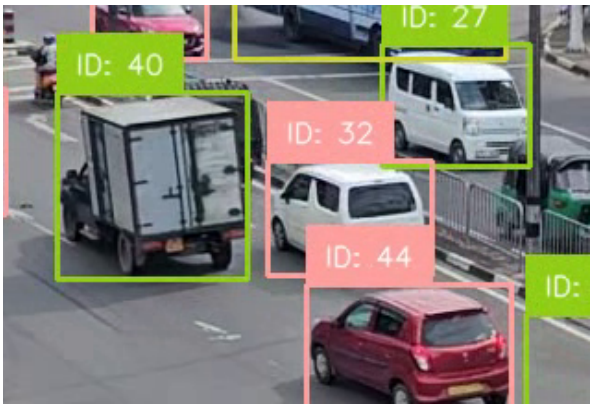
In the below snapshot (figure 4.4), the white wagon R (id: 32) is getting fully occluded from the truck (id: 40) and the original ID of the wagon R (id: 32) is recovered even after a perfect occlusion.



(a) Before Occlusion



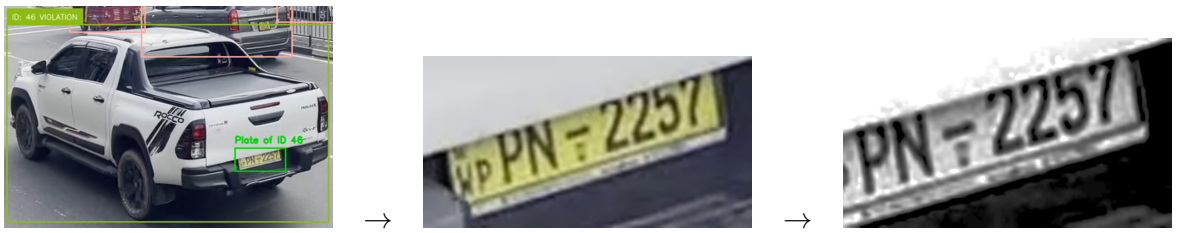
(b) During Occlusion



(c) After Occlusion

Figure 4.4: Snapshots of a Vehicle being Occluded

4.4.5 Number Plate Detection Results



Violated Image → Cropped Image → Processed Image

Figure 4.5: Visual pipeline for license plate extraction: from raw violation frame to preprocessed OCR-ready image.

	A	B	C	D	E
1	vehicle_id	image	predicted_plate	violation_frame	violation_time
2	46	cropped_plates/plate_vid46_frame00571.png	PN-2257	redlight_violation_frames\violation_frame548_2025-07-23_02-46-37_46.jpg	2025-07-23_02-46-37
3	53	cropped_plates/plate_vid53_frame00654.png	KH-0421	redlight_violation_frames\violation_frame615_2025-07-23_02-46-48_53.jpg	2025-07-23_02-46-48
4	54	cropped_plates/plate_vid54_frame00687.png	LK-6516	redlight_violation_frames\violation_frame646_2025-07-23_02-46-52_54.jpg	2025-07-23_02-46-52
5	60	cropped_plates/plate_vid60_frame00799.png	LH-2670	redlight_violation_frames\violation_frame717_2025-07-23_02-47-04_60.jpg	2025-07-23_02-47-04

Figure 4.6: Screenshot of the saved CSV file

Chapter 5

Discussion

The proposed pipeline integrates a number of state-of-the-art models to efficiently detect and handle traffic violation effectively by focusing on real-world robustness, modularity, and automation. It is built around the meticulous balance between accuracy, speed, and scalability, validated by a series of comparative experiments.

5.1 Comparisons

5.1.1 Comparative Evaluation of Re-Identification Models

One of the main decision points of system design was choosing the most suitable vehicle re-identification (ReID) model. Two notable models, FastReID and TransReID, trained with Veri, VehicleID datasets are tested on the footages we recorded.

