

# Research on an Identification System for Illegal Occupation of Bus-only Lanes Based on YOLOv8-Seg and ByteTrack

Shengyuan Lei

*Chang'an University, Xi'an, Shaanxi, China*

**Abstract:** Traffic violations involving social vehicles illegally occupying bus-only lanes have occurred frequently. Traditional regulatory approaches based on manual monitoring and patrol-based image capture often struggle to ensure detection efficiency and supervision accuracy during peak traffic periods or under adverse weather conditions. Based on computer vision technology and integrated with advanced visual algorithms, this study develops an identification and detection system for illegal occupation of bus-only lanes by motor vehicles. The dataset used in this study was obtained from videos captured by fixed roadside devices. During the data preprocessing stage, bus lanes and vehicles were annotated, and data augmentation strategies such as instance paste were employed to improve dataset quality. To achieve end-to-end synchronous detection of vehicle positions and lane areas, the YOLOv8-Seg model was adopted for training and detection. To establish a scientifically sound and compliant violation judgment mechanism, this study introduced the ByteTrack high-precision vehicle target tracking algorithm and integrated bottom projection, the ray casting method, and a time-threshold-based state machine to determine whether a vehicle continuously occupied a bus-only lane, thereby reducing false alarms caused by short-term lane crossing and legal lane changing. Experimental results show that the high-precision visual algorithm constructed in this study ensures the accuracy of identifying illegal occupation of bus-only lanes and effectively reduces the probability of misjudgment. While lowering hardware deployment costs, the proposed approach can assist urban traffic law enforcement and provide a reference for intelligent traffic management and control in modern cities.

**Keywords:** Intelligent Transportation System; YOLOv8; ByteTrack; Object Detection;

## Instance Segmentation

### 1. Introduction

By the end of 2025, the total number of civilian automobiles in China had reached 366.11 million. The rapid growth in motor vehicle ownership has further intensified traffic congestion. As a high-capacity mode of public transportation, buses can carry 10 to 15 times as many passengers per trip as ordinary non-bus vehicles, while occupying only approximately half of the road space. In the process of prioritizing public transportation, the establishment of bus-only lanes has become an important measure. Bus-only lanes provide dedicated right-of-way for buses through physical separation or time-based separation, with the aim of reducing conflicts in mixed traffic flows. However, traffic violations involving non-bus vehicles illegally occupying bus-only lanes occur frequently. Therefore, how to promptly identify and penalize the illegal use of bus-only lanes through non-manual means has become a key research focus in both traffic management practice and academia.

The significance of this study is mainly reflected in the following three aspects. First, at the theoretical level, this study integrates object detection technology with multi-object tracking algorithms to achieve spatiotemporal correlation analysis of traffic violations, thereby providing new methodological support for the theoretical framework of dynamic and complex traffic scene understanding. Second, at the social level, the automatic monitoring of motor vehicle violations through technical means can effectively curb lane encroachment, safeguard the priority right-of-way of public transportation, and help alleviate urban traffic congestion. Third, at the engineering application level, the automatic identification of violating vehicles and the generation of an enforcement evidence chain containing information such as time and location can support the construction of an off-site law enforcement system, greatly reduce the cost of manual patrols, and facilitate the intelligent

transformation of urban traffic violation enforcement.

This paper aims to design an intelligent traffic monitoring system for complex urban road traffic video data. The system is intended to achieve high-precision, robust, and intelligent judgment of illegal occupation of bus-only lanes by motor vehicles, and to establish corresponding recognition and judgment rules adapted to the system. This paper first determines the technical route for bus-lane occupation detection. Existing studies on traffic violation detection are mainly based on two types of acquisition devices: vehicle-mounted mobile cameras and fixed cameras. Regarding vehicle-mounted mobile cameras, Chen Meng developed a bus-lane occupation violation detection system based on an improved Hough transform algorithm and a Haar-feature-based AdaBoost cascade classifier for license plate detection [1]. Regarding fixed cameras, Dawood et al. used YOLOv8 training and PaddleOCR-based license plate recognition to automate the enforcement process for bus-lane occupation violations [2]. In China, Sun Xiyun proposed a CLDL\_YOLOv10 joint model architecture and applied it to the scenario of illegal use of bus lanes by motor vehicles through multi-task joint judgment logic [3]. Considering existing research achievements and hardware constraints, this paper adopts fixed cameras as the data acquisition method and conducts research by integrating advanced visual algorithms. The core of the proposed system lies in accurately defining restricted areas, namely lane markings, and continuously tracking violation targets, namely vehicles. Based on this, the current status of lane marking recognition, vehicle detection, and target tracking technologies at home and abroad is reviewed as follows.

Early research on lane marking recognition mainly relied on the inherent features of lane markings, such as shape, texture, and edges, including methods such as Lab color space transformation and inverse perspective mapping. However, such methods are highly prone to failure under adverse weather conditions or nighttime scenarios. To improve robustness, traditional model-driven strategies have been widely applied. Li Zhiyuan et al. extracted regions of interest after preprocessing based on traditional edge operators, thereby further improving detection accuracy [4]. In recent years,

deep-learning-driven strategies have become mainstream. Ge et al. proposed an end-to-end lane marking recognition algorithm based on the integration of convolution and Transformer structures [5].

Vehicle detection serves as the prerequisite for violation capture. Early vehicle detection methods were mostly based on motion information and handcrafted features. Methods such as optical flow, background subtraction, and Haar-like features combined with frame differencing consume substantial computational resources and are susceptible to illumination and shadow interference. With the rapid development of deep learning, convolutional neural networks have become the mainstream approach. In terms of two-stage algorithms, Girshick's team successively proposed R-CNN and Fast R-CNN, and later Ren et al. proposed Faster R-CNN, which established an end-to-end detection framework and significantly improved detection accuracy [6]. To meet industrial requirements for real-time performance, one-stage algorithms emerged. Represented by the YOLO series, one-stage algorithms, such as the YOLOv8 framework, have demonstrated unparalleled engineering advantages in both vehicle detection accuracy and inference speed [7].

After obtaining vehicle coordinates in a single frame, tracking technology must be introduced to realize temporal judgment of violation behavior. Traditional tracking methods, such as the Mean Shift algorithm, Particle Filter algorithm, and Kalman Filter algorithm, have limited tracking stability. At present, the tracking-by-detection paradigm is the mainstream approach for solving multi-object tracking problems. To further improve association success rates in scenarios involving small targets and severe occlusion, Zhang et al. proposed the ByteTrack algorithm. By introducing a secondary matching mechanism for low-confidence detection boxes, ByteTrack reduces its strong dependence on appearance features and demonstrates high tracking stability in complex traffic flows [8].

Current mainstream methods for detecting illegal occupation of bus-only lanes still face several limitations, including inaccurate recognition caused by sparse vehicle-type samples, vehicle tracking loss caused by video stuttering or frame skipping, and high hardware deployment costs. Therefore, this paper designs a lightweight and

fully automated intelligent detection system. The main research contents include the following five aspects. First, a dedicated traffic dataset is constructed and enhanced, enabling the system to accurately recognize buses even when the number of bus images is limited. Second, in the study of YOLOv8-based vehicle and bus-lane recognition, this research constructs an end-to-end multi-task perception network similar to YOLOP, in which vehicle recognition and bus-lane recognition share the same backbone network, thereby avoiding computational redundancy caused by training two separate models. Third, ByteTrack target tracking technology is introduced to link the same vehicle across frames into a continuous motion trajectory, effectively preventing missed detections caused by video frame extraction or frame skipping. Fourth, anti-misjudgment violation capture rules are designed to reduce the false alarm rate. Fifth, the system automatically records data and generates analytical charts, thereby completing a closed-loop process from algorithm-based violation capture to final result report generation.

## 2. Dataset Construction and Preprocessing

### 2.1 Data Acquisition and Image Preprocessing

To ensure the robustness and generalization ability of the intelligent monitoring model in real-world road environments, this study conducted targeted data acquisition for actual traffic scenarios and used a computer vision library to automatically preprocess the original videos.

#### 2.1.1 Acquisition location and scene construction

The data acquisition site was selected from road sections around Zhuhong Road in Xi'an, where traffic flow is heavy and road conditions are relatively complex. The acquisition period covered the midday off-peak hours with sufficient illumination, the morning and evening peak hours with significant backlight interference, and nighttime periods with relatively weak lighting conditions. The data were collected from five intersections around Yuanshuo Road in Xi'an. During the experiment, As shown in Figure 1, partial videos from four of these intersections were used as the original dataset to ensure diversity in scene distribution.

