A ROBUST LANE VIOLATION DETECTION METHOD FROM MOBILE VISION: A GEOMETRIC-TEMPORAL REASONING APPROACH

Nguyen Ngoc Trung[1], Le Duong Khang[1], Dao Van Minh[1], Ngo Viet Huy Hoang[1], Phong-Phu Le[2], Huynh Thanh Tung[1]*, Duy-Tuan Dao[1]

[1]The University of Danang - University of Science and Technology, Vietnam
[2]NeuralTrans Technology Solution Joint Stock Company, Vietnam

*Corresponding author: httung@dut.udn.vn

(Received: January 14, 2026; Revised: March 03, 2026; Accepted: March 11, 2026)

DOI: 10.31130/ud-jst.2026.24(3).032E

Abstract – Crowdsourced dashcam traffic surveillance offers a scalable alternative to fixed infrastructure but faces challenges with geometric instability. This paper presents a framework for automated lane violation detection and license plate recognition using mobile vision. The pipeline integrates UFLDv2 for lane segmentation and YOLOv11 for vehicle localization, followed by OCR for license extraction. A novel geometric-temporal reasoning algorithm models the intersection between the vehicle's rear-wheel baseline and road markings, augmented by temporal filtering to minimize camera vibration noise. Experimental results on 2,753 frames from Da Nang, Vietnam, demonstrate 94% accuracy. Outperforming the conventional bounding-box baseline, the proposed method demonstrates strong feasibility and practical potential for intelligent traffic monitoring in dynamic urban environments.

Keywords - Lane violation detection; UFLDv2; YOLOv11; License plate recognition; OCR.

1. Introduction

The rapid urbanization and increasing complexity of modern traffic systems have led to a surge in road violations, particularly lane encroachments and illegal crossings of solid lines. These behaviors pose significant safety risks and are a leading cause of traffic accidents in metropolitan areas. Traditionally, traffic surveillance relies heavily on fixed camera infrastructures installed at strategic intersections. While effective in specific locations, these systems are burdened by high deployment and maintenance costs, and they inherently suffer from limited coverage scalability, leaving vast stretches of road unmonitored [1].

To address these limitations, the paradigm of traffic monitoring is shifting towards crowdsourced data collected from mobile devices, such as dashcams and smartphones. This approach transforms road vehicles into mobile surveillance nodes, offering a flexible and cost-effective solution to expand monitoring coverage [2]. Recent domestic studies have also demonstrated that deep learning applied to visual data can effectively detect lane-related violations in the context of Vietnam's traffic [3]. However, analyzing video data from mobile cameras presents unique challenges compared to fixed viewpoints. Unlike stationary cameras with calibrated angles, mobile cameras are subject to continuous vibration, changing perspectives, and dynamic environments. Consequently, traditional violation detection methods that rely solely on 2D bounding boxes often fail to accurately determine the spatial relationship between the vehicle and road markings [4]. For instance, a bounding box might overlap with a lane line due to perspective distortion even when the vehicle's wheels have not actually crossed the line, leading to high false positive rates.

In this paper, we propose a robust deep learning-based framework for automated lane violation detection and license plate recognition using mobile vision. Unlike previous approaches that focus primarily on object detection [5], [6], our method reformulates lane violation detection as a geometric-temporal reasoning problem tailored for unstable mobile dashcam environments. By estimating the rear-wheel baseline - the physical contact line between the vehicle and the road - based on robust wheel detection principles [7], and checking its intersection with the road markings identified by the UFLDv2 model [8], we achieve a precise definition of lane violation that is resilient to perspective changes. Furthermore, to mitigate the impact of camera vibration and transient detection errors, we employ a temporal filtering mechanism that validates violations over a sliding time window, inspired by majority voting techniques in action recognition [9], [10].

The system is fully integrated with a license plate recognition module that utilizes YOLOv11 and Optical Character Recognition (OCR), enabling the automatic generation of objective evidence for enforcement. The main contributions of this study are summarized as follows:

1. A comprehensive end-to-end framework is developed for mobile traffic monitoring, integrating state-of-the-art deep learning models (UFLDv2, YOLOv11) to detect lane violations and recognize license plates in real-world conditions.

2. The novelty of this work lies in integrating geometric and temporal reasoning into a unified framework specifically designed for unstable mobile camera data. A geometric violation reasoning scheme is designed to approximate the vehicle's effective ground-contact baseline directly in the original image space, avoiding explicit homography-based ground-plane reconstruction while preserving structural cues for vehicle classification.

3. A temporal consistency validation mechanism is incorporated to ensure stable decision-making under camera vibration and perspective fluctuations, enabling reliable deployment in mobile and infrastructure-free settings.

