JOIV : Int. J. Inform. Visualization, 8(3-2): IT for Global Goals: Building a Sustainable Tomorrow - November 2024 1515-1527

INTERNATIONAL JOURNAL ON INFORMATICS VISUALIZATION

journal homepage : www.joiv.org/index.php/joiv

Traffic Violation Detection Using Computer Vision Techniques

Chin Sin Ong [a], Tee Connie [b,*], Michael Kah Ong Goh [b]

[a] *Faculty of Information Science and Technology, Multimedia University, Jalan Ayer Keroh Lama, Melaka, Malaysia*
[b] *Multimedia University, Jalan Ayer Keroh Lama, Melaka, Malaysia*
Corresponding author: [*]tee.connie@mmu.edu.my

***Abstract*—The increasing number of road accidents is still a global concern. Traditional approaches to detecting traffic violators on the road, such as radar guns and sensors, are expensive and time-consuming to maintain and install. This often results in inefficient and ineffective detection of traffic violators. This paper proposes a more cost-effective and efficient approach to traffic violation detection utilizing visual data from CCTV footage. Specifically, the method targets two common violations: crossing red lights and overtaking on double lines. In this study, YOLO is integrated for road object detection, providing the detection of vehicles and traffic lights on the road for our system. Then, the Deep SORT tracker tracks detected vehicles, ensuring continuous monitoring over time. An automated lane detection technique is formulated to identify the stopping line/lane for red light violation detection, enabling precise detection of vehicles that cross the stop lane during red light. For overtaking detection, the system detects the double line to serve as the boundary that vehicles should not cross, identifying illegal overtaking. Furthermore, point-line distance calculation is utilized to detect traffic violators by analyzing their tracked trajectories and positions. The proposed solution is evaluated using real-world CCTV footage from online repositories to reflect the real-world scenarios as closely as possible. Experimental results show that the proposed techniques achieve promising detection of real-time traffic violators, which leads to a safer environment for road users.**

***Keywords*— Traffic violation detection; computer vision; YOLO; lane detection; state recognition; point-line distance.**

Manuscript received 11 Apr. 2024; revised 26 Jul. 2024; accepted 20 Oct. 2024. Date of publication 30 Nov. 2024.

I. INTRODUCTION

The traffic surveillance system is crucial for traffic authorities to ensure public safety and maintain an efficient transportation network. It is being utilized worldwide to manage traffic flow, improve road safety, and detect traffic violators. Unfortunately, road accidents remain one of the top 5 causes of death in Malaysia, occupying 3.2% of the principal causes of death according to the statistics published by the Department of Statistics Malaysia (DOSM) [1]Most accidents are undoubtedly caused by reckless human driving behaviors, such as running a red light and illegal overtaking.

The high frequency of accidents from humans' reckless driving has caused many people, including innocents, to lose one's life and be injured. Most accidents are caused by these reckless drivers who break the traffic rules, bringing the safety of road users in danger [2]. This issue should be addressed and given due attention. Implementing an AI-based traffic surveillance system offers a potential solution to detect these violators and ensure they face appropriate consequences. With the high frequency and increasing volume of accidents over the years, it has become increasingly difficult for traffic authorities to manage and control the traffic effectively [3]. Consequently, road users face various challenges, such as congestion and safety issues. Integrating AI technology into traffic surveillance systems has become essential for traffic authorities to manage traffic flow more efficiently.

Currently, most traffic violation detection systems still use the traditional approach with radar guns and sensors, which can be costly to maintain and install. Moreover, these sensors often demand more frequent maintenance than CCTV systems. Sometimes, even manual police patrols are required to catch traffic offenders, a method that can be both costly and inefficient. Therefore, an adaptable system for integration into CCTV infrastructure can significantly reduce the overall cost and labor usage.

This paper proposes a visual-based system that can be integrated into a CCTV system to control traffic flow and detect traffic violators. The proposed system is designed to detect traffic violators, including illegal overtaking and red light running, which can help lessen reckless drivers and reduce the road accident rate. Therefore, implementing the

system could help smooth the traffic flow as drivers are more likely to adhere to traffic regulations, leading to fewer disruptions and delays on the road network.

The system is designed to process real-time video footage received from CCTV cameras installed at selected locations. Given its integration with CCTV technology, the system's coverage extends to any area where the proposed model is deployed. With its versatility, the system can effectively detect traffic violations across diverse settings, including highways, roads, and intersections. The system combines different algorithms that are responsible for different tasks. YOLO is used to detect road objects, including vehicles and traffic lights. Nonetheless, lane detection is designed to be automated using various image processing techniques, such as Canny edge detection and Hough transform, to outline the actual lane in the video frames. Point-line distance is then calculated according to the vehicles and the identified line for traffic violation detection.

Research on traffic surveillance systems has been gaining more attention in recent years. Due to the rapid advancement of AI technology, there has been an increasing interest in using machine learning and deep learning techniques to monitor and manage traffic flow on roads, highways, and intersections. Therefore, this paper presents a comprehensive review of the existing research literature in the field of traffic surveillance systems. The literature review mainly focuses on reviewing the conventional and deep learning methods proposed by researchers for traffic surveillance detection in recent years. There have been various methods utilized by researchers in the field of traffic surveillance, such as calculating the deceleration rate of vehicles when they come close to the traffic stop lane using a Support Vector Machine (SVM) and Random Forest (RF) [4], [5], Vehicular ad hoc Network (VANET) [6]. Particle Swarm Optimization (PSO) [7], Convolutional Neural Network (CNN) [8], [9] and other algorithms for detecting red light running and other common traffic violations on the road, with their corresponding advantages and drawbacks.

A. Conventional Methods

In the study by Katanyoo et al. [10], the authors used MSD to identify red-light runners and lane-altering violations before traffic lights. HSV color space is used instead of RGB, which is more perfectly suited for computer vision. In addition, HSV color space is also used to track vehicles, which works by deducting the background frame's average hue value from the current frame's average hue value. The difference in hue value is then used to identify the possible presence of vehicles. This has reduced the frequency of mistakenly detecting shadows as vehicles. Vehicle trajectories are computed using MSD. The lower difference in MSD value indicates a higher likelihood that the vehicle in different frames is similar. Lane-altering violations are detected by analyzing the direction changes in vehicle trajectories. On the other hand, when a vehicle crosses the red light's reference line, it will be recognized as a red-light runner. The findings show that traffic light signal detection was 100% accurate. However, the proposed system performs poorly on lane-altering detection, with only 28 violations successfully detected. Vehicle violation detection got an average accuracy of 79%, whereas small, medium, and large vehicles have an accuracy of 78%, 85%, and 74%, respectively.

Nonetheless, Mochamad et al. [11] researched to detect traffic light runners at road intersections using BLOB and MST methods. The road intersections are given reference lines for markers in the CCTV to detect traffic offenses. Thus, vehicles will be marked as running the red lights if they cross the line when the traffic lights are red. The dataset was pre-processed with a few stages from thresholding, contour, and convex hull to Binary Large Object (BLOB). The MST algorithm was then applied in vehicle tracking to detect traffic violations. The authors utilized the MST algorithm to track and predict the position of vehicles over time by examining the changes in their position in consecutive frames. The pro of the proposed system is that cars are more likely to be classified correctly because they are only classified and tracked after passing the reference line. This ensures that all the vehicles have similar physical dimensions captured from the same perspective and orientation. In contrast, the performance of the proposed system could be more stable due to the inability to deal perfectly with a dynamic environment. For instance, congestion would affect the mean shifting performance, and the system has the risk of wrongly identifying the lights reflected on the road as a car. The results correctly classify cars and motorcycles are 71% and 79%, respectively. However, only 58% accuracy in detecting traffic light runners was achieved in this research using the BLOB and MST algorithms.