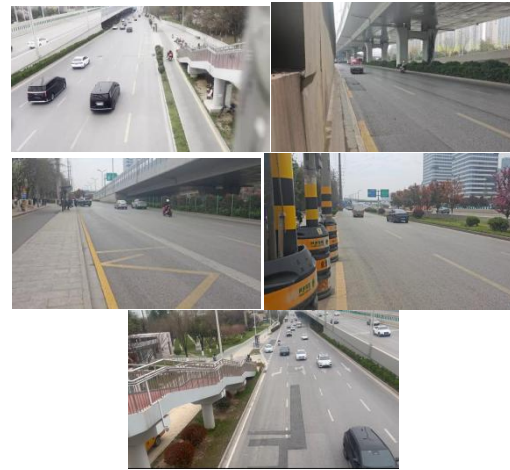


**Figure 1. Yuanshuo Road Intersection in Xi'an**

#### 2.1.2 Video frame extraction and data cleaning based on OpenCV

Since the input of the deep learning object detection model YOLOv8 consists of static images, this study used Python and its open-source computer vision library OpenCV to develop an automated video frame extraction script, converting continuous video streams into a discrete image dataset. After automated frame extraction with OpenCV and rigorous data cleaning, a total of 484 high-definition and high-quality base images were obtained, with 121 images from each intersection. Finally, a high-quality original image dataset containing typical urban road characteristics was constructed, providing a reliable data source for subsequent model training.

### 2.2 Dataset Preprocessing and Multi-Task Annotation

This system aims to achieve joint determination of vehicle localization and violation behavior; therefore, differentiated classification annotation is required for individual vehicles and lane areas. The annotation of the dataset strictly follows national standards and incorporates intelligent simplification based on the engineering characteristics of fixed camera positions.

#### 2.2.1 Fine-grained classification annotation of vehicle targets

For dynamic vehicle targets on the road, this study used the LabelImg annotation tool for processing. Detailed annotation was performed on the images. According to the classification specifications of Road Traffic Signs and Markings (GB 5768) [9] and actual law enforcement requirements, vehicle targets were finely divided into different categories, such as

car, bus, and truck.

To prevent conflicts in vehicle target classification during subsequent code training and later monitoring and capture by surveillance devices, labels were predefined in advance. Label number 1 represents bicycles, label number 2 represents private cars, label number 3 represents motorcycles, label number 4 represents bus lanes, label number 5 represents buses, and label number 7 represents trucks.

#### 2.2.2 Bus-only lanes and the static ROI strategy

According to the national standard for traffic markings, bus-only lanes are usually delineated by yellow dashed or solid lines. In conventional instance segmentation tasks, it is necessary to outline the lane polygon in each frame, which requires an extremely large amount of manual work. Considering that the camera positions of surveillance devices are relatively fixed during the actual deployment of this system, the lane markings in the image belong to static physical facilities. Therefore, this study proposes an extremely simplified engineering strategy of “one-time annotation plus static ROI.” Specifically, the Labelme tool was used to accurately draw the bus-lane polygon along the inner edge of the yellow lane markings on a single, or very small number of, reference images with clear intersection backgrounds, generating lane mask coordinates in JSON format. In the subsequent algorithm pipeline, these coordinates were directly converted into a global static region of interest (ROI). The system only needs to calculate whether the center point of the bounding box or the chassis area of a dynamic vehicle geometrically collides with the static ROI to determine whether the vehicle is suspected of illegally occupying the bus lane. This completely eliminates the heavy workload of repeatedly annotating lane markings in thousands of images. In practice, the bus-lane coordinates were marked at four intersections and then copied in batches into the corresponding TXT-format label files.

#### 2.2.3 Fine-grained polygon annotation of vehicle targets

Due to the perspective effect of the camera, distant bus lanes appear narrow while nearby lanes appear wide, forming a trapezoid or irregular quadrilateral in the image. Only pixel-level polygon contours can enable the system to accurately determine whether the tires of a bus have pressed against the edge line of the lane. Therefore, in addition to learning the center

point, length, and width of the vehicle, the model also needs to learn the precise edge contour of the vehicle. The Labelme tool was still used to perform polygon annotation on vehicle targets and generate mask coordinates. The mask polygon labels were then made downward compatible with the object detection task.

### 2.3 Data Augmentation Strategy for Long-Tail Distribution

In naturally collected traffic flows, the proportion of non-bus vehicles is significantly higher than that of buses. This may cause the model to over-recognize non-bus vehicles while lacking sufficient recognition ability for buses. Such a typical long-tail data distribution can lead to insufficient learning of violation-related features by the model. To address this problem, this study designed a joint data augmentation scheme based on generative adversarial methods and instance stitching.

#### 2.3.1 Instance copy-paste

To directly expand the sample size of buses, which are the core target for detection, this system designed an automated instance-paste strategy. Specifically, first, target extraction technology was used to cut out normally operating bus targets from the original images, thereby constructing a bus instance material library. Second, instance-level Copy-Paste data augmentation was adopted. That is, Python and OpenCV were used to forcibly paste transparent bus instances onto intersection background areas with no vehicles or only a few vehicles, according to dynamically adaptive scaling ratios and random coordinates. Finally, the program automatically generated new coordinate labels for the pasted positions. This scheme can generate a large number of realistic synthetic samples in batches and with high efficiency without consuming substantial computing power to render the entire image. It thus alleviates class imbalance at the physical data level. Therefore, this experiment employed this method for data augmentation to increase the amount of bus-background image data. In this study, 200 base images were selected and pasted with extracted bus instances.

The target extraction technology used in this process was U<sup>2</sup>-Net [10] for salient object detection. Specifically, the rembg library, a Python tool library based on U<sup>2</sup>-Net, was used to accurately extract real bus images into transparent PNG files. As shown in Figure 2, 3

and 4.



**Figure 2. Background Image with Pasted Bus Instance**



**Figure 3. Extracted Bus Image**



**Figure 4. Example of Pasting the Bus Image onto the Background**

## 2.4 Transfer Learning Strategy

Due to the limited scale of the self-built Yuanshuo Road dataset in Xi'an, directly training the deep neural network YOLOv8 model from scratch with randomly initialized weights would make the model highly prone to falling into local optima, while also resulting in a long training cycle and a high risk of overfitting. Therefore, this experiment adopted a transfer learning strategy[11]. First, the officially pre-trained best.pt weight file on the large-scale general-purpose COCO dataset [12], which contains 330,000 images and 80 categories, was loaded. This pre-trained model already possesses strong low-level visual feature extraction capabilities, such as edge, texture, and basic geometric shape extraction. Subsequently, some shallow feature extraction layers of the network were frozen, and only the high-level semantic layers and detection head of the model were fine-tuned using the self-built vertical traffic dataset in this study. This strategy enabled the model to rapidly adapt to the specific bus-lane violation capture task and achieve high-precision convergence with very low computational cost

and time consumption.

## 3. Detection and Segmentation Algorithm Based on YOLOv8

This chapter focuses on the application and training of the core algorithm in this study. To meet the dual requirements of “target localization” and “region delineation” in bus-lane violation monitoring, the YOLOv8 object detection algorithm was adopted to identify vehicles and bus lanes in videos. Instead of using a traditional two-model cascade scheme, this study improved the YOLOv8 architecture by introducing a parallel lane-line segmentation branch into the detection head, thereby constructing an end-to-end multi-task perception network similar to YOLOP. This architecture can synchronously output vehicle positions and lane semantic regions, significantly reducing computational overhead while improving system integration. It also lays the algorithmic foundation for the construction of the subsequent violation judgment model.

### 3.1 Design of the Joint Task Detection Algorithm

Multi-task perception-based violation detection schemes usually require the sequential operation of an object detection model for vehicle recognition and a semantic segmentation model for lane recognition. This can cause serious computational redundancy on mobile devices. Based on the YOLOv8-Seg network for instance segmentation, this study constructed a lane-vehicle dual-task joint detection architecture. The algorithm shares a backbone network to extract image features. After the input image undergoes deep semantic feature extraction through the C2f module, multi-scale feature fusion is performed in the neck network. At the output end, the network is divided into two independent branches. The detection branch adopts an anchor-free mechanism and is responsible for regressing vehicle bounding boxes and predicting vehicle categories. The segmentation branch adds a ProtoNet-based mask generation branch, which is specifically used to extract pixel-level features of bus-lane regions. At the same time, it also extracts pixel-level features of all vehicles, including cars, buses, and trucks. This provides a more accurate basis for determining whether a violating vehicle occupies a bus-only lane. This end-to-end output mode ensures absolute alignment between

vehicle positions and lane regions in the spatiotemporal dimension, avoiding error accumulation caused by multi-model cascading.