2. Related works

Mobile-based Traffic Monitoring Systems. Unlike traditional surveillance systems anchored to fixed infrastructure [1], mobile-based approaches utilize sensors and cameras on vehicles or smartphones to extend

monitoring coverage dynamically [3]. Alasmary et al. [5] proposed a crowdsourcing framework that uses smartphone cameras to detect general traffic violations. While this system demonstrated the cost-effectiveness of mobile sensing, it relied heavily on GPS and basic visual features, lacking the geometric precision required to identify subtle lane encroachments.

More recently, Dede et al. [6] introduced an AI-assisted mobile surveillance system utilizing dashcam footage. Their approach combined YOLO object detection with the StrongSORT tracking algorithm to monitor vehicle trajectories. Although effective in tracking moving vehicles, their system focused primarily on motion models and lacked a dedicated road marking recognition module. Consequently, it remains limited in evaluating violations governed specifically by lane constraints, such as crossing solid lines or driving in prohibited zones. Our work addresses this limitation by integrating a specialized lane detection branch that operates in tandem with vehicle localization.

Deep Learning for Lane Detection. Lane detection is a fundamental component of autonomous driving and ADAS. Early methods relied on handcrafted features, such as Hough transforms, which are sensitive to lighting and occlusion [11]. In the deep learning era, semantic segmentation approaches have achieved high accuracy but often suffer from high computational costs, making them unsuitable for real-time mobile applications.

To balance speed and accuracy, row-anchor-based methods have emerged as a superior alternative. Specifically, Z. Qin et al. [8] proposed UFLDv2 (Ultra-Fast Lane Detection v2), which formulates lane detection as a row-based classification problem rather than a pixel-wise segmentation problem. This method significantly reduces inference latency while maintaining robust performance under complex scenarios. Domestic studies in Vietnam have also explored deep learning for lane violation detection [2]. Still, few have successfully optimized these heavy models for mobile platforms while handling the region's unique and chaotic traffic characteristics.

Violation Detection Strategies and Geometric Reasoning. The core challenge in violation detection lies in accurately determining the interaction between the vehicle and road markings. Traditional methods typically define a violation based on the Intersection over Union (IoU) or the overlap between the vehicle's 2D bounding box and the lane area [12]. However, Y. Qin et al. [4] pointed out that bounding boxes often encompass non-contact areas (e.g., the vehicle body hanging over the line), leading to false positives due to perspective projection errors.

3. Methodology

The proposed traffic violation detection system is designed as a sequential pipeline capable of processing video streams from mobile cameras in real-time. As illustrated in Figure 1, the framework comprises three primary functional blocks: (1) Lane Analysis, which segments lane lines and classifies their types (solid/dashed); (2) Object Localization, which detects vehicles and recognizes license plates; and (3) Violation Reasoning, a geometric-based decision module that synthesizes inputs from the previous steps to identify violations. The input data consists of video frames captured at 30 frames per second (fps). The system outputs a violation warning coupled with the vehicle's license plate information only when a consistent solid-line crossing behavior is verified.

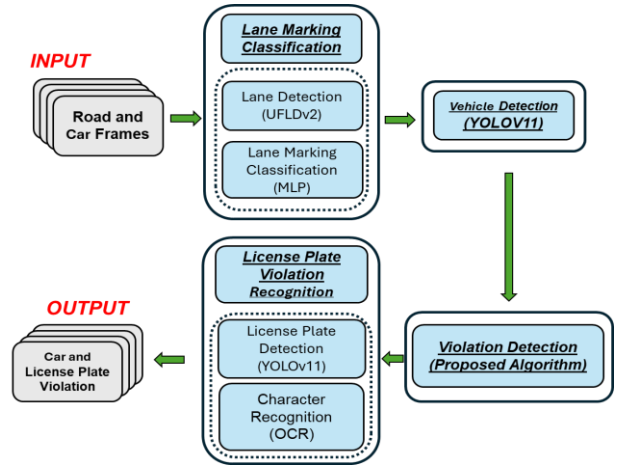

Figure 1. *Overview of the pipeline of the proposed traffic violation detection system*

3.1. Lane Segmentation and Classification

To balance computational efficiency and accuracy on mobile edge devices, we employ a hybrid approach that combines row-anchor-based detection with a lightweight classifier (Figure 2).

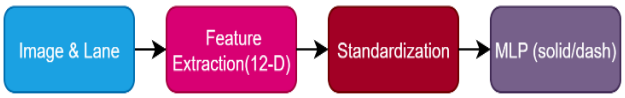

Figure 2. *Lane type classification process*

Lane Detection (UFLDv2): We utilize the UFLDv2 (Ultra-Fast Lane Detection v2 [8]) model to address the high computational latency typically associated with traditional pixel-wise segmentation networks. This model reformulates the lane detection task as a row-based classification problem using global features.

- Input: A resized RGB frame.

- Mechanism: The input image is divided into a grid of fixed rows. For each predefined row, the model predicts the specific column index (grid cell) that contains the lane marking. This approach significantly reduces the computational search space compared to processing every pixel.

- Output: The module generates a set of 2D coordinates $\mathcal{L} = \{(x_1, y_1), \ldots, (x_k, y_k)\}$ representing the geometric centerline of each detected lane on the image plane 3.