There exists a recent work by Akhilalakshmi et al. [12] proposing a traffic violation detection system capable of detecting vehicles swerving and vehicles blocking pedestrian lanes. Haar Training tool, a software tool used to train classifiers for object detection in OpenCV, was utilized in vehicle detection. Genetic Algorithm (GA) was used to determine whether a vehicle violated traffic rules. The algorithm consists of multiple steps, starting with initializing the population by generating values for a and b, referred to as the chromosomes. Best chromosomes are chosen for the fitness function, and chromosomes that produce low fitness values are eliminated. The trimmest fit members of the population are then replaced with new chromosomes whose gene values are changed. This process continues until the algorithm finds the optimal solution. Tesseract software is used to detect and capture the license plate number of vehicles that were found to have violated the traffic rules. In this paper, the authors found that the proposed algorithm is more efficient in finding an optimal solution than the standard algorithms, as it uses an ample search space. However, computational resources are expensive and time-consuming using an ample search space. The result shows that the fitness value increases over generations, which indicates that the algorithm is heading towards more optimal solutions over generations. Overall, the proposed system by the authors shows promising results for detecting traffic violations and has the potential to be further improved through future research.

B. Deep Learning Methods

With the increasing popularity and adoption of deep learning techniques, the study by Joel et al. [13] shows an

overall accuracy of 100% in traffic light runner detection and 92.1% in speeding detection. The authors utilized SSD, which uses CNN as its base architecture in object detection. The authors performed feature extraction on the dataset beforehand to improve system performance. Three feature extractors, including MobileNet, Inception V2, and ResNet 50, are used with SSD and compared with each other in the feature extraction phase. The NMS process has been done to remove duplicate bounding boxes on each vehicle. A stop line has been set up to detect if a vehicle runs a red light when the traffic light is indicating red. Euclidean Distance has been used in the system to estimate the vehicle's speed. The Euclidean Distance equation is computed to measure the displacement of vehicles in pixels. Then, a unit conversion mathematical equation is applied to convert the speed to the unit of km/h. Out of the three available feature extractors, MobileNet was chosen in the proposed system due to its high accuracy of 98.90% and processing speed of 0.08 seconds per frame, which is the fastest among the three. Overall, the proposed system can detect all traffic light runners accurately. However, the proposed system requires manual traffic light timer insertion to determine the current traffic light's state, which is inefficient regarding real-time traffic monitoring.

In the same year, Srinivas et al. [14] developed a traffic violation detection system using the Region-based Convolutional Neural Network (R-CNN) method to identify vehicles running red lights. The proposed system could identify plate numbers and estimate the speed of any vehicle that violates the traffic rules, enabling relevant authorities to take any required action. Support Vector Machine (SVM) and CNN have been used to extract objects and features from images in object detection. Predefined lines are drawn on the screen to determine whether a vehicle has crossed the line when the traffic lights turn red, which results in violating the traffic rules. Optical Character Recognition (OCR) recognizes the vehicle's number plate. The experiments show the system's capability to detect red light runners and capture the violating vehicles. It takes 6.55 seconds to execute the whole process. The proposed system allows real-time traffic monitoring, which provides efficient traffic monitoring management and reduces the workload of traffic authorities. However, the system requires manual insertion of a traffic stop line, and it is highly dependent on the video quality for accurate license plate recognition.

Besides the mentioned works, Ilker et al. [15] propose a contextual anomaly detection method to detect various traffic anomalies, including point and contextual anomalies. The surveillance system was implemented with a Contextual Anomaly Detection Network (CADNet) on an Unmanned Aerial Vehicle (UAV). Instead of letting the machine learn about the anomalies, standard environmental patterns were fed into the machine. Besides the differences in training using standard traffic patterns, the proposed method uses UAV instead of CCTV, which is much more costly. "CADNet consists of an encoder, decoder, and a context subnetwork" [15]. The encoder is responsible for processing incoming data, and the context subnetwork gathers more contextual attributes from the environment and produces different encoded representations to capture essential information from the input. The decoder then creates a reconstructed sample from concatenated encoded representations. The method uses reconstruction error to detect anomalies through a reconstruction-based approach. As the reconstruction error increases, the likelihood of anomaly increases. Synthetic-generated anomalies were inserted into the dataset to evaluate the effectiveness of CADNet in point anomaly detection. The authors reported that their proposed method has higher accuracy than others. The accuracy of detecting point anomalies was 91.2%, and the accuracy of detecting contextual anomalies was 86.6%. The proposed model can detect various traffic anomalies, from point anomalies to contextual anomalies, which are challenging with relatively high accuracy. Still, unlike other existing models, the proposed model uses UAV, which uses CCTV and thus result in higher overall costs, including maintenance, equipment, and deployment cost. A summary of state-of-the-art methods is provided in Table 1.

TABLE I
A SUMMARY OF STATE-OF-THE-ART METHODS

Author	Method	Result	Pros	Cons
Katanyoo et al. [10]	MSD	Traffic light detection: 100% Violation detection: 79%	Uses HSV in vehicle detection which reduces frequency of mistakenly detect shadow as vehicle	Not well-performed in lane altering violation
Mochamad et al. [11]	MST	Accuracy: 58%	Vehicle more likely to be classified correctly due to consistent position and angle	Hard to detect vehicles during traffic jam, morning lights on road can be mistakenly recognised as vehicle
Akhilalakshmi et al. [12]	GA	0.056 fitness value from 45 to 100 generation	Provide optimal solution	Large search space means more computational cost, and time-consuming
Joel et al. [13]	SSD	Red light violation detection: 100% Speeding detection: 92.1%	Able to detect all traffic light runner accurately	Manual insertion of traffic light timer which is inefficient in real-time traffic monitoring
Srinivas et al. [14]	R-CNN	Processing time: 6.55s	Proposed system allows real-time traffic monitoring	Requires manual insertion of traffic stop line, and highly dependent on the video quality for accurate license plate recognition
Ilker et al. [15]	CADNet	point anomaly: 91.2% Contextual anomaly: 86.6%	Ability to detect various traffic anomalies	Model uses UAV, overall cost of UAV is expensive

When comparing the disparities between the proposed and existing systems, the proposed system enhances the existing works due to its lane detection and traffic light state automation without manual intervention. Furthermore, the proposed system reduces overall costs and improves efficiency by integrating CCTV instead of a UAV system.

II. Materials and Method

The proposed system can detect two traffic violation scenarios: red light running and illegal overtaking. When a vehicle crosses the detected stop line when the traffic light is red, it will be immediately identified as a red-light runner. On the other hand, vehicles that overtake another vehicle illegally will be determined as illegal overtaking. This can be determined when a vehicle overtakes in the occurrence of a double line where overtaking is not allowed. Fig. 1 shows an overview of the proposed system. In general, when the proposed system receives an input video, road objects, including vehicles and traffic lights in each frame of the video, are detected using YOLO [16]. Each detected vehicle will be given with its unique identifier and tracked using Deep SORT [17] technique. Next, the traffic stop lane and overtaking double line are identified using Canny Edge Detection [18] and Hough Transform [19] techniques.