### 3.1.1 YOLOv8-seg instance segmentation algorithm

The limitations of traditional object detection methods, such as ordinary YOLO-series models, in intelligent transportation systems should first be clarified. When rectangular bounding boxes are used for localization, they cannot effectively handle irregular lane-line boundaries caused by camera perspective distortion. In addition, during morning and evening peak hours, overlapping bounding boxes caused by dense traffic may lead to the failure of violation judgment logic. Therefore, this study introduces the YOLOv8-Seg network based on instance segmentation, aiming to obtain the true geometric topological structures of vehicles and lanes through pixel-level masks, thereby providing high-precision spatial data support for subsequent lane-crossing violation judgment.

In the overall network architecture, the backbone and neck networks are the same as those of the standard YOLOv8 model. The key differences lie in the decoupled head and the mask generation mechanism. Unlike earlier coupled outputs, YOLOv8-Seg adopts a highly decoupled head design. In addition to the traditional classification and bounding box regression branches, the model establishes an independent pixel segmentation pipeline at the end of the feature fusion layer. This pipeline draws on the efficient mask generation mechanism of the YOLACT algorithm [13] and consists of two parallel subtasks. The first is prototype mask generation, namely ProtoNet, which uses a fully convolutional network to generate  $k$  global prototype masks. These prototypes can respond to different spatial image features, such as vertical edges and horizontal lane-line distributions. The second is mask coefficient prediction. While predicting bounding boxes, the detection head outputs a coefficient vector of length  $k$ , representing the weights of each prototype mask for the current instance, such as a specific bus.

Finally, the pixel mask of the target instance is generated through a linear combination of the prototype matrix  $P$  and the coefficient vector  $C$ , and is activated by the Sigmoid function. Its mathematical expression is as follows:

$$\text{Mask} = \sigma(P \times C^T) \quad (1)$$

This mechanism avoids the high computational

overhead of the target-by-target RoIAlign operation used in traditional Mask R-CNN, thereby ensuring real-time inference speed for intersection video streams.

### 3.2 Dataset Division

The original dataset used in this experiment consisted of 484 high-quality images collected and enhanced from Yuanshuo Road in Xi'an. To verify the generalization ability of the model and prevent overfitting, the dataset was strictly shuffled and divided according to the conventional ratio in the field of deep learning. The training set accounted for 80% of the dataset, containing 387 images, and was used for iterative updating of model weights. The validation set accounted for 20%, containing 97 images, and was used to evaluate model performance and fine-tune hyperparameters during training.

As shown in Figure 7, the processed dataset presents the category distribution and size characteristics of the targets. It shows the number of instances for each category, among which the "car" category has a larger number than the other categories, indicating that ordinary samples are relatively abundant. The box plot in the upper-right corner presents the distribution of target sizes. The density plot in the lower-left corner represents the concentration tendency of targets in the image, while the density plot in the lower-right corner shows the common ranges and distribution characteristics of target width and height. This indicates that the processed dataset can relatively accurately simulate the target distribution in real-world environments, including a large number of small-scale and occluded targets, thereby laying a solid data foundation for the training of the object detection algorithm.

### 3.3 Model Training Implementation and Core Parameter Description

#### 3.3.1 Analysis of training details and batch settings

This experiment conducted fine-tuning training based on the YOLOv8 instance segmentation base model, namely yolov8n-seg.pt. To enable the network to fully learn the spatial features of buses and bus-only lanes at specific intersections, this study carefully configured the training hyperparameters.

The input resolution was set to  $\text{imgsz}=640$ , meaning that all input images were uniformly

resized to  $640 \times 640$  pixels. This resolution preserves the details of traffic signs and vehicle contours while effectively controlling GPU memory usage. The number of training epochs and batch size were set to epochs=150 and batch=16, respectively. The maximum number of training epochs was set to 150, and the batch size was set to 16. This ensured that the model had sufficient training cycles for gradient descent while maintaining a balance between memory usage and computational efficiency on a single graphics card. An early stopping mechanism was introduced with patience=40 to prevent overfitting in the later stage of training. If the accuracy metrics of the model on the validation set, such as mAP, did not improve for 40 consecutive epochs, the training process would automatically terminate and save the historically best weights. Parallel loading was set to workers=4, which called four CPU threads for asynchronous data reading and preprocessing, avoiding idle GPU computing resources and improving the overall training throughput.

### 3.3.2 Built-in dynamic data augmentation strategy of YOLOv8

In addition to the offline instance-level Copy-Paste augmentation designed for the long-tail distribution described earlier, the YOLOv8 framework also integrates a highly efficient online dynamic data augmentation pipeline during the DataLoader stage. This mechanism is triggered in real time when images are read in each training epoch, ensuring that the image distribution seen by the model in each round is different. As a result, the generalization ability of the model in complex road environments is greatly improved. The core augmentation strategies include: (1) Mosaic augmentation; (2) dynamic disturbance in the HSV color space; (3) spatial affine transformation and random flipping; and (4) mask adaptive alignment.

## 3.4 Model Training Analysis

After training the model on the dataset collected in this study based on the YOLOv8-Seg architecture, the obtained performance evaluation curves are shown below.

### 3.4.1 Quantitative analysis of object detection performance based on bounding boxes

In Figure 5, the F1 curve shown in subfigure (a) reflects the evaluation performance of model classification confidence. This metric is composed of the harmonic mean of precision

and recall, with a value range from 0 to 1. The PR curve shown in subfigure (b) characterizes detection performance through the interaction between precision and recall, and the area under the curve is defined as the average precision (AP) metric. The P curve in subfigure (c) presents the response characteristics of precision as the category confidence threshold changes, with the horizontal axis representing the confidence threshold. The R curve in subfigure (d) describes the relationship between recall and the confidence threshold, comprehensively reflecting the detection completeness of the model under different threshold settings.

First, the model achieved high-precision convergence in spatial localization. According to the bounding-box P-R curve, the mean average precision of all-class detection, namely mAP@0.5, reached 0.889, indicating excellent overall performance. Among them, the mAP@0.5 values of the core business targets of the system, namely the bus lane and bus bounding-box detection, reached 0.990 and 0.989, respectively. The detection performance for cars, which represent the main non-bus traffic flow, also reached 0.961. This indicates that, relying on the previous data augmentation and mixed annotation strategy, the model achieved highly accurate position regression in the two-dimensional coordinate system, including coordinates  $x$  and  $y$  as well as width  $w$  and height  $h$ , without obvious bounding-box deviation.

Second, the optimal confidence threshold was determined. To provide a reliable filtering benchmark for downstream multi-object tracking algorithms, such as ByteTrack, this study analyzed the classification judgment threshold through the bounding-box F1-Confidence curve. The data show that when the confidence threshold was set to 0.365, the comprehensive F1 score of all categories reached its maximum value of 0.86. In future deployment for real bus-lane video stream inference, this value will be used as a baseline hyperparameter to achieve the optimal balance between the target interception rate, namely precision, and recall.

Third, the feature sparsity effect of distant small targets was observed. Regarding the weakness among detection categories, the bounding-box mAP@0.5 of motorcycles was 0.600. The physical reason for this phenomenon is that, under the perspective view of intersections, motorcycles not only occupy an extremely small

pixel area themselves, but their generated rectangular bounding boxes also inevitably contain a large amount of redundant background pixels, such as asphalt pavement and the edges of adjacent vehicles. This high background-noise ratio weakens the network's ability to represent small-target features, reflecting the inherent challenges of conventional rectangular-box detection in complex and dense traffic flows.