Though TransReID was slightly less responsive (11 FPS compared to 12 FPS), it captured overall higher re-identification accuracy. Identity swaps were minimized with better continuity of tracking with TransReID, more desirable to applications where violations assignment with precision is more critical. The comparatively minimal loss of inference speed was tolerable for the much greater payback of improved reliability added. As in figure 5.1, in red font it shows the cosine similarity value of the 2 vehicles and TransReID gave better results.

This is a similar trade off to the overall design philosophy of the system only prioritizing accuracy for mission-critical points such as capturing violations and ticket creation while maintaining close to real time speed for the bulk of the pipeline. For use cases where processing power is not abundant (i.e., edge roll-out or budget-conscious deployments), FastReID is always available as a fallback with its smaller load and acceptable sacrifice of accuracy.

There were some limitations nevertheless. Plate recognition would fail every now and then when lighting conditions were quite poor or when plates were displayed with very sharp

(a) Similarity value using FastReID

(b) Similarity value using TransReID

Figure 5.1: Comparison of Cosine Similarity Values using FastReID and TransReID

angles. Real time performance was also periodically impacted by the overall processing load from end to end. Our working Python based prototype outputted 11-12 FPS with the low spec GPU configuration we happened to test with, but real world applications involving legacy CCTV or working around network delay might have smaller throughput. As we remarked when we tested it, achieving real time performance is limited due to performance constraints. Furthermore, with more input streams introduced, the re identification association step may become the processing bottleneck. To take this into account, future deployment strategies should take note of distributed processing (e.g., one GPU for every 2–4 camera streams) and apply frame down sampling where it is feasible to reduce the load on computation. For verification purposes, we captured at 30 FPS with the help of the camera on the smartphone. In the legacy CCTV systems, the frame rate is typically 15 FPS. Nevertheless, we were always able to meet the inference rate of 11 FPS, which is fairly good given the current configuration of the system. It can additionally be tuned to real-time rates on deployed systems with more than 12 GB of VRAM on high-power GPUs.

5.1.2 Superiority Over Single-Camera Systems

Unlike conventional systems operating on single-camera setups, our pipeline is advantageously equipped with cross-camera tracking. It drastically reduces vehicle loss due to occlusion or camera field of view limitations. By leveraging both spatio temporal reasoning and appearance

features, the system judiciously constrains candidate matches to feasible trajectories among time-synchronized cameras.

The use of a Graph Convolutional Network (GCN) to model transitions between overlapping Fields of View (FoVs) is a novel advantage. This eliminates the need for naive exhaustive matching among all cameras, which is computationally expensive and time consuming. GCNs alternatively offer probabilistic weights for angular coverage, distance, and past transitions, facilitating targeted re identification. This not only improves matching accuracy but also scalability.

5.1.3 Strength in Modular Design and Automation

The modular form of the pipeline permits it to process infractions in programmatic, automated manners:

1. YOLOv11s detects traffic elements efficiently with high recall.
2. ByteTrack handles temporal consistency and object-level tracking, preserving identity across frames.
3. The license plate recovery module, enhanced with Zero-DCE, ensures high OCR performance under challenging visibility conditions.
4. Microsoft's TrOCR, when combined with preprocessing techniques, delivers robust OCR across diverse fonts and lighting conditions.

Each module was fine tuned or carefully curated to real world constraints city differences in lighting, partial occlusions, and non standard license plates from Sri Lanka. The combination of heuristic reasoning (e.g., ROI for the stop line/lane/parking for stopping behavior prediction and parking departure prediction) with deep models for perception and for OCR results in one that is efficient as well as interpretable.

5.1.4 Operational Efficiency and System Constraints

Benchmarking on an Intel i7 CPU and RTX 2070 GPU provided a steady throughput of 11 FPS with real-time capability for small to medium sized configurations. Latency is still prone to real world variation depending on networking, storage latency, or camera frame rate change. Flexibility is being provided in the prototype implementation as of now in Python

but is provided with the optimized performance through the utilization of optimized inference runtimes like TensorRT or OpenVINO for production use cases.