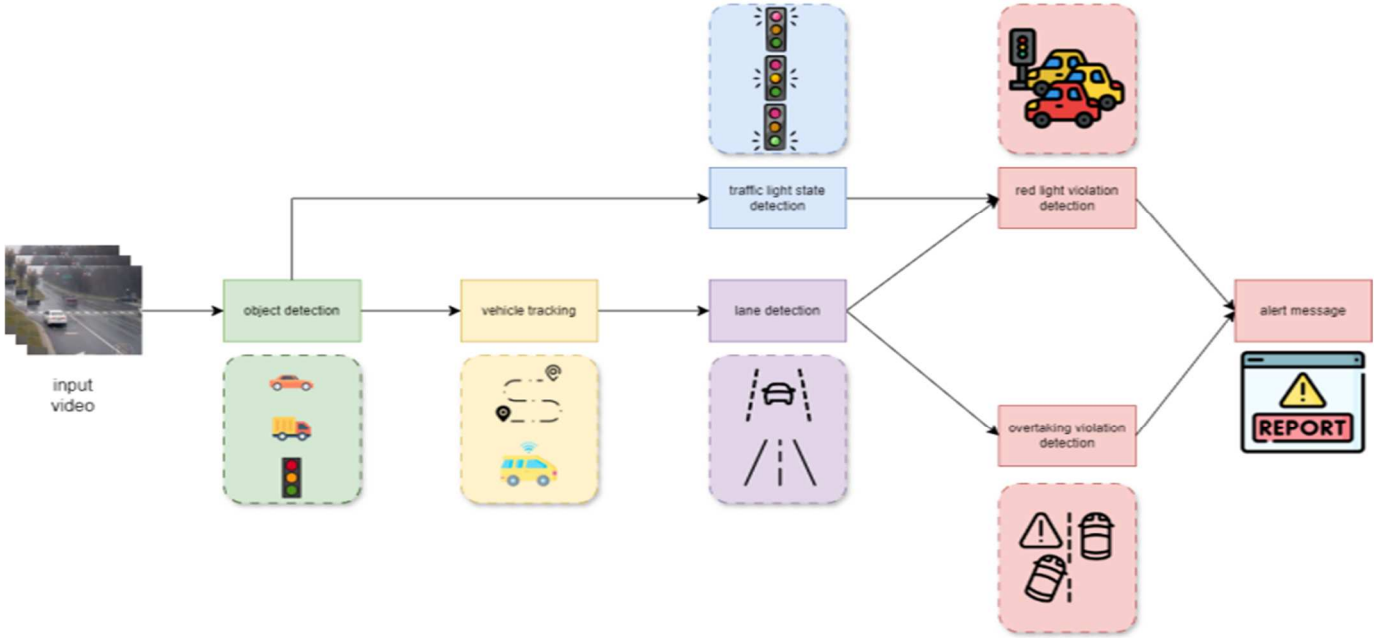


Fig. 1 Flowchart of the proposed traffic violation detection system

The traffic stop lane indicates the line that is not allowed to be crossed when the traffic light is in a red state, and the overtaking double line indicates the area not allowed to be crossed by vehicles. In converse to the detected vehicles, detected traffic lights in each frame will be processed for current traffic light state identification to determine whether the current state is red. After these procedures, the proposed system detects any traffic violations occurring, including traffic light violations and illegal overtaking violations. There are two conditions to determine whether a vehicle has violated the traffic light rule: the current traffic light state is red, and the vehicle crosses the identified lane. If both conditions are fulfilled, that vehicle will be detected as a traffic violator and thus send an alert message to the traffic authorities. On the other hand, when a vehicle crosses the identified overtaking double line, that vehicle will be classified as a traffic violator and send an alert message to the traffic authorities.

A. Object Detection

YOLO is a popular and widely used algorithm in real-time object detection. It is given the name You Only Look Once because of its ability to process images quickly and detect real-time objects with high accuracy. The algorithm applies CNN [20] that divides the image into small pieces and predicts the class probabilities by processing those individual pieces. When an object is detected, it will be assigned a bounding box with its predicted class and the likelihood of it belonging to that class. These objects are classified by training the model with a training set annotated with their corresponding predefined classes. The trained model will then predict the occurrence of objects on an image based on the confidence score according to the predefined classes. The detected object will be assigned a bounding box with the highest confidence score object class. When an image or video is fed into YOLO, it divides the image into equally small grids. Each of these small grids is responsible for predicting objects within its boundaries. Each grid cell has its respective vector, as shown in Equation 1.

$$C_{1,1} = (P_c, B_x, B_y, B_w, B_h, C_1, C_2) \quad (1)$$

Pc represents the probability of the presence of an object within the grid cell. *Bx* and *By* represent the bounding box's location (center coordinates). At the same time, *Bw* and *Bh* represent the object's size with its respective width and height. Finally, *C1* and *C2* represent the probability of the object belonging to a particular class. The number of *C* is dependent on the total number of predefined classes. Non-

Maxima Suppression (NMS) [21] is applied in post-processing to overcome the common problem of redundant and overlap bounding boxes in object detection. This will allow YOLO to remove high overlapping bounding boxes and keep the bounding box with the highest object score.

Object detection is the first yet important part of accurately detecting the objects on the road in the proposed system. As YOLO has a good performance, high speed, and high accuracy in real-time object detection tasks and its capability of detecting multiple objects within a single image, it is utilized in this paper to detect road objects, including vehicles and traffic lights. It is crucial to accurately detect the objects on the road as accurate object detection highly improves the performance and accuracy of red-light violation detection. Therefore, three different variations of YOLO algorithms, including YOLOv7 [22] model, YOLOv8n [23] model, and YOLOv8x [24] The model has been evaluated using the collected testing set to find the optimal solution for road object detection.

Input frames are being processed for object detection within frames using YOLO. Frames are converted to PyTorch tensor to normalize the pixel values to the range of [0,1]. Inference is made using a forward pass through the YOLO algorithm to produce predictions. In the post-process, NMS filters out redundant bounding boxes with low confidence scores. It then annotates the original frames with bounding boxes, labels, and other prediction information. The bounding boxes of the detected vehicles are extracted within the frames, and their corresponding confidence scores and labels are passed for tracking. It associates the detections with existing tracks and predicts new tracking for unassociated detections [25].

B. Vehicle Tracking

Deep SORT is a multiple-object tracker algorithm capable of tracking multiple objects in a single frame. When comparing to the SORT algorithm, it is more efficient, reliable, and accurate as it is designed as an extension of the SORT algorithm to address the limitations of SORT. One of the advantages of Deep SORT algorithm is its capability at handling occlusions using deep learning-based appearance features. In addition, it is more robust in tracking objects that changes their appearance which may be due to the occurrence of rotation or scaling.

In Deep SORT, two main algorithms, including the Kalman Filter [26] and Hungarian Algorithm [27] are utilized for object tracking. Kalman Filter is integrated into Deep SORT to estimate the state of the detected objects in subsequent frames. Thus, it helps predict the detected objects' future position based on their current state and motion, which is essential in object tracking. On the other hand, the Hungarian Algorithm is incorporated in Deep SORT to find the best match between the predicted positions by the Kalman Filter and the newly detected objects across frames using the cost matrix. The Hungarian Algorithm aims to determine whether a predicted position belongs to a specific detected object.

Therefore, Deep SORT, with its reliable and efficient ability to track multiple objects, is utilized in this paper to track multiple vehicles on the road in real-time. This allows the proposed model to keep track of the trajectories and motion of each vehicle, thus further processing for red light violation detection. Fig. 2 shows the architecture of Deep SORT receiving the input of detected objects using YOLOv8 and internally processing the detected objects for real-time object tracking.