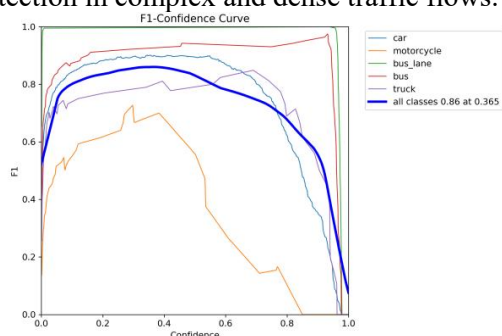


**Figure (a) F1 Curve**

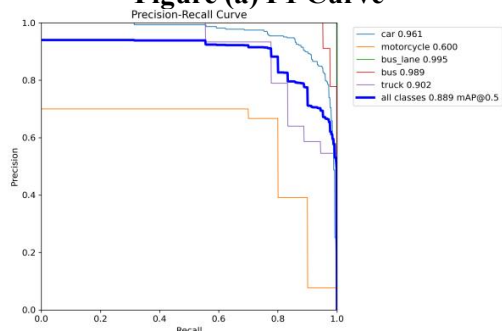


**Figure (b) PR Curve**

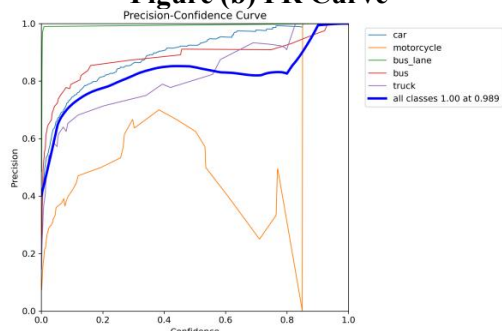


**Figure (c) P Curve**

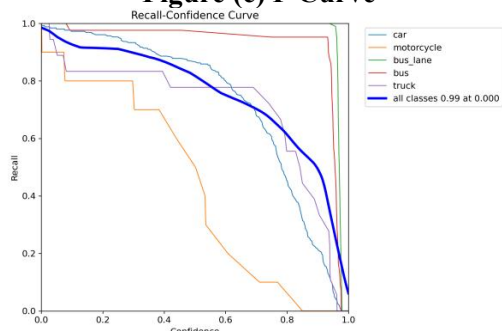


**Figure (d) R Curve**

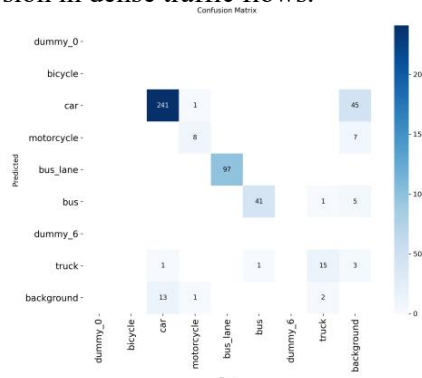
**Figure 5. Training Curves of the Bounding Box Model**

### 3.4.2 Confusion matrix

The confusion matrix intuitively reflects the classification accuracy of the model for each category, as well as the specific directions of misclassification. By analyzing the normalized confusion matrix shown in the figure 6, the following conclusions can be drawn.

First, there was no confusion for the core targets. The model exhibited extremely high feature discriminability for the core business categories of the system. Specifically, the prediction accuracy of bus lanes reached 100% (1.00), while the accuracy of buses reached 98% (0.98), with only 2% of buses being misclassified as trucks. This result strongly demonstrates that the integrated strategy of static feature dictionary mapping and background-isolated data augmentation successfully eliminated the negative impact of inter-class long-tail distribution, enabling the model to establish a highly clear decision boundary.

Second, traffic-flow recognition was accurate. Among non-bus vehicles serving as background traffic flow, cars, which constituted the category with the largest sample size, maintained a high accuracy of 95%. The accuracy of trucks was 83%, with the main error flows being misclassification as background (11%) or as buses (6%). This small amount of category confusion was mainly attributed to the similar geometric aspect ratios of large vehicles under distant perspective views and physical occlusion. Third, the limitations of small-target recognition were further verified. The classification accuracy of motorcycles was 80%, with some samples being misclassified as cars (10%) or background (10%). The distribution results of the confusion matrix are consistent with the weakness in motorcycle accuracy shown in the previously discussed P-R curve, further confirming, from the perspective of data flow, the physical limitation that small targets are prone to feature dispersion in dense traffic flows.



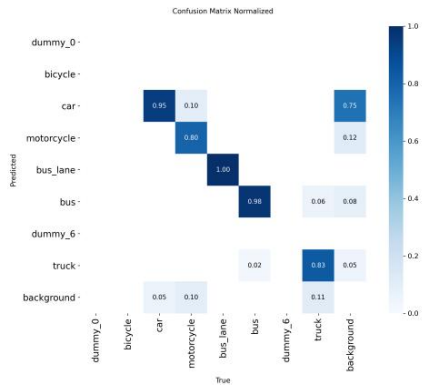


**Figure 6. Confusion Matrix**

### 3.4.3 Instance segmentation (mask)

In traffic violation judgment tasks, the pixel-level separation of irregular lane markings and overlapping vehicles constitutes a core technical challenge of the system. This study further evaluated the precise foreground extraction capability of the YOLOv8-Seg model in the instance segmentation (Mask) task. The relevant performance metric curves are shown in subfigures (e), (f), (g), and (h) of Figure 7.

First, the model demonstrated efficient representation of core edge features. According to the P-R curve, the mean average precision of all-class masks, namely  $mAP@0.5$ , reached 0.837. Among them, the pixel-level segmentation  $mAP@0.5$  values of the core business targets, namely the bus lane and bus, reached 0.995 and 0.989, respectively. These extremely high values indicate that the high-fidelity foreground generated through U<sup>2</sup>-Net extraction enabled the network not only to learn the rough positions of vehicles, but also to accurately capture high-dimensional spatial topological structures, such as the boundary between bus tires and the ground and the trapezoidal lane boundaries. This provides reliable pixel-level mask support for the subsequent and highly stringent lane-crossing judgment.

Second, mask generation confidence was calibrated. Compared with bounding-box regression, pixel-level classification is more sensitive to background noise. According to the F1-Confidence curve, the mask segmentation task achieved the optimal comprehensive F1 score of 0.81 when the confidence threshold was set to 0.417. In actual bus-lane monitoring deployment, setting the polygon mask generation threshold near this value can most effectively filter mask burrs caused by background noise, such as road-surface water reflections and tree shadows, thereby ensuring

the smoothness of the output contours.

Third, the physical challenges of the mask segmentation task were observed. The data show that the mask segmentation  $mAP@0.5$  of motorcycles was only 0.420, which was even lower than their bounding-box performance. From the perspective of algorithmic principles, motorcycles under distant surveillance views occupy very few pixels and have irregular contours. After downsampling through deep convolutional networks, their edge features are highly prone to dispersion. Moreover, pixel-level segmentation requires classification of every overlapping edge, which is far more difficult than generating an external bounding rectangle. This objectively confirms that fine-grained segmentation of small targets under dense occlusion remains a challenging problem in the current field of computer vision, and future work should introduce high-resolution feature preservation mechanisms for further improvement.

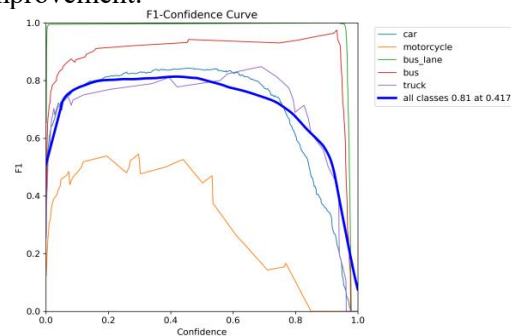


**Figure (e) F1 Curve**

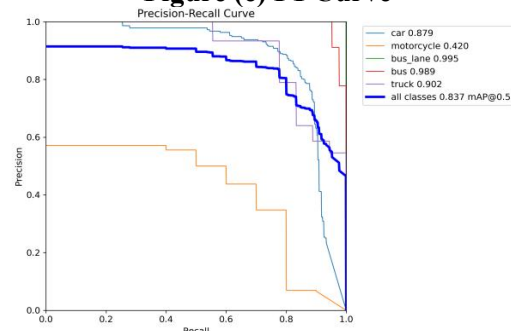


**Figure (f) PR Curve**

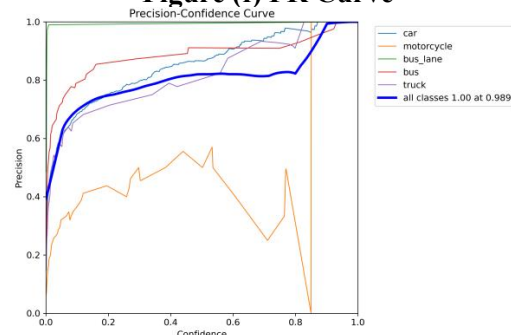


**Figure (g) P Curve**

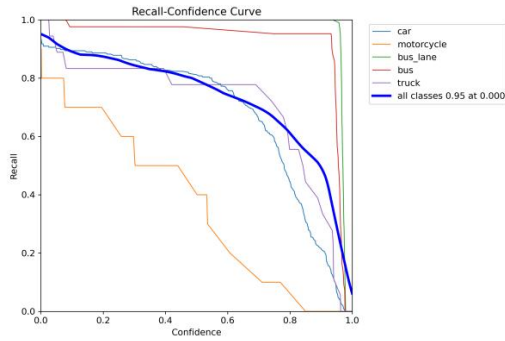


Figure (h) R Curve

### Figure 7. Training Curves of the Mask Model

#### 3.4.4 Analysis of the model training process and convergence

To verify the learning performance of the model and the rationality of the network parameter configuration, this study recorded and analyzed the variation curves of the loss functions and accuracy metrics of the YOLOv8-Seg model over 150 training epochs. The results are shown in Figure 8.