Even with proper performance, the system's precision is compromised by some environmental and physical conditions—i.e., harsh weather, night reflections, or camera movement. It is also reduced when plate recognition is at drastically sharp angles and is illegible for re identification when the appearance of the vehicle is highly similar (i.e., fleet cars).

But in contrast to many heuristic or scholar-derived schemes, our pipeline is truly deployable in the real world through the use of common CCTV video without incurring the cost of pricey sensors. This renders it cost effective and deployable to low budget cities or police departments.

5.2 Future Work

To mature Occluwatch from an operational prototype to a fully functional field system, directions for future development are identified. The developments encompass technical and architectural enhancements to user level improvements that would enable real time operation, expanded coverage of violations, and deployability across city infrastructure.

5.2.1 Real Time Field Deployment via Edge Computing

Occluwatch currently operates in an offline mode using recorded CCTV. A step at hand is to integrate the system with live city surveillance streams. In order to reduce latency and enable timely ticketing, we intend to containerize the whole pipeline using Docker and host it on edge servers physically located at or near intersections. The nodes, integrating embedded GPUs (e.g., NVIDIA Jetson or Intel Movidius), will locally run the detection, tracking, and initial OCR modules. Only cropped event end of violation evidence would be uploaded to a centralized backend for ticketing, reporting, and storage.

The system reduces bandwidth consumed in continuous streaming of the video, as well as offering faster response times. It also enables better fault tolerance (e.g., power/network outages) and aids in decentralized scaling with additional cameras across the city.

5.2.2 Expanded Violation Detection Capabilities

While the current system can sense red light, lane, and parking violations, real world enforcement situations require broader coverage. The subsequent phases of development will

be:

1. Speed Violations: Using stereo vision, temporal tracking across camera baselines, or even LiDAR integration, we plan to calculate vehicle velocity accurately in real time.
2. Illegal Turns: By analyzing trajectories within predefined turn zones and cross referencing with traffic signal logic, the system can detect unauthorized right or left turns.
3. Wrong-Way Driving and U turns: Additional logic for vehicle heading estimation can be integrated into the ROI-based violation system.
4. Stop Sign Violations: Incorporating additional object detection logic for stop signs and monitoring vehicle halts in their proximity.

5.2.3 Enhanced Occlusion Handling and Visual Recovery

Occlusion remains an inherent problem with dense traffic or camera blind zones. To improve robustness:

1. Multi View Reconstruction: Using overlapping camera streams, 3D scene reconstruction can be employed to infer vehicle paths even in the occlusion periods.
2. Depth Sensing Integration: Depth sensors (e.g., stereo cameras or ToF sensors) are not required and can aid in better object segmentation as well as occlusion sensitive tracking.
3. Neural Occlusion Completion: Advances on generative models as well as occlusion reasoning (e.g., Gated Graph Neural Networks or trajectory estimation via transformers) can predict object behavior behind obstacles, reducing identity switches or track loss.

5.2.4 User Interaction, Transparency, and Legal Compliance

1. Appeals Workflow, enabling users to challenge violations or submit explanatory information directly.
2. Multilingual Notifications and Localization, tailored for local law enforcement jurisdictions and driver demographics.
3. Accessibility and Privacy Settings, allowing users to manage their data preferences and ensuring compliance with regional data protection regulations (e.g., GDPR equivalent policies).

5.2.5 Scalable Infrastructure

1. Microservice based Backend: Decouple violation processing, OCR, ticketing, and database services into independently scalable modules.

2. Streaming Data Pipelines: Use frameworks like Apache Kafka or Redis Streams for real-time event handling and violation tracking.

Chapter 6

Conclusions

This project extended the end-to-end system for video based automatic detection of traffic violations in dense urban environments with exclusive video input. We built a multi camera tracking pipeline ("Occluwatch") that weaves together fast object detection (YOLOv11s), strong tracking (ByteTrack), and deep re identification (TransReID), all of which handle explicitly occlusions due to high density traffic. The system was effective in identifying and tracking cars between CCTV overlapping views, scanning license plates in night conditions, and properly identifying various infractions, including red light running, unsafe lane change, and illegal parking.