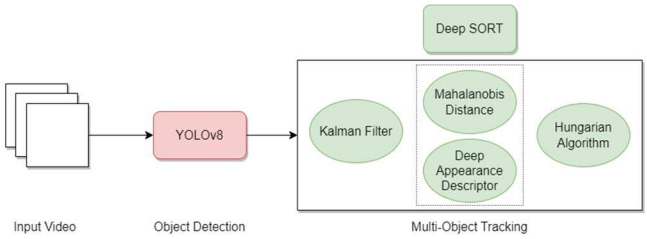


Fig. 2 Deep SORT Architecture [28]

C. Lane Detection

Canny Edge Detection is a well-known multi-state edge detection algorithm. In canny edge detection, the gaussian filter is first applied to the image to smoothen it by blurring it thus resulting in noise reduction. Then, the blurred image is passed for filtering with horizontal Sobel kernel and vertical Sobel kernel to get the first derivative in the horizontal direction *Gx* and the vertical direction *Gy*. By using the *Gx,Gy*, gradient magnitude of each pixel indicating the intensity of the edges as shown in Equation 2 and the gradient direction of each pixel indicating the orientation of the edges as shown in Equation 3 can be computed. Equation 4 and Equation 5 show the horizontal and vertical Sobel kernel [29].

$$G = \sqrt{Gx^2 + Gy^2} \tag{2}$$

$$\theta = arctan\left(\frac{Gy}{Gx}\right) \tag{3}$$

$$\begin{matrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{matrix} \tag{4}$$

$$\begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix} \tag{5}$$

After gradient calculation, the recognized edges may appear to have different thicknesses. Therefore, Non-Maxima Suppression (NMS) is performed to thin the thick edges. It works by suppressing all the gradient values to zero except for the remaining local maxima, representing the presence of an edge in that pixel. The double threshold is applied after NMS to determine potential edges. In double thresholding, a high and low threshold is set to determine the strong and weak pixels. Gradients greater than the high threshold are known as the firm edges that will contribute to the final edge. In contrast, gradients between the high and low thresholds may or may not be included in the final edge, depending on their connectivity. Lastly, edge tracking by hysteresis is performed to turn the weak edges into firm edges if they are connected. Otherwise, they are suppressed.

Fig. 3 shows an example image of the output edges detected after applying canny edge detection.

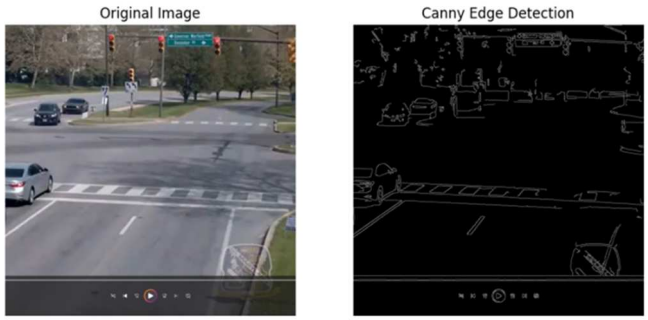


Fig. 3 A sample of Canny Edge Detection result

Hough Transform is a feature extraction technique to detect lines, circles, or other shapes and curves. Its goal is to find imperfect (broken or distorted) instances of objects within similar shapes by voting. After an image edge has been detected using a technique like canny edge detection, it represents the line using Equation 6. r represents the rho which is the shortest distance from the line to the origin (0,0) in the image. θ represents the theta which is the angle between x-axis and the line drawn from the origin to this closest point whereas x, y are the coordinates of the points on the line. Then, voting is performed to all possible lines that could pass through the edge point in the image represented by a pair of r,θ values. Lastly, the cells with the most votes will be chosen for drawing a complete line [30]. Fig. 4 shows the result of Hough Transform after performing Canny Edge Detection.

$$r = x \cos\theta + y \sin\theta \tag{6}$$

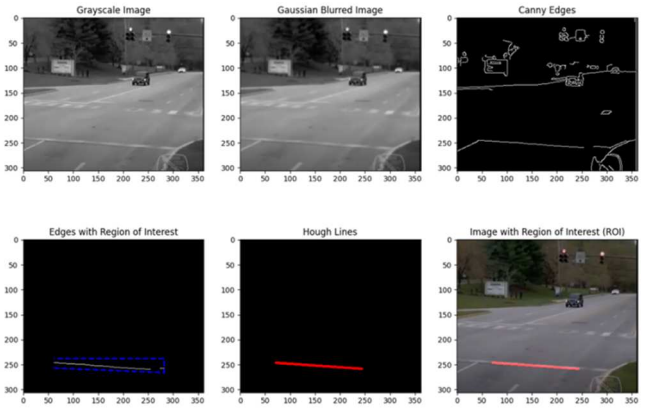


Fig. 4 Hough Transform

In red light violation and overtaking violation detection, it is crucial to accurately determine the lane on the road. Therefore, Canny Edge Detection and Hough Transform have been applied in lane detection to detect the traffic stop line to identify red-light violators and the double line to identify illegal overtaking violators on the road.

D. Traffic Light State Detection

YOLO is trained by feeding the model many traffic light images, which are used to detect traffic lights in each video frame. When the recognized object is a traffic light, it will pass the bounding box as a parameter to identify the current traffic light states. Traffic lights with their determined states will be shown in the video after the proposed system recognizes the current color state. Since the only important characteristic in our scenario is to determine when the traffic light is in the red state for red light violator detection, the proposed system is designed to only categorize whether the current traffic light state is red or not red. The proposed system will show "Stop" on the traffic light when the current traffic light state is red, whereas the system will show "Go" in the video indicating that the current traffic light state is either yellow or green, but not red. This allows simplicity in indicating the traffic light state instead of indicating different traffic color state (red, yellow, green).

As shown in Fig. 5, the dimension of the bounding box is first segmented into three regions representing each region where a different color state (red, yellow, green) is expected to appear. Each segmented region is converted to HSV color space to determine its color coding. With the predetermined red mask, the red pixels are counted in the first segmented region and if there is an existence of red pixel, the traffic light state will be determined as red otherwise not red. By segmenting the traffic light into three equal regions, it reduced the error of traffic light state identification as some of the border of the traffic lights is painted with yellow color which could mislead the proposed system in detecting the correct traffic light state. Algorithm 1 shows the process of the proposed system in identifying the current traffic light state.

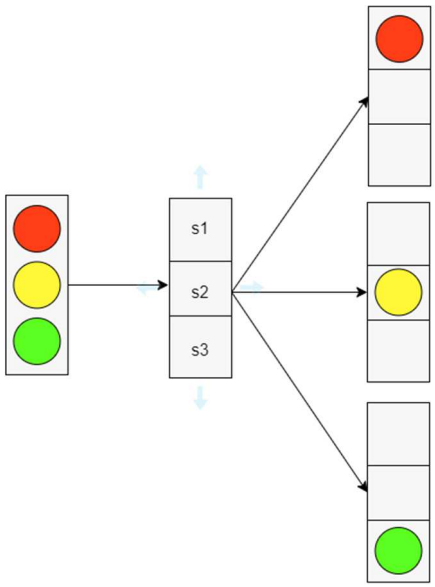


Fig. 5 Traffic Light State Diagram

Algorithm 1 Pseudocode Procedure for Traffic Light State Detection

```
imgheight, imgwidth, _ = img.shape
height = imgheight / 3
convert each square to HSV colour space
count red pixels in first segmented square
if red > 0 then
    return "stop"
else
    return "go"
end if
```

E. Red Light Violation Detection

Red light violations are a significant traffic safety concern that should be addressed to reduce the accident rate on the road [31]. This can be achieved by detecting the position and movement of vehicles in relation to the traffic signals. Therefore, the proposed system identifies red light violators by tracking the vehicles' movement and determining whether a vehicle has crossed the traffic stop line when the traffic signal is in the red state.

In real-time, there may be more than one traffic light from different pathways captured by the CCTV camera. This may yield to the system recognizing traffic lights from other pathways thus affecting the traffic light violation detection accuracy. To address this issue, a constraint is set to process only the detected traffic light with the highest confidence score for traffic light state identification.

Each of the detected vehicles is tracked using Deep SORT. Each vehicle has their center point calculated on the bottom edge of their corresponding bounding boxes representing the vehicle's position for movement and trajectory tracking. If the traffic light is in "Stop" state (red light), it further checks whether the vehicle's center point has crossed the defined stop line detected earlier at intersection. Any vehicle that is found to cross the defined stop line when the traffic light state is red will be identified as red-light violator with a red bounding box on that particular vehicle displayed in the video.