Lane Detection (UFLDv2): Since the UFLDv2 model identifies the location of lanes but does not inherently distinguish their semantic type, a lightweight Multilayer Perceptron (MLP) is employed for classification.

- Feature Extraction: Geometric features are derived directly from the lane coordinates $\mathcal{L}$ obtained in the previous step. These features include the length of continuous segments, gap frequency, and periodicity, which effectively characterize the visual difference between solid and dashed lines.

- Lane Classification: The extracted features are

normalized and fed into an MLP to predict a binary state $C_{lane} \in \{Solid, Dashed\}$ for each lane.

- Mask Generation: Based on the classification results, a binary Prohibited Mask $\mathcal{M}_t$. In this mask, pixels corresponding to identified solid lines are assigned a value of 1, establishing the restricted zones used for the subsequent violation reasoning.

3.2. Vehicle Detection and License Plate Recognition

Vehicle Detection: We utilize the YOLOv11 [13] architecture, fine-tuned on the COCO dataset, to localize road vehicles. YOLOv11 is selected for its superior feature aggregation capabilities (C3k2 blocks and SPPF), which enhance the detection of small objects in complex environments.

For each frame I_t, the model outputs a set of bounding boxes $\mathcal{B}_i = (x_1, y_1, x_2, y_2)$, along with class labels (car, truck, bus). These coordinates act as the primary input for the geometric reasoning algorithm.

License Plate Recognition: Upon confirmation of a violation (*as described in 3.3*), the system triggers the license plate recognition pipeline:

- Localization: Another YOLOv11 [14] sub-model detects the license plate region within the vehicle bounding box.

- Enhancement: To mitigate motion blur and low resolution common in dashcam footage, the cropped plate image is processed by a Real-ESRGAN model [15] to strictly enhance super-resolution.

- Character Extraction: An OCR module (fine-tuned YOLOv11) detects individual alphanumeric characters. A frequency-based voting mechanism is applied to the sequence of detected characters to ensure the robustness of the final recognition result.

3.3. Proposed Violation Detection Algorithm

This module constitutes the core contribution of our work. Unlike existing methods that rely on the overlap between a 2D bounding box and lane pixels - which often yields false positives due to perspective projection errors - we propose a fine-grained geometric reasoning approach. This algorithm models the physical interaction between the vehicle's footprint and the road surface (Figure 3).

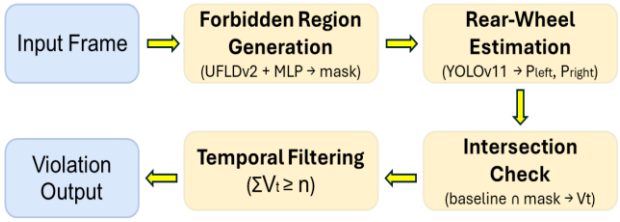

Figure 3. *Algorithm flowchart for violation detection*

3.3.1. Definition of a violation

In the context of this study, a traffic violation is rigorously defined as a temporal event in which a vehicle crosses or encroaches upon a solid longitudinal road marking. To translate this physical event into a computer vision task, we establish two criteria:

1. Spatial Criterion (Frame-level): A frame is considered a "violation candidate" if the line segment connecting the vehicle's two rear wheels intersects with the solid line area. We focus on the rear wheels rather than the vehicle body or front wheels because the rear axle represents the non-steering, stable pivot of the vehicle's trajectory, providing a more reliable indicator of lane discipline.

2. Temporal Criterion (Event-level): A violation is not concluded from a single frame. Instead, it is only recognized when the spatial violation persists for a sufficient duration within a consecutive time window. This ensures that momentary detection jitters or camera vibrations do not trigger false accusations.

3.3.2. Rear-Wheel Baseline Estimation

Since mobile dashcams lack static calibration parameters to project pixels into 3D world coordinates, we employ a heuristic geometric estimation based on the structural invariants of four-wheeled vehicles.

For a vehicle detected by YOLOv11 with a bounding box $\mathcal{B}_i = (x_1, y_1, x_2, y_2)$, the width $w = x_2 - x_1$ and height $h = y_2 - y_1$. The contact points of the left (P_L) and right (P_R) rear wheels on the image plane are approximated using the following linear transformation 4:

$$P_L = (x_1 + \alpha.w,\ y_2 - \beta.h) \quad (1)$$

$$P_R = (x_2 - \alpha.w,\ y_2 - \beta.h) \quad (2)$$

where: α is the horizontal correction coefficient, representing the ratio of the wheel's inset relative to the vehicle's full width. β is the vertical correction coefficient, accounting for the perspective projection where the rear bumper visually occludes the actual ground contact point of the wheels. In our implementation, these parameters are empirically set to $\alpha = 0.18$ and $\beta = 0.06$. The line segment connecting P_L and P_R is defined as the Rear-Wheel Baseline ($\mathcal{S}$).