First, the convergence of the loss functions was analyzed. By observing the loss curves on the left side, it can be seen that, on the training set, the bounding-box regression loss (train/box\_loss), mask segmentation loss (train/seg\_loss), and classification loss (train/cls\_loss) all showed a smooth and continuous downward trend. This indicates that the network was stably and efficiently extracting

spatial and semantic features. On the independent validation set, represented by the validation curve group, all losses decreased rapidly in the early stage of iteration and gradually converged to a stable state after 50 epochs. No obvious rebound or divergence was observed in the validation loss curves, confirming that the data augmentation strategy introduced in this study effectively suppressed overfitting and that the model possessed excellent generalization ability. It should be noted that sem\_loss in the figure remained constantly equal to 0 because this study adopted instance segmentation rather than semantic segmentation, which is consistent with the expected behavior.

Second, the evolution of the evaluation metrics was analyzed. By observing the accuracy metric curves on the right side, it can be seen that the precision, recall, and core metric mAP@0.5 of object detection, denoted as B for Box, and instance segmentation, denoted as M for Mask, achieved a stepwise increase within the first 20 epochs, demonstrating the rapid learning rate of the model. As the training iterations progressed, all mAP metrics remained stable at high levels and eventually converged, with the mAP@0.5 values for both detection and segmentation staying within a high-performance range.

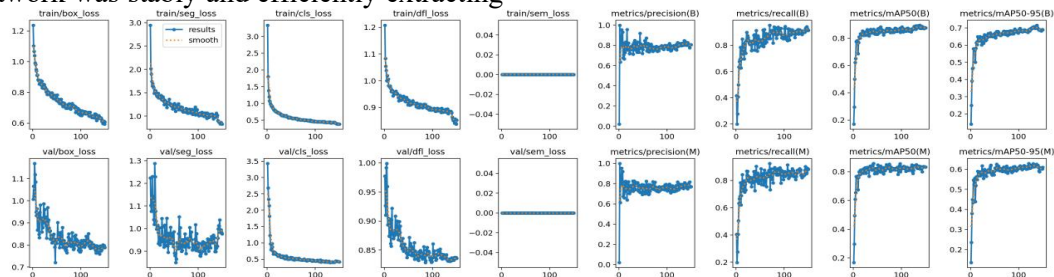


Figure 8. Training and Validation Curves of Various Losses



Figure 9. Original Image



**Figure 10. Training Result Visualization**

Based on the quantitative performance throughout the entire training process described above, the current setting of 150 training epochs and the hyperparameter configuration are highly reasonable. The final generated model weights, namely best.pt, achieved stable and excellent performance in both target localization and pixel-level foreground extraction tasks, fully meeting the practical requirements for real-time inference on subsequent traffic video streams. The visualization effects before and after training are shown in Figures 9 and 10.

#### 4. Vehicle Tracking, Violation Rule Judgment, and Experimental Result Analysis Based on ByteTrack

In Chapter 3, the system achieved accurate vehicle localization and pixel-level extraction of bus-lane regions within single-frame images through the YOLOv8-Seg algorithm. However, traffic violations, such as the illegal occupation of bus-only lanes, are not only spatial concepts but also dynamic events involving temporal continuity. Spatial lane crossing in a single frame alone cannot constitute a complete law enforcement process. Therefore, this chapter uses the ByteTrack multi-object tracking algorithm as the foundation, combines it with dynamic mask features, designs an occlusion-resistant lane-region extraction mechanism, establishes violation judgment rules based on a state machine, and finally develops an interactive GUI and data visualization module, thereby realizing a full-chain closed loop from the underlying algorithm to the upper-level application.

##### 4.1 ByteTrack Tracking Algorithm

In dense urban bus-lane scenarios, vehicles are

highly prone to mutual occlusion, causing the confidence scores of bounding boxes output by the object detector, namely YOLOv8, to drop sharply in certain frames. Traditional tracking algorithms, such as SORT, usually discard these low-confidence detection boxes directly, which leads to broken target trajectories and frequent identity ID switches. To address this problem, this system adopts the ByteTrack tracking algorithm proposed in 2021. As shown in Figure 11.

The core innovation of ByteTrack lies in its BYTE data association method. This method not only utilizes high-confidence detection boxes, but also effectively identifies targets from low-confidence results that may have been misjudged due to occlusion or image blurring, thereby improving tracking continuity. The BYTE algorithm achieves robust target tracking through two independent data association processes.

First, the first data association process, namely high-score box matching, is performed. ByteTrack first divides all detection boxes in the current frame into high-confidence boxes and low-confidence boxes according to a confidence threshold. In the first association stage, the algorithm uses Kalman filtering to predict the positions of existing trajectories in the current frame and calculates the intersection over union (IoU) similarity between these predicted trajectories and the high-confidence detection boxes. The Hungarian algorithm is then used to achieve globally optimal matching. After matching, the unmatched high-confidence boxes, existing trajectories, and all low-confidence boxes enter the next stage.

Second, the second data association process, namely low-score box recall, is conducted. In

this stage, the algorithm calculates the IoU similarity between the low-confidence boxes and the tracking trajectory boxes that failed to match in the first stage, and then performs matching again using the Hungarian algorithm. This mechanism can effectively filter background noise and recover real targets that are partially occluded.

Third, after the two rounds of association, low-confidence detection boxes that are not matched with any trajectory are directly deleted. High-confidence detection boxes that are not matched with existing trajectories are regarded as new targets, for which new trajectories are created and assigned new IDs. For trajectories that still fail to match any detection box, the system assumes that the corresponding targets are either severely occluded or have moved out of the frame, and therefore retains them for silent prediction over 30 frames. If no matching detection box is found after 30 frames, the trajectory is completely removed by the system. By fully utilizing information from low-confidence boxes, ByteTrack effectively solves the problem of trajectory loss caused by occlusion or target blurring, greatly reduces trajectory fragmentation, and significantly improves both tracking accuracy and robustness.



**Figure 11. Comparison of Tracking Performance Between ByteTrack and Other Trackers**

## 4.2 Violation Recognition and Judgment Rules

### 4.2.1 Spatial boundary localization of bus-only lanes

To determine whether a vehicle has violated traffic regulations, the primary task is to clearly define the physical boundary of the “restricted violation zone.” In terms of engineering implementation, this system provides two flexible lane localization strategies to adapt to

different front-end hardware deployment scenarios.

First, the static global coordinate import method is used for fixed surveillance devices. At most real-world intersections, electronic police surveillance cameras are fixed through rigid installation. In such application scenarios, the bus-only lane in the image is an absolutely static physical facility. Therefore, the system supports direct import during the initialization stage. Specifically, polygon vertices are manually selected on a reference frame and imported into the program in the form of a static coordinate array, such as  $[(x_1, y_1), (x_2, y_2), \dots]$ . This method transforms lane delineation into a static global region of interest (ROI) mask, which does not consume any runtime inference computing power. As shown in Figure 12.

Second, the dynamic joint detection extraction method is used for non-fixed devices or patrol vehicles. If the system is deployed on mobile police vehicles or surveillance devices with slight camera-position deviations, the static coordinate method becomes invalid. In this case, the system can seamlessly integrate with the lane-vehicle dual-task joint detection architecture designed in Chapter 3. During forward propagation in each frame, YOLOv8-Seg dynamically outputs the real-time polygon mask of the bus-only lane. The system then converts this mask into a dynamic ROI, ensuring that the spatial structure of the bus-only lane used as the judgment benchmark is adaptively aligned with the video stream in real time. As shown in Figure 13.



**Figure 12. Manually Entered Coordinate Bounding Frame**

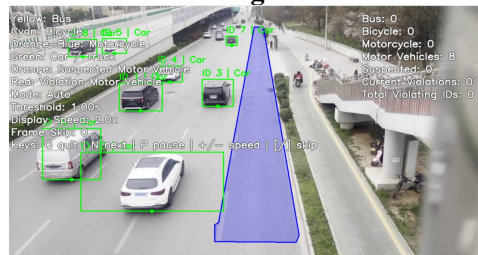


**Figure 13. Dynamically and Automatically Recognized Bounding Frame**

#### 4.2.2 Violation rule logic and judgment model

Violation judgment requires the integration of spatial and temporal information. According to relevant road traffic safety regulations, non-bus vehicles are prohibited from entering bus-only lanes during the designated control periods, namely 7:00–9:00 and 17:00–20:00. If a vehicle temporarily borrows the lane due to special circumstances, such as avoiding obstacles, it must leave the lane promptly. Based on this, the rule-based judgment state machine designed in this system includes the following core logic.

First, vehicle category verification is performed. The category of the tracked target is filtered. If the target belongs to the bus category, it is directly allowed to pass. If the target is a non-bus vehicle, such as a car or truck, it enters the collision detection stage. Meanwhile, the bounding boxes of different vehicles and the bus-only lane are displayed in different colors to avoid confusion.