Point-line distance is used with its formula as shown in Equation 7 to determine whether a vehicle has crossed the stop line when the traffic light is red. *x0, y0* represents the center point of the vehicle, while *x1, y1, x2, y2* represents the coordinates of the intersection stop line. In red light violation detection, a distance threshold of 1 is set. A vehicle with its computed point-line distance less than 1 will be identified as crossing the stop line. The traffic violation detection process is shown in Algorithm 2.

$$d = \frac{\left|(y_2 - y_1)x_0 - (x_2 - x_1)y_0 + x_2y_1 - y_2x_1\right|}{\sqrt{(y_2 - y_1)^2 + (x_2 - x_1)^2}} \quad (7)$$

Algorithm 2 Pseudocode Procedure for Traffic Violation Detection

```
function point_line_distance(point,line)
    x0, y0 = point
    x1, y1 = line[0]
    x2, y2 = line[1]
    numerator = |(y2 - y1)x0 - (x2 - x1)y0 + x2 x y1 - y2 x x1|
    denominator = sqrt[(y2 - y1)^2 + (x2 - x1)^2]
    distance = numerator / denominator
        return distance
end function

function violation(center,line,threshold=1)
    distance = point_line_distance(center,line)
    if distance < threshold then
        return True
    else
        return False
    end if
end function

for Each frame in video do
    for each detected vehicle in frame do
        center = (centerx, centery)
        line = [(startx, starty), (endx, endy)]
        result = violation(center,line)
        if True then
            Mark vehicle with red bounding box AND display "red light violation" text
        end if
    end for
end for
```

F. Overtaking Violation Detection

Any illegal overtaking violation is known as a vehicle overtake under the existence of double line. Therefore, overtaking is identified by determining whether a vehicle has crossed the detected double line where overtaking is not allowed. The center point of vehicles is first determined by calculating their corresponding bounding boxes bottom edge center point. The center point is then utilized to determine whether it has intersected the detected double line. Fig. 6 illustrates the detection of overtaking violation by the proposed system. When a vehicle is detected using YOLO, the assigned bounding box with its corresponding coordinates *x1, x2, y1, y2* will be used for the calculation of the center point on the bottom edge of the bounding box using Equation 8. If the center point crossed the detected lane, that particular vehicle will be immediately recognized as traffic violator thus generating an alert message of overtaking detected on that vehicle with a red bounding box. The designed logic uses a point-line distance formula as stated in Equation 7 to check whether the vehicle has crossed the double line. The distance threshold in overtaking detection is set as 0.5. By using the point-line distance calculation as shown in Equation 7, the proposed system is able to identify whether a vehicle has crossed the line causing overtaking to appear. The distance threshold of 0.5 units is used due to the overtaking events typically involving closer proximity to the double line. With a smaller distance threshold, it allows a more precise detection of the overtaking events.

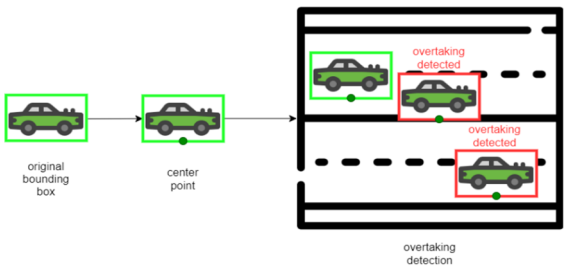


Fig. 6 Overtaking Violation Detection

$$center = \frac{x_2 + x_1}{2}, \frac{y_2 + y_2}{2} \quad (8)$$

III. Results and Discussion

This section presents a comprehensive evaluation of the experimental studies conducted to assess the performance of various object detection models in Subsection A, traffic light state recognition in Subsection B, red-light violation detection in Subsection C, and overtaking violation detection in Subsection D. Experimental results of each proposed model component are presented in this section. The experiments were conducted using real-life CCTV footage collected from online sources to allow the experiments to reflect the real-world scenarios as closely as possible.

The proposed system primarily receives live traffic video feeds captured by CCTV cameras deployed at strategic locations along the roadway. The camera should be placed on a higher level with a top-down view, giving it a clear perspective to identify vehicles, lanes, and the traffic light state. The captured video feeds serve as the data source for the system's detection of traffic violators on the road.

CCTV camera with specific criteria is compulsory for the traffic surveillance set up to capture the live traffic video feeds:

- Internet Protocol (IP) camera,
- RGB CMOS sensor,
- Waterproof,
- High resolution,
- Wide-angle lenses.

Based on these requirements, a few cameras have been selected for consideration:

- Axis Communications Q1786-LE,
- Hanwha Vision PNO-A9081RLP,
- Honeywell HC30WB5R2.

All experiments were conducted on Kaggle with:

- GPU: NVIDIA T4 x2, having 2 x 2560 CUDA cores,
- CPU: Intel(R) Xeon(R) CPU @ 2.20GHz, with 4 cores,
- RAM: 32 GB,
- Disk: 20 GB.

A. Object Detection

This section presents the evaluated performance of different variations of object detection models, including the YOLOv7 model, YOLOv8n model, and YOLOv8x model. The evaluation is conducted using a confusion matrix and accuracy as shown in Fig. 7. The explanation of each matrix is stated below:

- True Positive (TP): Model correctly identifies the presence of a condition,
- True Negative (TN): Model correctly identifies the absence of a condition,
- False Positive (FP): Model predicts a condition to be present, but it is absent, also known as Type I Error,
- False Negative (FN): Model predicts a condition to be absent, but it is present, also known as Type II Error.

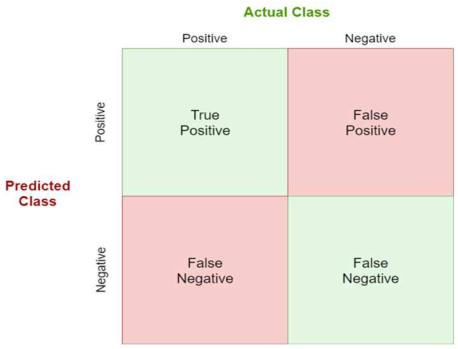


Fig. 7 Confusion Matrix

Several performance metrics can be derived from the confusion matrix, allowing a deeper understanding of the model's performance. This includes precision, recall, F1 score, and accuracy [32-33]. The explanation and equation of each metric are stated below:

- Precision: Measures the proportion of positive class correctly identified by the model,
- Recall: Measures the proportion of actual positive identified correctly by the model,
- F1 Score: Measures the harmonic mean of precision and recall,
- Accuracy: Measures the overall correctness of the model.

$$Precision = \frac{TP}{TP + FP} \tag{9}$$

$$Recall = \frac{TP}{TP + FN} \tag{10}$$

$$F1\ Score = \frac{2\ (Precision * Recall)}{Precision + Recall} \tag{11}$$

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{12}$$

To evaluate the performance of each model, a confusion matrix has been utilized to assess the performance of each model variation in detecting traffic lights and vehicles. In addition, the accuracy of each object detection variation is computed using Equation 11 for comparison purposes and to determine the optimal solution for traffic surveillance tasks. The traffic light and the overall accuracy are computed for each model. To ensure a fair comparison between different models, 50 images obtained from the testing set have been used to evaluate the models by computing their confusion matrix. Each model variation, along with its confusion matrix, prediction result, and accuracy, is presented in this section.

1) YOLOv7 Model: Table 2 shows the confusion matrix of YOLOv7 model. From the confusion matrix, the computed *Accuracy TL* of the YOLOv7 model is 86.56%, whereas the computed *Accuracy OVERALL* is 88.83%.