Figure 4 illustrates the geometric relationship between the vehicle bounding box $\mathcal{B}_i$, the estimated rear-wheel contact points P_L and P_R, the resulting baseline $\mathcal{S}$, together with the detected lane markings used for violation reasoning.

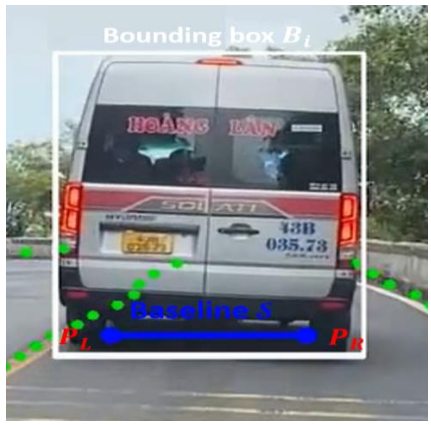

Figure 4. *Rear-wheel baseline estimation from vehicle bounding box*

3.3.3. Intersection Analysis

To verify the spatial criterion, we analyze the interaction between the estimated baseline and the lane markings.

- Prohibited Region Mask ($\mathcal{M}_t$): As described in Section 3.2, a binary mask is generated where pixels belonging to solid lines are set to 1.

- Intersection Check: The baseline $\mathcal{S}$ is rasterized onto

the image plane with a specific thickness τ (experimentally set to τ = 6 pixels). This thickness parameter is crucial as it compensates for minor spatial uncertainties in the UFLDv2 lane segmentation output. A frame t is flagged as a violation candidate ($\hat{v}_t$ = 1) if the thickened baseline overlaps with any active pixel in $\mathcal{M}_t$:

$$\hat{v}_t = \begin{cases} 1 & ,if\ area(\mathcal{S}_\tau) \cap \mathcal{M} \neq \emptyset \\ 0 & ,otherwhise \end{cases} \quad (3)$$

3.3.4. Temporal Noise Filtering

Mobile surveillance footage is inherently noisy due to road surface vibrations and potential flickering from segmentation. Relying on a single frame $\hat{v}_t$ can lead to unstable results. To address this, we implement a Majority Voting Mechanism over a sliding temporal window.

For a window of size m consecutive frames ending at time t, the system calculates the cumulative number of violation candidates. A final violation warning V_{final} is issued only if this sum exceeds a confidence threshold n:

$$V_{final} = \begin{cases} 1 & ,if\ \sum_{k=t-m+1}^{t} \hat{v}_k \geq n \\ 0 & ,otherwhise \end{cases} \quad (4)$$

This mechanism acts as a temporal low-pass filter, ensuring that only consistent, deliberate lane-crossing behaviors are penalized while suppressing transient noise.

4. Experimental results

4.1. Data Collection

4.1.1. Lane Data

The lane dataset was collected by the authors using dashcams mounted on their vehicles while traveling on roads in the Da Nang area (Vietnam). Videos were recorded at a resolution of 1280×720 pixels at 30 fps, from which 2,753 frames were extracted for labeling according to the input format of the UFLDv2 model. In each frame, lanes were classified into two groups: solid lines and dashed lines. Labeling data was stored in.lines.txt files, where each lane was represented by a set of feature points in a row-column anchor structure. Specifically, the image was divided into fixed rows along the y-axis, and at each row, the column coordinates corresponding to the lane position were determined. This discrete representation enables the model to learn the shape and trajectory of the lane efficiently, while reducing the labeling cost compared to full-image segmentation methods. The dataset is split into 70% for training and 30% for validation. This dataset is used to fine-tune the UFLDv2 model for the lane detection task and to train the MLP network for the line type classification problem.

4.1.2. Vehicle and License Plate Data

a. Vehicle detection

The dataset for the vehicle detection problem is built on the standard COCO dataset, which uses only three common vehicle classes: cars, trucks, and buses (corresponding to classes 2, 5, and 7 in COCO). Images are labeled in COCO format, with object positions described by bounding boxes [14].

In this study, three vehicle categories: cars, trucks, and buses are selected as representative four-wheeled vehicles for experimental evaluation. The proposed violation detection algorithm is not limited to these categories and can be extended to other types of four-wheeled vehicles.

b. Plate detection

The training data for the license plate detection problem utilizes the Vietnam License Plate Segment Dataset from Kaggle [16], which comprises approximately 5,000 images of Vietnamese license plates under various lighting conditions, weather conditions, shooting times, and viewing angles. Initially, the data was divided into two classes: single-row license plates (*LpD*) and double-row license plates (*LpV*); however, to focus on the license plate area detection task, these two classes were merged into a single class called license_plate.