Second, spatial collision detection is conducted to determine whether the vehicle crosses the lane boundary. Traditional judgment methods often use the intersection over union (IoU) between the target detection box and the lane area. However, perspective errors are likely to occur under different camera pitch angles. Therefore, this system adopts the bottom-center projection method, in which the midpoint of the bottom edge of the vehicle detection box, denoted as  $((x, y))$ , is extracted as the actual ground contact point of the vehicle. The ray casting method is then used to determine whether this contact point is strictly located inside the bus-lane polygon, thereby accurately judging whether the vehicle has crossed the boundary or occupied the lane. As shown in Figure 14.

##### (1) Bottom-center projection method

The spatial representation of targets output by current mainstream object detection algorithms, such as the YOLO series, is usually a two-dimensional external bounding box. If the geometric center of the bounding box is directly selected as the vehicle position reference, serious perspective errors may occur. This is because large vehicles, such as trucks and buses, have relatively high bodies, and under oblique-view projection, the center point of their two-dimensional bounding boxes often shifts significantly upward in the image coordinate system. Such perspective errors caused by the height dimension may easily lead to severe misjudgment, where the vehicle tires have not

yet crossed the lane line, but the center point appears to have “suspended” across the boundary.

Therefore, this study adopts the bottom-center projection method to approximate the three-dimensional spatial position of the target in two dimensions. This method discards the height feature of the target and extracts the midpoint of the bottom edge of the detection box as the true physical ground-contact anchor point of the vehicle. Let the coordinate set of the external bounding box of the (i)-th vehicle target in the image be  $B_i = (x\{min\}, y\{min\}, x\{max\}, y\{max\})$ . Then, the equivalent ground-contact point coordinates  $P_i(X\{c\}, Y\{b\})$  of the vehicle are defined as follows:

$$X = (xmin + xmax) \quad (2)$$

$$Y = ymax \quad (3)$$

This geometric estimation model is physically equivalent to the projection of the contact point between the vehicle's front or rear wheels and the ground plane. It fundamentally filters out coordinate drift caused by the projection of three-dimensional object height onto a two-dimensional plane and provides a high-precision spatial anchor point for subsequent lane-crossing judgment.

##### (2) Ray Casting Method

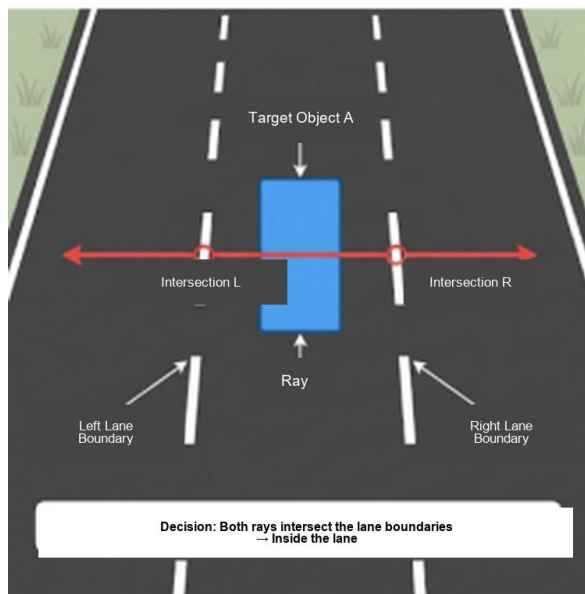
After obtaining the high-precision physical ground-contact point ( $P_i$ ) of the vehicle and extracting the dynamic irregular polygon contour ( $\Omega$ ) of the bus-only lane in the preceding module, the physical judgment of whether the vehicle crosses the lane line can be abstracted as a classical computer graphics problem: the point-in-polygon (Point-in-Polygon, PIP) topological inclusion test.

Considering that bus-lane polygons in road networks are often affected by occlusions and may present non-convex or even highly complex geometric characteristics, traditional IoU-threshold methods have accuracy limitations. Therefore, this system adopts the ray casting algorithm, which has strong universality and high computational efficiency, to solve this topological relationship.

The core mathematical logic of the ray casting method is based on the odd-even rule derived from the Jordan curve theorem. The specific judgment process is as follows. First, a test ray is constructed. Taking the vehicle ground-contact point ( $P$ ) to be tested as the starting point, an infinitely long one-way ray ( $R$ ) is drawn in an

arbitrary fixed direction in the image coordinate system. In this system, the positive direction of the (X)-axis is used. Second, edge intersections are counted. All boundary line segments of the bus-lane polygon ( $\Omega$ ) are traversed, and the total number of valid intersections between ray (R) and these boundary segments is calculated and denoted as (N). Third, the topological relationship is determined. According to the Jordan curve theorem, the parity of the total number of intersections (N) is checked. If  $N(\text{mod}2)=1$ , namely (N) is an odd number, it indicates that the ray penetrates from the interior of the polygon to the exterior. In this case, point (P) is strictly located inside polygon ( $\Omega$ ), and the system determines that the vehicle has physically entered the bus-only lane. If  $N(\text{mod}2)=0$ , namely (N) is an even number, it indicates that the ray enters and exits the polygon successively. In this case, point (P) is located outside polygon ( $\Omega$ ), and the system determines that the vehicle is in a lawful driving state.

#### Example of In-Lane Determination



#### Example of Out-of-Lane Determination

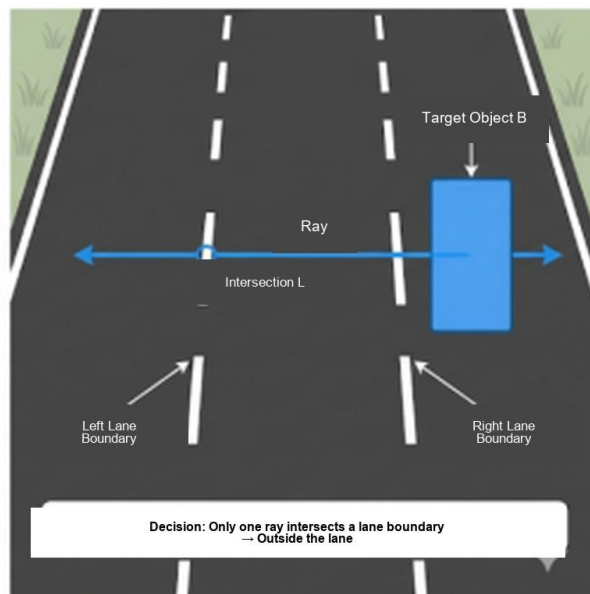


**Figure 14. Schematic Diagram of Determining Bus-Lane Occupancy Using the Ray Casting Method**

In the engineering implementation, this study uses the OpenCV vision library to achieve high-speed parallel computation of the ray casting method through low-level optimized floating-point multiply-accumulate operations, ensuring zero-delay response of the state-machine judgment logic under dense traffic-flow scenarios.

Third, temporal accumulation judgment is performed. To prevent instantaneous misjudgment caused by lane-line edge detection errors and to lawfully exclude short-term behaviors such as normal temporary lane borrowing or lane crossing, the system constructs a temporal state machine based on the unique vehicle IDs assigned by ByteTrack. A frame-count judgment threshold is introduced:

$$T_{\text{frames}} = \text{FPS} \times t_{\text{violation}} \quad (4)$$

(where the legally  $t_{\text{violation}}$  defined duration for determining a violation is 1 second). Vehicle behavior is abstracted into three finite states:

- 1) Normal: The vehicle is traveling in an ordinary lane.
- 2) Suspected: The vehicle's ground contact point enters the bus-only lane, the internal timer (t) starts accumulating, and the target tracking box turns orange as an early warning.
- 3) Violation: If and only if the number of consecutive frames in which a certain vehicle ID remains within the ROI exceeds the  $T_{\text{frames}}$  threshold, that is, when the accumulated time exceeds the preset threshold, the system triggers a violation alert and captures evidence. The system records the ID in the database and automatically extracts a violation evidence chain, including the high-definition original image and a cropped close-up image of the vehicle, for subsequent storage.

### 4.3 Implementation of the Bus-Lane Violation Capture System

On the basis of the principles described above,

this study integrates deep learning models with traditional computer vision algorithms to develop a complete monitoring system for capturing bus-lane violations. This section introduces the system development environment, major functional modules, core logic for lane extraction and violation judgment, as well as the implementation process of the final graphical user interface and data visualization modules.

#### 4.3.1 Development and runtime environment configuration

To ensure smooth system operation and code maintainability, the development and runtime environment of the system was configured as follows. Visual Studio Code (VS Code) was used as the programming software, and Python 3.9 was adopted as the programming language. In terms of the deep learning framework, PyTorch was used, with CUDA configured to enable GPU acceleration for model inference. The Ultralytics library was used to load the trained YOLOv8-Seg model weights. For computer vision processing, OpenCV was used to read video streams, crop and resize frame images, smooth polygon contours, and draw detection boxes and text prompts on video frames. For interface development and data processing, Python's built-in Tkinter library was used to develop the graphical user interface (GUI), Matplotlib was used for data analysis and statistical chart generation, and the JSON and CSV libraries were used to record violation logs.