TABLE II
CONFUSION MATRIX OF YOLOV7

		Actual class		
		Traffic Light	**Car**	**Background**
Predicted class	**Traffic Light**	161	0	10
	Car	0	173	8
	Background	15	9	0

2) YOLOv8n Model: Table 3 shows the confusion matrix of the trained model using YOLOv8n with 100 epochs. From the confusion matrix, the computed Accuracy TL of the YOLOv8n model is 58.56%, which indicates that it does not generalize well in detecting traffic lights on the road.

TABLE III
CONFUSION MATRIX OF YOLOV8N WITH 100 EPOCHS

		Actual class	
		Traffic Light	**Background**
Predicted class	**Traffic Light**	106	5
	Background	70	0

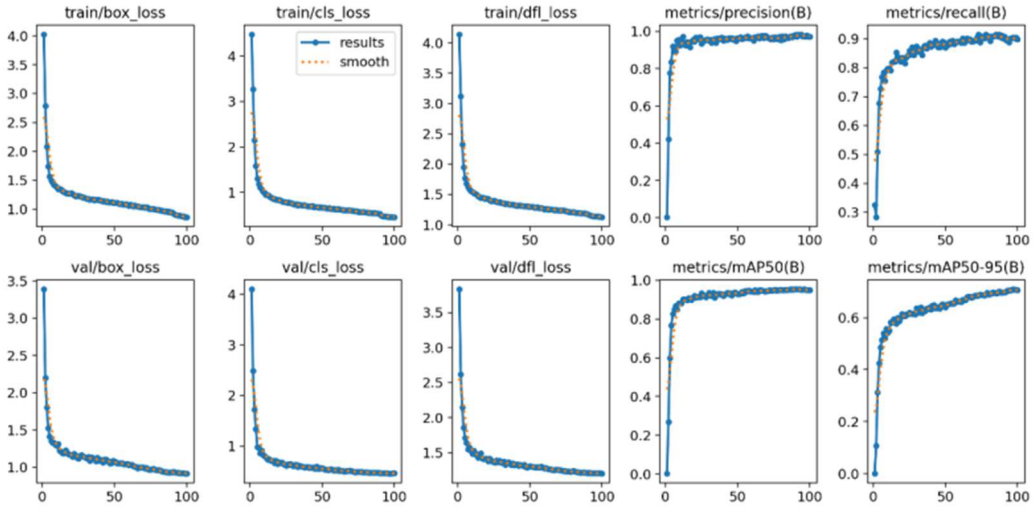


Fig. 8 Loss and Metrics Chart of YOLOv8n with 100 Epochs

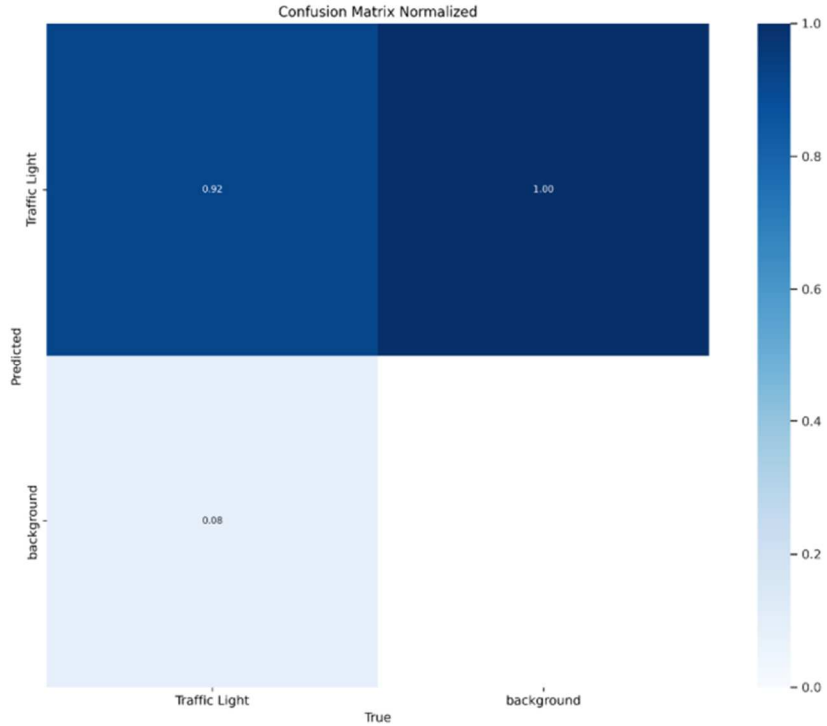


Fig. 9 Confusion Matrix of YOLOv8n with 100 Epochs

When observing the loss and metrics chart and confusion matrix of the trained model using YOLOv8n in Fig. 8, the model is learning well without any signs of overfitting. The loss curve decreases rapidly, indicating that the model is learning effectively. In addition, the precision, recall, and mAP values are quickly increasing, reflecting that the model is getting good predictive performance. The confusion matrix of the YOLOv8n model in Fig. 9 also shows that the model can correctly predict the traffic light class 92% of the time.

However, the model validation on the 50 images obtained from the frames of the traffic videos shows that the model accuracy was reduced to 58.56%. This may be due to the lack of variations in the traffic light dataset, which caused the trained model not to generalize well, especially when the video resolution is low and when it is in a challenging environment, such as at nighttime when the lighting environment is dim.

3) YOLOv8x Model: Table 4 shows the confusion matrix of the YOLOv8x model. Based on the confusion matrix, the YOLOv8x model achieves an *Accuracy TL* of 85.64%. Conversely, the YOLOv8x model achieves an *Accuracy OVERALL* of 91.28%.

TABLE IV
CONFUSION MATRIX OF YOLOV8X

		Actual class		
		Traffic Light	**Car**	**Background**
Predicted class	**Traffic Light**	155	0	5
	Car	0	180	4
	Background	21	2	0

When comparing the accuracy of traffic light detection (*Accuracy TL*) and the overall accuracy (*Accuracy OVERALL*), YOLOv8x gives the best overall accuracy of 91.28%. Even though YOLOv7 has a higher traffic light accuracy than YOLOv8x, the difference in their accuracy is less than 2%. Therefore, YOLOv8x was chosen for our object detection task in detecting objects on roads.

B. Traffic Light State

Several video footages from the real world are fed into the model as a testing set to evaluate how well the proposed model can identify the traffic light and their state. The experimental result of traffic light detection is listed in Table 5, and the experimental result of traffic light state detection is listed in Table 6. From the result, traffic light detection in the morning can achieve a success rate of 100%. This indicates that the model can detect the traffic light as a "traffic light" without much problem in the morning. However, traffic light detection at night has a lower detection accuracy of 84.62% success rate, mainly due to the lighting issues. The model has difficulty recognizing the traffic lights at nighttime, especially when the video has a lower resolution. Additionally, experiments have been done to evaluate the model's performance in accurately identifying the current state of traffic lights. The results obtained in the morning and night-time are 100% and 84.62%, respectively. This indicates that the model can perform well in identifying the current state of traffic lights in the morning but has difficulty determining the red-light state at night.

TABLE V
EXPERIMENTAL RESULT OF TRAFFIC LIGHT DETECTION

Times of Day	**Total Input**	**Detected**	**Success Rate**
Morning	25	25	100%
Night	26	22	84.62%

TABLE VI
EXPERIMENTAL RESULT OF TRAFFIC LIGHT STATE

Times of Day	**Total Input**	**Detected**	**Success Rate**
Morning	25	25	100%
Night	13	11	84.62%

Based on the experimental results, the proposed model has difficulty detecting traffic lights at night due to the dim environment, which causes the traffic light to be partially absent. This has caused the proposed model not to be able to recognize it as a traffic light due to the challenging lighting environment. Fig. 10 shows an example of undetected traffic lights due to the dim lighting, resulting in vague images.