Images are labeled as polygons defining the four corners of the license plate, helping the model learn the actual shape and proportions, especially in cases where the license plate is tilted, distorted, or obscured. Data is split into 70% for training and 30% for evaluation.

c. License plate character detection

We utilized a publicly available dataset from GitHub [17], comprising 3,832 images of Vietnamese license plates captured under diverse lighting, weather, viewing angle, and camera quality conditions. The images were cropped and manually labeled character by character with bounding box position in YOLO format, serving to train the YOLOv11 model for the character detection and recognition problem. Data is split into 80% for training and 20% for validation to ensure diversity and objective evaluation of effectiveness.

4.2. System Setup

Experiments were performed in the Google Colab environment with an NVIDIA Tesla T4 GPU (16 GB). The system was implemented using PyTorch, employing OpenCV for image preprocessing and NumPy for numerical operations. This configuration was used consistently throughout the experiment.

All input videos were processed at a fixed rate of 30 fps. System parameters were kept constant throughout the evaluation to ensure consistency between experiments.

In the experiments, geometric coefficients $\alpha = 0.18$ and $\beta = 0.06$ were used to estimate the position of the two rear wheels relative to the bounding box. The line segment connecting the two rear wheels was drawn with a thickness of $\tau = 6$ pixels to compensate for spatial error. To stabilize the decision over time, a majority voting mechanism was applied with a sliding window $m = 7$ frames and a threshold $n = 3$. An violation event is defined as a violation behavior sustained for at least n consecutive frames (≈ n/FPS seconds). Thus, a vehicle must exhibit lane encroachment in at least 3 frames (approximately 0.1 seconds in total at 30 fps) within the 0.23-second temporal window to be identified as a violation event. This mechanism reduces transient noise while maintaining responsiveness to short but deliberate violations.

Experimental evaluation showed that the system's performance was insensitive to small changes in parameters within reasonable value ranges (e.g., $\alpha \in [0.15, 0.25]$, $\beta \in [0.05, 0.10]$, and $m \in [5, 9]$). Therefore, the parameter values reported in this paper are considered representative configurations and were kept fixed throughout the evaluation process.

4.3. Lane Detection Results

The training process of the UFLDv2 model for the lane detection problem shows that the training loss and validation loss values decrease steadily over the epochs, reflecting the model's efficient learning ability. In the first 40 epochs, the loss decreases rapidly; then, the rate of decrease slows down and approaches a stable state, indicating that the optimization process has converged.

The gap between the training loss and validation loss remains small throughout the training process, indicating no significant overfitting. The model was trained using SGD with a batch size of 16 and a learning rate of 0.0125, and the training was stopped at epoch 107 using an early stopping mechanism when the validation loss no longer improved significantly. The evolution of the loss curves is illustrated in Figure 5.

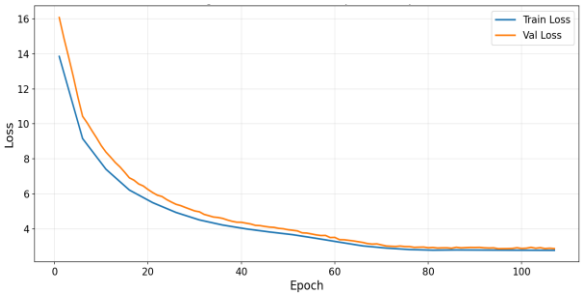

Figure 5. *Training loss and validation loss curves of the UFLDv2 model over the epochs*

Table 1. *Performance evaluation metrics of the UFLDv2 model on the test set*

Metric	TP	FP	FN	Precision	Recall	F1
Value	980	120	665	0.90057	0.5957	0.7187

In the lane spot detection problem, the evaluation metrics are determined based on three fundamental quantities. These metrics are calculated using the following formulas:

$$Precision = \frac{TP}{TP + FP} \quad (5)$$

$$Recall = \frac{TP}{TP + FN} \quad (6)$$

$$F1 = \frac{2.Precision.Recall}{Precision + Recall} \quad (7)$$

Where:

- **TP**: Correct identification of actual lane markings.

- **FP**: Incorrect identification of other objects as lane markings.

- **FN**: Failure to identify actual lane markings.

- **Precision**: Percentage of correctly identified lane markings out of the total number of detected markings.

- **Recall**: Percentage of correctly identified lane markings out of the total number of actual markings.

- **F1**: An overall assessment of the balance between Precision and Recall.

The results in Table 1 indicate that the model achieved a precision of approximately 0.9, demonstrating its ability to identify valid lane points accurately. However, a recall of around 0.6 suggests that some lane points are missed under challenging conditions such as low light or faded markings. An F1 score of approximately 0.72 reflects a moderate balance between precision and recall. Although the recall is relatively modest, the subsequent geometric–temporal reasoning stage mitigates the impact of incomplete lane detection, as evidenced by the overall detection accuracy reported in Section 4.8. Qualitative results in Figure 6 further show that the model preserves lane geometry under complex conditions such as shadows, partial vehicle obstruction, and faded markings.

Figure 6. *Lane detection results of the UFLDv2 model on real-world images*

4.4. Lane Classification Results

During training, the loss function on both the training and evaluation sets gradually decreased with the number of epochs and approached a steady state, indicating that the MLP model converged and showed no signs of overfitting. The evolution of the loss function is shown in Figure 7.