#### 4.3.2 Interactive system design and Graphical User Interface (GUI)

To lower the operational threshold in practical engineering applications and improve system usability, this system adopts the Tkinter framework to build a lightweight interactive application hub. This module serves as the top-level entry point of the system, responsible for collecting user intentions and dispatching the underlying visual management interface. The parameter configuration module of the GUI supports dynamic configuration of multiple core hyperparameters. The configuration data are automatically persisted through the `settings_manager` module and serialized in JSON format, thereby avoiding repeated user input.

First, the input parameters include the following items: (1) video source, which supports either a single video file path or a folder path containing multiple videos; (2) output directory, which specifies the storage path for detection results, including evidence screenshots and log reports;

(3) model path, which loads the fine-tuned YOLOv8-Seg weight file, namely `best.pt`; (4) violation threshold, which sets the time threshold for determining how long a target vehicle remains across the lane boundary, measured in seconds; (5) inference size, which refers to the maximum side length of the input image during preprocessing and is used to balance inference accuracy and speed; and (6) control parameters, including playback speed and frame-skipping control, which allow users to accelerate playback or skip redundant frames during review.

Second, the monitoring system supports scenario-adaptive switching. To address the potential generalization bottleneck of AI models in complex traffic scenarios, the GUI introduces a single-choice fallback strategy for the detection mode, corresponding to the two bus-lane spatial boundary localization methods described above. In the AI automatic recognition mode, the system is fully controlled by the deep learning model, which automatically segments and extracts the bus-lane region from the video stream. In the manual selection mode, the system captures the first frame of the video and opens an ROI selection tool, requiring the user to manually outline the polygon vertices of the bus-only lane with the mouse, thereby providing a strong manually defined prior boundary for the algorithm.

Third, the system provides output and feedback functions. Specifically, the GUI embeds a scrolling text box for real-time logs, asynchronously printing the processing progress, operating mode, and potential abnormal states of the current video.

#### 4.3.3 Implementation of the main model algorithm and core business logic

The main model module is the core business execution component of the system, carrying the complete computational logic from pixel-level feature extraction to temporal state evolution. Its core workflow is as follows.

First, preprocessing and model inference are performed in the input stage. The system uses OpenCV's `VideoCapture` to read the video stream frame by frame. After each frame is extracted, `resize_for_inference` is first called to normalize it to the preset network input size, such as 960 px. The processed frame is then input into the instantiated YOLO tracking engine, namely `track_model.track`. The system configures `bytetrack.yaml` as the tracker configuration file and filters the results to retain

only specific classes, namely `classes=TARGET_CLASSES`. To improve the fineness of lane edges, the `retina_masks=True` parameter is enabled during inference.

Second, bus-lane establishment is performed, which constitutes the core innovation of this module. The system executes different geometric construction logic according to the `detection_mode` identifier passed from the GUI. In automatic mode, the output is a dynamic spatiotemporal polygon. For the mask matrix corresponding to category ID 4, namely the bus lane, output by the AI model, the system rescales it back to the original image size. To eliminate the jagged effect of edge pixels, `cv2.approxPolyDP` is introduced for polygon smoothing. Furthermore, to address physical occlusion, a spatiotemporal mask fusion algorithm is adopted. A `global_lane_mask` memory matrix is established, and the visible lane pixels in each frame are accumulated using a logical OR operation. Over time, the system dynamically “grows” the most complete lane contour. In manual mode, the system directly intercepts and uses the quadrilateral manually drawn by the user during the GUI stage as the only spatial benchmark.

Third, violation rule judgment is conducted. The system traverses all non-bus vehicles assigned with Track IDs. For spatial projection judgment, the bottom-center point of the detection box is extracted, and `cv2.pointPolygonTest` is called to determine whether this ground-contact anchor point lies inside the constructed lane polygon. For temporal state transitions, the system maintains a `track_state` dictionary in memory. If the ground-contact point of a vehicle is located inside the lane, its internal timer `inside_time` starts accumulating. Once the vehicle leaves the lane, the timer is reset to zero. When `inside_time` reaches the violation threshold set in the GUI, the state machine transitions to `Violation=True`.

Fourth, evidence output is performed. When a violation state is confirmed, the system immediately calls the `save_violation_evidence` function and generates three types of key outputs on the local hard disk: (1) evidence images, including a panoramic video frame highlighted with a red box and a cropped close-up image of the violating vehicle; and (2) structured logs, in which the vehicle ID, timestamp, vehicle type, residence duration, confidence score, and evidence image path are appended to `violation_log.csv`.

#### 4.3.4 Runtime monitoring visualization

To enable monitoring personnel to grasp the traffic situation at intersections in real time and verify the accuracy of system judgments, this system designs a real-time monitoring visualization window with high information density and immediate feedback during video stream processing. This window not only renders the original video image, but also overlays rich AI inference processes and business status information. As shown in Figure 15.

First, bus-lane visualization is implemented. After the system starts and enters either automatic or manual detection mode, the visualization window first establishes the spatial benchmark for violation judgment. A semi-transparent bus-lane layer is used, in which the dynamically extracted or manually selected bus-lane polygon region is rendered as a blue semi-transparent overlay on the current video frame. This design does not block the visual details of traffic participants in the underlying image, while clearly delineating the “restricted violation zone.” To further enhance boundary perception, the edge of the bus-lane polygon is outlined with a solid blue line, intuitively displaying the processing effect of the polygon smoothing anti-aliasing algorithm and the spatiotemporal fusion boundary.

Second, target tracking and status visualization are implemented. The system tracks all key targets in the video frame in real time and visually reflects the current status of each target through rectangular detection boxes and text labels in different colors. For target detection box localization, each captured vehicle is drawn with a rectangular box closely fitting its body. A solid circular point is drawn at the midpoint of the bottom edge of the vehicle, namely the equivalent ground-contact point, to visualize the anchor position of the bottom-center projection method described above. For multidimensional attribute labels, the core attributes of the vehicle are displayed above the upper-left corner of the detection box, including the unique tracking ID, vehicle category, and residence duration.

Third, vehicle traffic behavior status visualization is implemented. This system concretizes the violation judgment rules into four visual color states, mapping vehicle traffic behavior in real time. For normal bus passage, shown in yellow, when the system identifies a bus, its detection box and label are rendered in yellow regardless of whether it is inside the bus

lane. The label is displayed as “Bus” or “Bus in Lane,” indicating that the vehicle has priority right-of-way. For normal passage of non-bus vehicles, shown in green, when a car or truck travels in an ordinary lane without touching the bus-lane boundary, its detection box is rendered in green, and the system does not start violation timing. For suspected illegal lane occupation, shown in orange, once the bottom-center anchor point of a non-bus vehicle crosses into the blue bus-lane region, the violation timer is triggered, namely  $\text{Inside Time} > 0$ , and the vehicle detection box instantly changes from green to orange. At this point, the label is updated to “ID X | Suspected Car | [current duration]/[threshold]s,” intuitively presenting the developing violation behavior. For confirmed lane-crossing violations, shown in red, when a suspected violating vehicle continuously remains in the bus lane longer than the preset threshold, such as 1.0 s, the state machine transitions to confirmed violation. At this time, the vehicle detection box turns red, the label is fixed as “ID X | Violation Car | [total duration]s,” and the underlying automatic screenshot and log-writing mechanism is triggered. In addition, the system also performs auxiliary tracking and classification display for other road participants, including non-motor vehicles such as bicycles, shown in cyan, and motorcycles, shown in orange-blue, to present a complete traffic ecosystem of the road network.

Fourth, statistical data panel visualization is implemented. To help operation and maintenance personnel quickly evaluate the current concurrent traffic volume, the system renders two groups of on-screen display (OSD) text panels in the upper-left area of the image. The left legend and status panel dynamically displays the current operating parameters, including mode, such as Auto or Manual, threshold, such as 1.0 s, playback speed, and number of skipped frames, along with a color legend guide and shortcut key instructions, such as pressing P to pause and using +/- to adjust speed. The right traffic statistics panel updates in real time by scrolling and displays the number of each vehicle type in the current frame, as well as the current numbers of suspected violations and confirmed violations. In addition, the system accumulates and counts the total number of violating vehicles captured in the current video, enabling monitoring personnel to obtain a clear macro-level understanding of road conditions

without waiting for the video processing to end.