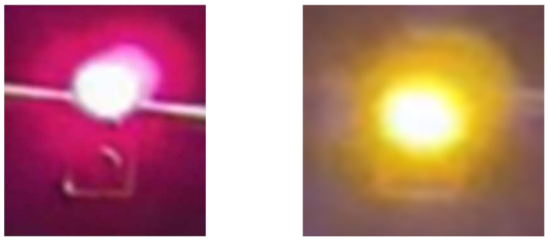
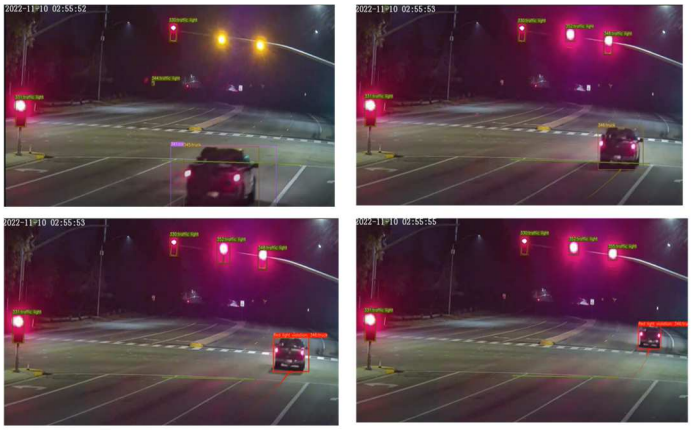


Fig. 10 Vague Traffic Light at Night-Time

In fact, the instability of traffic light recognition at nighttime has caused the traffic light state identification to have a lower accuracy than in the morning. The traffic light state identification is highly dependent on traffic light detection because it is required to detect a traffic light before being processed for traffic light state identification. As the detection model may not be able to detect the traffic light all the time, this has caused the traffic light state not to be identified as no traffic light has been detected.

C. Red Light Violation

This section shows the result of the proposed red light violation detection by feeding the model with real-life traffic videos obtained from real-time CCTV footage. Fig. 11 shows the proposed model's capability in detecting red light violators on the road. When the traffic light is in its red state, a vehicle is not allowed to cross the traffic stop lane detected. Otherwise, the corresponding vehicle will be immediately classified as a red-light violator.



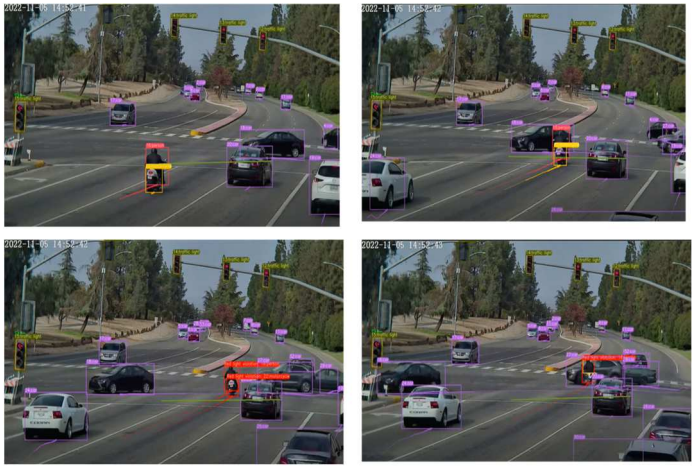


Fig. 11 Red Light Violation Detection Result

During model evaluation, traffic video under the green light state has also been fed into the model to assess the model accuracy and reliability. As shown in Fig. 12, when the current traffic light is in green state, red light violation alert will not be triggered even when a vehicle has crossed the stop line.

Fig. 12 Green Light Result

D. Overtaking Violation

This section shows the results of the overtaking detection by feeding the model with real-life traffic videos obtained from real-time CCTV footage. A traffic video without a double line has been utilized to demonstrate the overtaking scenario, assuming that the predefined lane is the double line on the road. The result in Fig. 13 shows the proposed model output in detecting illegal overtaking on the road. When a vehicle crosses the indicated overtaking lane, that particular vehicle will be immediately classified as overtaking by turning its bounding box to red color and with a text on top stating overtaking has been performed by that specific vehicle as an alert message for the relevant authorities.

Despite the advancements in object detection and tracking technologies, the complexities inherent in real-world traffic environments present several challenges impacting the system's performance. One area for improvement of the proposed system in red light violation detection is its high dependency on the detected traffic stop line. It was found that the proposed model could not detect the red-light violator due to the incomplete detection of the stop line when the vehicle is crossing the identified line in a red state. The stop line was initially accurately detected, but the lane detection became partially obscured when the vehicle crossed the line. Based on the analysis, the reason for it not being able to detect the stop line accurately may be due to the occlusion by the vehicle. The vehicle's presence has blocked the camera's view, leading to the lane not being fully detected at the part where occlusion occurs. To address this issue, it is essential to position the camera strategically to ensure that occlusion would not affect the line detection result.

Fig. 13 Overtaking Violation Detection Result

In contrast, misclassification of overtaking could happen on the vehicle near the edge of the camera frame when it did not perform illegal overtaking. This could be due to the proposed model's inability to effectively handle scale variation and perspective changes. As the vehicle is moving further away from the camera, the scale variation of the vehicle and the changes in perspective may cause the distance calculations to be skewed, leading to a wrong calculation when determining if a car has been overtaken illegally. The misclassification of overtaking detection could be addressed by adjusting the position of the camera viewpoint. In terms of technical enhancement, the possible solution could be to enhance the preprocessing of the image to normalize the perspective distortion using techniques like Homography.

IV. Conclusion

In conclusion, the proposed system shows a promising result in identifying traffic violators on the road. The system works by first receiving an input video. Then, each frame of the video is processed for road object detection including the detection of vehicles and traffic lights. Detected vehicles are provided with their unique identifier by Deep SORT tracker, which is used to keep track of each vehicle as an individual. On the other hand, detected traffic lights in each frame will be processed for traffic lights state identification to identify

whether the current state is red, yellow, or green. In terms of automated lane detection which indicates the white line at traffic lights and the double line for overtaking identification, the lane is detected by combining both Canny Edge Detection and Hough Transform techniques for the automated lane detection in each frame. After these operations, the system processes the frames to determine whether any vehicle has violated the traffic rule by deciding whether they have passed the detected lane when the traffic light state is red or crossed the double line on the road. However, the proposed model depends on the environmental lighting conditions and the camera's position to accurately identify traffic violators. Furthermore, the proposed model relies on the accuracy of object detection. A high object detection accuracy would yield a better performance in identifying red light violators and overtaking violators.

ACKNOWLEDGMENT

This research was funded by the TM R&D Fund (MMUE/220023).

REFERENCES

[1] B. Mohamad, "Statistics on Causes of Death, Malaysia, 2023," Ministry of Economy Department of Statistics Malaysia.

[2] S. K. Ahmed et al., "Road traffic accidental injuries and deaths: A neglected global health issue," Health Science Reports, vol. 6, no. 5, May 2023, doi: 10.1002/hsr2.1240.

[3] Y. Ren, "Intelligent Vehicle Violation Detection System Under Human–Computer Interaction and Computer Vision," International Journal of Computational Intelligence Systems, vol. 17, no. 1, Feb. 2024, doi: 10.1007/s44196-024-00427-6.

[4] D. Zaheri and M. Abbas, "An Algorithm for Identifying Red Light Runners from Radar Trajectory Data," 2015 IEEE 18th International Conference on Intelligent Transportation Systems, vol. 34, pp. 2683–2687, Sep. 2015, doi: 10.1109/itsc.2015.431.