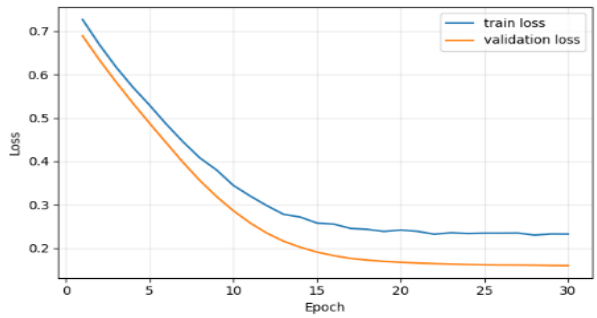

Figure 7. *Graph of the loss function during training*

At the 30th epoch, the validation loss value reached 0.1593, corresponding to an accuracy of 92.74% on the evaluation set. Detailed classification results are presented in Table 2.

Table 2. *Classification results of the MLP model on the evaluation set*

Metric	**Meaning**	**Value**
TP	Correctly identified Solid	354
TN	Correctly identified Dash	732
FP	Misidentified Dash as Solid	28
FN	Misidentified Solid as Dash	57

Evaluation of videos collected under real-world conditions shows that the module is capable of distinguishing solid and dashed lines in various road contexts. Qualitative classification results are presented in Figure 8, where lanes are labeled according to the actual state of the road markings.

Some discrepancies still appear in cases where the road

markings are heavily worn, leading to local confusion between the two types of markings. However, the temporal smoothing mechanism helps stabilize classification results over time, reducing label fluctuations between consecutive frames and ensuring consistency in video processing. Experimental results show that the lane classification module achieves suitable performance for integration into a traffic violation detection system under real-world conditions.

Figure 8. *Lane classification results*

4.5. Vehicle Detection Results

In tests using video data captured from dashcams, the YOLOv11n model pre-trained on the COCO dataset was applied directly without additional fine-tuning. The vehicle detection results on a real video segment are presented in Figure 9.

Figure 9. *Vehicle (bus) detection results on real footage*

In this test scenario, the model correctly identified the vehicle ahead as a bus with a confidence level of 0.91 and accurately determined its position using the bounding box, even in complex curves and challenging background conditions.

The experimental results show that the YOLOv11n model maintains stable vehicle detection capabilities under real-world motion conditions.

4.6. License Plate Detection and Recognition Results

4.6.1. License Plate Detection

The YOLOv11n model was evaluated on a dataset of 1,145 images containing a total of 1,313 license plates. Quantitative results are presented in Table 3, with a precision of 0.989 and a recall of 0.980, indicating that most license plates were detected accurately with a low miss rate.

The @50 index of 0.995 reflects high detection efficiency at IoU thresholds ≥ 0.5, while mAP@50–95 of 0.908 shows that the model maintains good performance at stricter IoU thresholds. The results of license plate detection on real-world data are presented in Figure 10.

Figure 10. *Actual license plate detection results*

Table 3. *Results of evaluating the license plate detection module*

Metric	Images	Instances	Precision	Recall	mAP@ 50	mAP@ 50-95
Value	1,145	1,313	0.989	0.980	0.995	0.908

Note:

- mAP (mean Average Precision) measures the overall effectiveness of the object detection model.

- Average Precision (AP) represents the area under the Precision–Recall curve for each specific class:

$$AP = \int_0^1 P(R)\, dR \tag{8}$$

Where $P(R)$ is Precision at the Recall value R.

- mAP is the mean of the AP values across all classes:

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i \tag{9}$$

Where AP_i is the AP for the i-th class and N is the total number of classes.

- mAP@50: calculate mAP with IoU ≥ 0.5.

- mAP@50-95: calculates the average mAP across multiple IoU thresholds from 0.5 to 0.95 (steps of 0.05), providing a more generalized level of accuracy.

4.6.2. License Plate Character Recognition

Before performing character recognition, license plate images were pre-processed using a Real-ESRGAN model to improve resolution and detail. This pre-processing step helps clarify character edges and reduce noise in license plate images captured from real cameras.

Based on the enhanced images, the YOLOv11n model was used to detect and classify characters (letters and numbers) on license plates. The model was fine-tuned over 30 epochs with an input size of 640×640 pixels and a batch size of 32. Images of other sizes were normalized through resizing and padding to ensure consistent input.

Table 4. *Performance evaluation results of the YOLOv11n model on the validation dataset*

Metric	Precision	Recall	F1	mAP@50	mAP@50–95
Value	0.9693	0.9742	0.9717	0.9815	0.7587

Evaluation results on the validation dataset showed that the model achieved a precision of 0.9693, a recall of 0.9742, and an F1-score of 0.9717. Furthermore, mAP@50 reached 0.9815 and mAP@50–95 reached 0.7587, indicating stable character recognition capabilities under

real-world conditions. Detailed metrics are presented in Table 4.