Experimental outcomes confirm that both YOLOv11s detector and ReID models maintain at or above benchmark level expectations with high precision and recall in diverse urban settings. In particular, our camera handoff strategy through ReID based approaches and complemented by spatial temporal filtering drastically eliminated identity switching and tracking loss even at extremely crowded intersections. This dramatically enhances long-term car tracking accuracy across multiple cameras.

Moreover, by integrating these modules based on AI with a responsive notification site, Occluwatch offers real time notifications and complete violation reports, providing a strong proof of concept for smart, multi camera based traffic enforcement. This is a very sound basis for future deployment in smart city infrastructure, as scalable, automated traffic monitoring systems play a key role in road safety as well as efficiency.

Bibliography

[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[2] 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 *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022.

[3] S. He, H. Luo, P. Wang, F. Wang, H. Li, and W. Jiang, "Transreid: Transformer-based object re-identification," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021, pp. 15 013–15 022.

[4] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[5] C. Guo, C. Li, J. Guo, C. C. Loy, J. Hou, S. Kwong, and R. Congrun, "Zero-reference deep curve estimation for low-light image enhancement," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 44, no. 8, pp. 4225–4238, 2021.

[6] M. Li, M. Liu, W. Zhang, W. Guo, E. Chen, and C. Zhang, "A robust multi-camera vehicle tracking algorithm in highway scenarios using deep learning," *Applied Sciences*, vol. 14, no. 16, p. 7071, Aug 2024.

[7] Z. Ouardirhi, S. A. Mahmoudi, and M. Zbakh, "Enhancing object detection in smart video surveillance: A survey of occlusion-handling approaches," *Electronics*, vol. 13, no. 3, p. 541, Jan 2024.

[8] S. He, X. Li, H. Zhang, and L. Zhang, "Transreid: Transformer-based object re-identification," *arXiv preprint arXiv:2102.04378*, 2021. [Online]. Available: https://arxiv.org/abs/2102.04378

[9] X. Liu, "Provid progressive and multi-modal vehicle re-identification for large-scale urban surveillance," https://vehiclereid.github.io/VeRi/, 2018, accessed: 2025-07-27.

[10] L. He, X. Liao, W. Liu, X. Liu, P. Cheng, and T. Mei, "Fastreid: A pytorch toolbox for general instance re-identification," *arXiv preprint arXiv:2006.02631*, 2020.

[11] J. Stark, "Image enhancement using histogram equalization technique," *IEEE Engineering in Medicine and Biology Magazine*, vol. 13, no. 5, pp. 97–103, 1994.

[12] C. Guo, Y. Li, J. Guo, C. C. Loy, J. Hou, S. Kwong, and R. Cong, "Zero-reference deep curve estimation for low-light image enhancement," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1780–1789, 2020.

[13] M. Li, Y. Zhang, L. Cui, N. Yao, D. Chen, S. Wang, Z. Liu, Z. Chen, Y. Wu, F. Huang, and M. Zhou, "Trocr: Transformer-based optical character recognition with pre-trained models," arXiv preprint arXiv:2109.10282, 2021.

[14] A. Reinman, "Nodemailer - send e-mails with node.js," https://nodemailer.com/, accessed: 2025-07-27.

[15] Express.js Contributors, "express-session - node.js session middleware," https://github.com/expressjs/session, accessed: 2025-07-27.

[16] B. Contributors, "bcrypt - a library to hash passwords securely," https://github.com/kelektiv/node.bcrypt.js, accessed: 2025-07-27.

[17] J. Hanson, "Passport.js — simple, unobtrusive authentication for node.js," http://www.passportjs.org/, accessed: 2025-07-27.

[18] D. Hardt, "Oauth 2.0 authorization framework," https://tools.ietf.org/html/rfc6749, 2012.

[19] G. Motdotla, "dotenv - loads environment variables from .env file," https://github.com/motdotla/dotenv, accessed: 2025-07-27.

[20] Y. Zhou *et al.*, "Vehicleid: A large-scale vehicle re-identification benchmark," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2018.