[5] A. Jahangiri, H. A. Rakha, and T. A. Dingus, "Adopting Machine Learning Methods to Predict Red-light Running Violations," 2015 IEEE 18th International Conference on Intelligent Transportation Systems, vol. 2, pp. 650–655, Sep. 2015, doi: 10.1109/itsc.2015.112.

[6] A Alaydrus, W K Putra, Y Nugroho, Muhammad, and N Surantha, "A review of traffic violation detection technology in reporting mechanism," *IOP Conf Ser Earth Environ Sci*, vol. 729, no. 1, pp. 1–7, Apr. 2021, doi: 10.1088/1755-1315/729/1/012005.

[7] S. S. Wankhede and P. Bajaj, "Traffic Violation Detection Model Using Soft Computing Tools," 2021 6th International Conference for Convergence in Technology (I2CT), pp. 1–5, Apr. 2021, doi:10.1109/i2ct51068.2021.9417887.

[8] R. Ravish, S. Rangaswamy, and K. Char, "Intelligent Traffic Violation Detection," 2021 2nd Global Conference for Advancement in Technology (GCAT), pp. 1–7, Oct. 2021, doi:10.1109/gcat52182.2021.9587520.

[9] N. C. Mallela, R. Volety, S. P. R., and N. R. K., "Detection of the triple riding and speed violation on two-wheelers using deep learning algorithms," Multimedia Tools and Applications, vol. 80, no. 6, pp. 8175–8187, Oct. 2020, doi: 10.1007/s11042-020-10126-x.

[10] K. Klubsuwan, W. Koodtalang, and S. Mungsing, "Traffic Violation Detection Using Multiple Trajectories Evaluation of Vehicles," 2013 4th International Conference on Intelligent Systems, Modelling and Simulation, Jan. 2013, doi: 10.1109/isms.2013.143.

[11] M. M. Bachtiar, A. Rahman Mawardi, and A. R. Anom Besari, "Vehicle Classification and Violation Detection on Traffic Light Area using BLOB and Mean-Shift Tracking Method," 2020 International Conference on Applied Science and Technology (iCAST), pp. 94–98, Oct. 2020, doi:10.1109/icast51016.2020.9557628.

[12] A. T. Bhat, Anupama, Akshatha, M. S. Rao, and D. G. Pai, "Traffic violation detection in India using genetic algorithm," Global Transitions Proceedings, vol. 2, no. 2, pp. 309–314, Nov. 2021, doi:10.1016/j.gltp.2021.08.056.

[13] J. C. de Goma, R. J. Bautista, M. A. J. Eviota, and V. P. Lopena, "Detecting Red-Light Runners (RLR) and Speeding Violation through Video Capture," 2020 IEEE 7th International Conference on Industrial Engineering and Applications (ICIEA), pp. 774–778, Apr. 2020, doi: 10.1109/iciea49774.2020.9102059.

[14] P.Srinivas Reddy, T. Nishwa, R. Shiva Kiran Reddy, Ch Sadviq, and K. Rithvik, "Traffic Rules Violation Detection using Machine Learning Techniques," *2021 6th International Conference on Communication and Electronics Systems (ICCES)*, pp. 1264–1268, Aug. 2021.

[15] I. Bozcan and E. Kayacan, "Context-Dependent Anomaly Detection for Low Altitude Traffic Surveillance," 2021 IEEE International Conference on Robotics and Automation (ICRA), pp. 224–230, May 2021, doi: 10.1109/icra48506.2021.9562043.

[16] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 779–788, Jun. 2016, doi: 10.1109/cvpr.2016.91.

[17] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," 2017 IEEE International Conference on Image Processing (ICIP), Sep. 2017, doi:10.1109/icip.2017.8296962.

[18] S. Srivastava and S. Gupta, "Path Detection for Self-Driving Carts by using Canny Edge Detection Algorithm," 2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), pp. 1–5, Sep. 2021, doi:10.1109/icrito51393.2021.9596109.

[19] K. Zhao, Q. Han, C.-B. Zhang, J. Xu, and M.-M. Cheng, "Deep Hough Transform for Semantic Line Detection," IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 1–1, 2021, doi:10.1109/tpami.2021.3077129..

[20] D. Bhatt et al., "CNN Variants for Computer Vision: History, Architecture, Application, Challenges and Future Scope," Electronics, vol. 10, no. 20, p. 2470, Oct. 2021, doi: 10.3390/electronics10202470.

[21] T. Diwan, G. Anirudh, and J. V. Tembhurne, "Object detection using YOLO: challenges, architectural successors, datasets and applications," Multimedia Tools and Applications, vol. 82, no. 6, pp. 9243–9275, Aug. 2022, doi: 10.1007/s11042-022-13644-y.

[22] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors," 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 7464–7475, Jun. 2023, doi:10.1109/cvpr52729.2023.00721.

[23] D. Reis, J. Hong, J. Kupec, and A. Daoudi, "Real-Time Flying Object Detection with YOLOv8," pp. 1–10, May 2024.

[24] M. C. Raja, "Utilizing YOLO 8x Models, Deep Learning-Based Head protector Detection for Construction Workers," Jun. 2023, doi:10.21203/rs.3.rs-3045767/v1.

[25] J. Terven, D.-M. Córdova-Esparza, and J.-A. Romero-González, "A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS," Machine Learning and Knowledge Extraction, vol. 5, no. 4, pp. 1680–1716, Nov. 2023, doi: 10.3390/make5040083.

[26] M. Khodarahmi and V. Maihami, "A Review on Kalman Filter Models," *Archives of Computational Methods in Engineering (2023)*, vol. 30, pp. 727–747, Oct. 2022.

[27] K. Du and A. Bobkov, "An Overview of Object Detection and Tracking Algorithms," *15th International Conference "Intelligent Systems,"* vol. 33, no. 1, pp. 1–6, Jun. 2023.

[28] A. I. B. Parico and T. Ahamed, "Real Time Pear Fruit Detection and Counting Using YOLOv4 Models and Deep SORT," Sensors, vol. 21, no. 14, p. 4803, Jul. 2021, doi: 10.3390/s21144803.

[29] N. D. Lynn, A. I. Sourav, and A. J. Santoso, "Implementation of Real-Time Edge Detection Using Canny and Sobel Algorithms," *IOP Conf. Series: Materials Science and Engineering*, vol. 1096, pp. 1–8, Mar. 2021.

[30] Q. Huang and J. Liu, "Practical limitations of lane detection algorithm based on Hough transform in challenging scenarios," *Int J Adv Robot Syst*, vol. 18, no. 2, pp. 1–13, Mar. 2021.

[31] J. J. Ng, K. O. Michael Goh, and C. Tee, "Traffic Impact Assessment System using Yolov5 and ByteTrack", *J. Inform. Web Eng.,* vol. 2, no. 2, pp. 168–188, Sep. 2023.

[32] M. S. Irsyad, Z. Che Embi, and K. I. B. Ghauth, "Assessing the Efficiency of Deep Learning Methods for Automated Vehicle Registration Recognition for University Entrance", *J. Inform. Web Eng.*, vol. 3, no. 2, pp. 57–69, Jun. 2024.

[33] S. A. Lashari, M. M. Khan, A. Khan, S. Salahuddin, and M. N. . Ata, “Comparative Evaluation of Machine Learning Models for Mobile Phone Price Prediction: Assessing Accuracy, Robustness, and Generalization Performance”, *J. Inform. Web Eng.,* vol. 3, no. 3, pp. 147–163, Oct. 2024.

[34] S. T. Jimoh and S. S Al-Juboori, “Cyber-Securing Medical Devices Using Machine Learning: A Case Study of Pacemaker”, *J. Inform. Web Eng.*, vol. 3, no. 3, pp. 271–289, Oct. 2024.