Visual illustrations of the license plate image preprocessing step and the character detection and recognition results are presented in Figures 11 and 12, respectively.

Figure 11. *License plate preprocessing results after passing through Real-ESRGAN*

Figure 12. *License plate character detection results*

4.7. Violation Detection Evaluation Results

The violation detection module was evaluated on 1,200 video frames, including instances of vehicles crossing solid lines and moving in the correct lane. Frames were extracted from various video segments and manually labeled to build the evaluation dataset for the violation detection module.

Table 5. *Violation Detection Module Evaluation Results*

Metric	Value
Number of Tested Frames	1,200
Number of Actual Violations	450
Number of Violations Detected by the System	420
Number of Actual Non-Violations	750
Number of Non-Violations Correctly Identified	708
Overall Accuracy	94%

According to Table 5, the module achieved an overall accuracy of approximately 94.0%, correctly classifying 420 violation frames and 708 non-violation frames out of a total of 1,200 test frames. When considering the violation class alone, the module achieved a precision of approximately 91.3% and a recall of roughly 93.3%, demonstrating effective violation detection while maintaining reasonable control over the number of false positives.

(a) vehicle not in violation *(b) vehicle in violation*

Figure 13. *Results of detecting violations on the actual frame*

The results of detecting violations on actual frames are presented in Figure 13. In cases where the vehicle is moving in the correct lane, the straight line connecting the two rear wheels does not intersect with the solid line area. Conversely, when the car crosses the solid line, the baseline crosses the solid line area, and the system issues a violation warning.

Analysis of misclassified frames reveals three systematic groups of errors, primarily stemming from environmental factors and the geometric characteristics of the landscape. The first group involves False Negatives occurring when faded or obscured solid lines to prevent the model from generating a complete prohibited mask, leading to missed violations. The second group consists of False Positives which arise when vehicles move extremely close to dashed lines where minor spatial deviations or classification errors trigger false intersection warnings. Finally, “intersection zone confusion” occurs at complex junctions where merging lanes and camera vibrations create temporary segmentation noise.

Although some local biases still exist, the quantitative results and illustrative examples show that the violation detection module operates stably and consistently under test conditions.

4.8. Comparison of Violation Detection Strategies

To evaluate the effectiveness of the proposed violation detection algorithm, the team compared it with several simpler violation detection strategies on the same test dataset of 1,200 manually labeled video frames. The comparison methods all used the same output from the lane detection module and the vehicle detection module, differing only in how the violation conditions were defined.

Table 6. *Comparison of Violation Detection Strategies*

Method	Brief Description	Accuracy
No time filtering (1 frame = violation)	Vehicle is considered to be in violation if baseline intersects the line at a single frame	86%
Using a bounding box instead of the rear wheel	Checks for intersection between the bounding box and the solid line	89%
Proposed method	Baseline of the two rear wheels + time filtering 3/7 frame	94%

The results in Table 6 show that the proposed method achieved the highest accuracy (94%) compared to the comparison strategies. The method without time filtering had lower accuracy (86%) due to sensitivity to short-term noise when decisions were made independently on each frame, a phenomenon also noted in previous studies [12]. The method using the vehicle's bounding box improves accuracy (89%); however, it does not accurately reflect the actual contact position between the vehicle and the road surface, especially in cases where the vehicle is large or changes direction of movement [4].

Conversely, the proposed method utilizes a baseline that connects the two rear wheels - the area of direct contact with the road surface - to more accurately model lane crossing behavior. This approach is based on studies exploiting local features at the wheel level, which provide higher reliability than whole-vehicle features [7]. The time-based filtering mechanism, utilizing a 3/7 frame sliding window, is similar to the majority voting technique in time series recognition [9], [10], and helps reduce noise and stabilize the detection decision. The combination of geometric constraints and time-based filtering enables the

proposed method to achieve higher accuracy and stability compared to competing strategies.

To further examine the impact of the temporal window parameter, we conducted a lightweight ablation study by varying the sliding window size while keeping all other components unchanged. Since the no-filtering case (W = 1) has been described in Table 6, this analysis focuses on practical multi-frame configurations (W ≥ 3). The evaluation was performed on the same 1,200-frame test set.

Table 7. *Ablation Study on Temporal Window Size*

Window Size (W)	Accuracy
3	90%
5	92%
7 (default)	**94%**
9	93%

The results show that introducing short temporal aggregation (W ≥ 3) improves robustness against frame-level noise. Performance increases as W grows and stabilizes around W = 7–9. Further expanding the window size is difficult to implement, as this could increase latency during authentication without providing a significant performance improvement. These results show that the chosen window size achieved a reasonable balance between stability and responsiveness, thereby ensuring performance when deployed on mobile devices in a real-time environment.

4.9. Overall Results

Figure 14. *Final results of the entire system pipeline*

The integrated system's modules operate consistently on real-world data. UFLDv2 identifies lane shapes, MLP classifies lane markings and provides semantic information, YOLOv11 detects vehicles and combines Real-ESRGAN for license plate recognition, and a geometric estimation step determines the position of the two rear wheels to check for intersections with solid lines. Figure 14 illustrates the output of the entire pipeline, showing the stable linkage between modules throughout the processing of real-world traffic data.

5. Conclusion

This paper proposes a cost-effective mobile traffic monitoring framework integrating UFLDv2 for lane analysis and YOLOv11 for vehicle localization. Our core contribution is a robust violation detection algorithm that leverages the rear-wheel baseline and temporal filtering to overcome the geometric instability inherent in dashcam footage. Experimental validation on a real-world dataset in Vietnam achieved 94% accuracy, demonstrating measurable performance gains over a conventional bounding-box-based baseline under the same experimental setting. Future work will extend the system to include motorcycle detection, which is crucial in the Vietnamese traffic context, and improve robustness under adverse weather and nighttime conditions. We also plan to conduct event-level evaluations to reflect real enforcement procedures better. Finally, a quantitative comparison with Inverse Perspective Mapping based (IPM)-based approaches will be performed, along with the development of an optimized IPM strategy tailored for mobile camera settings and Vietnamese road conditions.

Acknowledgment. This research is funded by The Murata Science Foundation and The University of Danang - University of Science and Technology, under Project Code: T2024-02-04MSF.

REFERENCES

[1] S. Djahel, R. Doolan, G.-M. Muntean, and J. Murphy, "A communications-oriented perspective on traffic management systems for smart cities: Challenges and innovative approaches," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 125–151, 2015.

[2] P. Mohan, V. N. Padmanabhan, and R. Ramjee, "Nericell: Rich monitoring of road and traffic conditions using mobile smartphones," in *Proc. 6th ACM Conference on Embedded Network Sensor System* (SenSys), Raleigh, NC, USA, 2008, pp. 323-336.

[3] N. D. T. Giang and N. H. Phuc, "Deep Learning–Based Detection of Lane Violation in Road Vehicles," *Journal of Marine Science and Technology*, no. 08, pp. 62–67, Vietnam Maritime University, 2022.

[4] Y. Qin *et al.*, "Efficient roadside vehicle line-pressing identification in intelligent transportation systems with mask-guided attention," *Sustainability*, vol. 17, no. 9, 2025.

[5] W. Alasmary, "An innovative smartphone-based solution for traffic rule violation detection," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 11, no. 1, pp. 625-636, 2020.

[6] D. Dede, M. A. Sarsıl, A. Shaker, O. Altıntaş, and O. Ergen, *"Next-gen traffic surveillance: AI-assisted mobile traffic violation detection system*," arXiv preprint arXiv:2311.16179, 2023.

[7] S. Ghanem and R. A. Kerekes, "Robust wheel detection for vehicle re-identification," *Sensors*, vol. 23, no. 1, p. 393, 2023.

[8] Z. Qin, P. Zhang, and X. Li, "*Ultra-fast deep lane detection with hybrid anchor-driven ordinal classification*," arXiv preprint arXiv:2206.07389, 2022.

[9] L. Wang *et al*., "Temporal Segment Networks for Action Recognition in Videos," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 11, pp. 2740-2755, 1 Nov. 2019.

[10] Y. Zhao and P. Krähenbühl, "Real-time online video detection with temporal smoothing transformers," in *Proc. European Conf. Computer Vision (ECCV)*, Tel Aviv, Israel, 2022, pp. 458–475.

[11] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 4th ed. New York, NY, USA: Pearson, 2018.

[12] S. Kumar and V. Vaidehi, "Traffic rule violation detection in traffic video surveillance," *International Journal of Computer Science and Electronics Engineering (IJCSEE)*, vol. 3, no. 4, pp. 301–307, 2015.

[13] G. Jocher and J. Qiu, "Ultralytics YOLO11," version 11.0.0, Ultralytics, 2024. [Online]. Available: *https://github.com/ultralytics/ultralytics*

[14] T.-Y. Lin *et al*., "Microsoft COCO: Common objects in context," arXiv preprint arXiv:1405.0312, 2014.

[15] X. Wang, L. Xie, C. Dong, and Y. Shan, "Real-ESRGAN: Training real-world blind super-resolution with pure synthetic data," arXiv preprint arXiv:2107.10833, 2021.

[16] D. Nguyen, "License plates," Kaggle Dataset, 2025. [Online]. Available: *https://www.kaggle.com/datasets/duydieunguyen/licenseplates* [Accessed Nov. 8, 2025].

[17] W. Nguyen, "Real-time auto license plate recognition with Jetson Nano," GitHub Repository, Dec. 2020. [Online]. Available: *https://github.com/winter2897/Real-time-Auto-License-Plate-Recognition-with-Jetson-Nano* [Accessed Dec. 29, 2025].