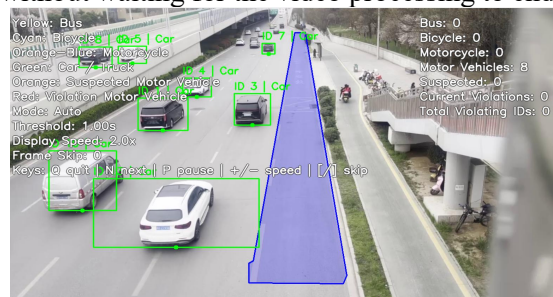


**Figure 15. Visualization Data Panel**

#### 4.3.5 Data Statistics and automatic visualization analysis

After the video batch-processing task is completed, the system uses the final overall visualization module to conduct in-depth mining and visual presentation of the summary.json and violation\_log.csv files output by the main model, with the aim of intuitively demonstrating the effectiveness of traffic governance.

First, data reading and integration are performed at the input stage. The system traverses all independent video folders under the general output directory, supports both new and old JSON storage path structures, and safely extracts the total number of non-bus vehicles, the total number of buses, the number of suspected violations, and the number of confirmed violations.

Second, chart rendering and output are performed. By combining rendering techniques with Chinese typesetting fonts, the system automatically generates a series of clearly structured and high-resolution data reports. These include: (1) a grouped bar chart comparing the number of violations across videos, which horizontally presents the absolute numerical transformation between suspected and confirmed violations at each checkpoint through physical coordinate interval arrangement; (2) a composite pie chart of the macro-level traffic composition, in which buses, law-abiding non-bus vehicles, and violating non-bus vehicles are plotted in a single pie chart. To emphasize the violating subjects, a visual explosion effect is applied to the violation sector with an extremely small proportion, and the specific quantity is directly displayed in the label; and (3) an independent bar chart showing the trend of the actual violation rate. By calculating the ratio of confirmed violations to the total number of passing non-bus vehicles, the absolute difference in average daily traffic volume is smoothed out, generating a ratio chart that accurately reflects

the actual compliance status at intersections.

#### 4.4 Experimental Results and Analysis

##### 4.4.1 Data preparation

Based on the visualization introduction of the entire monitoring system above, this study prepared video datasets from five intersections. Several videos were collected for each intersection, with each video lasting approximately 4 to 5 minutes. A total of 22 video datasets were used. Among them, 20 videos were captured by fixed facilities, and these videos were monitored and recognized using manually drawn bounding regions. The remaining two videos were used for experiments based on automatic bus-lane recognition. As shown in Figure 16.

##### 4.4.2 Visualization result output

After the system processes a single video, it creates a dedicated result folder for that video to record vehicle violation behaviors. A corresponding vehicle capture interface and a table recording the relevant violation details are then generated. As shown in Figure 17, Table 1 and Table 2.



**Figure 16. System Recognition Case Based on Automatic Region Detection**



**Figure 17 Interface for Capturing Traffic Violations**

**Table 1. Record Table of Violating Vehicles in a Single Video**

| video name              | track id | timestamp sec | vehicle type | inside time sec | confidence | bbox x1 |
|-------------------------|----------|---------------|--------------|-----------------|------------|---------|
| VID_20260402_120650.mp4 | 30       | 4.447         | Car          | 1.016           | 0.3175     | 2610    |
| VID_20260402_120650.mp4 | 73       | 10.009        | Car          | 1.016           | 0.5889     | 2535    |
| VID_20260402_120650.mp4 | 123      | 36.965        | Car          | 1.002           | 0.5279     | 2544    |
| VID_20260402_120650.mp4 | 127      | 53.116        | Car          | 1.002           | 0.3611     | 2592    |
| VID_20260402_120650.mp4 | 30       | 64.038        | Car          | 1.016           | 0.5485     | 2533    |
| VID_20260402_120650.mp4 | 2        | 1.016         | Truck        | 1.033           | 0.7478     | 2163    |
| VID_20260402_120650.mp4 | 25       | 3.581         | Car          | 1.049           | 0.3248     | 2540    |
| VID_20260402_120650.mp4 | 2        | 0.999         | Truck        | 1.016           | 0.7752     | 2159    |
| VID_20260402_120650.mp4 | 30       | 3.531         | Car          | 1.016           | 0.2808     | 2536    |
| VID_20260402_120650.mp4 | 122      | 59.01         | Car          | 1.049           | 0.5977     | 2540    |
| VID_20260402_120650.mp4 | 128      | 59.16         | Car          | 1.049           | 0.9311     | 1886    |

**Table 2. Record Table of Violating Vehicles in a Single Video (Continued)**

| bbox_y1 | bbox_x2 | bbox_y2 | violation_path                                                                                              | crop_path                                                                                           |
|---------|---------|---------|-------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------|
| 814     | 2740    | 949     | E:/BusLaneSystem/output\VID_20260402_120650\violation_frames\VID_20260402_120650_ID30_4_45s_violation.jpg   | E:/BusLaneSystem/output\VID_20260402_120650\vehicle_crops\VID_20260402_120650_ID30_4_45s_crop.jpg   |
| 843     | 2644    | 925     | E:/BusLaneSystem/output\VID_20260402_120650\violation_frames\VID_20260402_120650_ID73_10_01s_violation.jpg  | E:/BusLaneSystem/output\VID_20260402_120650\vehicle_crops\VID_20260402_120650_ID73_10_01s_crop.jpg  |
| 847     | 2660    | 934     | E:/BusLaneSystem/output\VID_20260402_120650\violation_frames\VID_20260402_120650_ID123_36_97s_violation.jpg | E:/BusLaneSystem/output\VID_20260402_120650\vehicle_crops\VID_20260402_120650_ID123_36_97s_crop.jpg |
| 842     | 2694    | 934     | E:/BusLaneSystem/output\VID_20260402_120650\violation_frames\VID_20260402_120650_ID128_59_16s_violation.jpg | E:/BusLaneSystem/output\VID_20260402_120650\vehicle_crops\VID_20260402_120650_ID128_59_16s_crop.jpg |

|     |      |      |                                                                                                             |                                                                                                     |
|-----|------|------|-------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------|
|     |      |      | 60402_120650\violation_frames\VID_20260402_120650_ID127_53_12s_violation.jpg                                | 60402_120650\vehicle_crops\VID_20260402_120650_ID127_53_12s_crop.jpg                                |
| 840 | 2641 | 929  | E:/BusLaneSystem/output\VID_20260402_120650\violation_frames\VID_20260402_120650_ID30_64_04s_violation.jpg  | E:/BusLaneSystem/output\VID_20260402_120650\vehicle_crops\VID_20260402_120650_ID30_64_04s_crop.jpg  |
| 779 | 2523 | 1119 | E:/BusLaneSystem/output\VID_20260402_120650\violation_frames\VID_20260402_120650_ID2_1_02s_violation.jpg    | E:/BusLaneSystem/output\VID_20260402_120650\vehicle_crops\VID_20260402_120650_ID2_1_02s_crop.jpg    |
| 807 | 2699 | 967  | E:/BusLaneSystem/output\VID_20260402_120650\violation_frames\VID_20260402_120650_ID25_3_58s_violation.jpg   | E:/BusLaneSystem/output\VID_20260402_120650\vehicle_crops\VID_20260402_120650_ID25_3_58s_crop.jpg   |
| 778 | 2518 | 1116 | E:/BusLaneSystem/output\VID_20260402_120650\violation_frames\VID_20260402_120650_ID2_1_00s_violation.jpg    | E:/BusLaneSystem/output\VID_20260402_120650\vehicle_crops\VID_20260402_120650_ID2_1_00s_crop.jpg    |
| 806 | 2694 | 968  | E:/BusLaneSystem/output\VID_20260402_120650\violation_frames\VID_20260402_120650_ID30_3_53s_violation.jpg   | E:/BusLaneSystem/output\VID_20260402_120650\vehicle_crops\VID_20260402_120650_ID30_3_53s_crop.jpg   |
| 848 | 2676 | 951  | E:/BusLaneSystem/output\VID_20260402_120650\violation_frames\VID_20260402_120650_ID122_59_01s_violation.jpg | E:/BusLaneSystem/output\VID_20260402_120650\vehicle_crops\VID_20260402_120650_ID122_59_01s_crop.jpg |
| 839 | 2315 | 1152 | E:/BusLaneSystem/output\VID_20260402_120650\violation_frames\VID_20260402_120650_ID128_59_16s_violation.jpg | E:/BusLaneSystem/output\VID_20260402_120650\vehicle_crops\VID_20260402_120650_ID128_59_16s_crop.jpg |

After the system processed all 22 videos, the terminal output included the following results: (1) total number of processed videos: 22; (2) total number of bus detections: 2,335; (3) total number of non-bus vehicle detections: 152,933; (4) total number of suspected violations: 2,894;

(5) total number of confirmed violations: 67; and (6) overall actual violation rate: 0.04%. The summarized visualization images were output to the dedicated result summary folder. As shown in Figure 18, 19, 20 and 21.

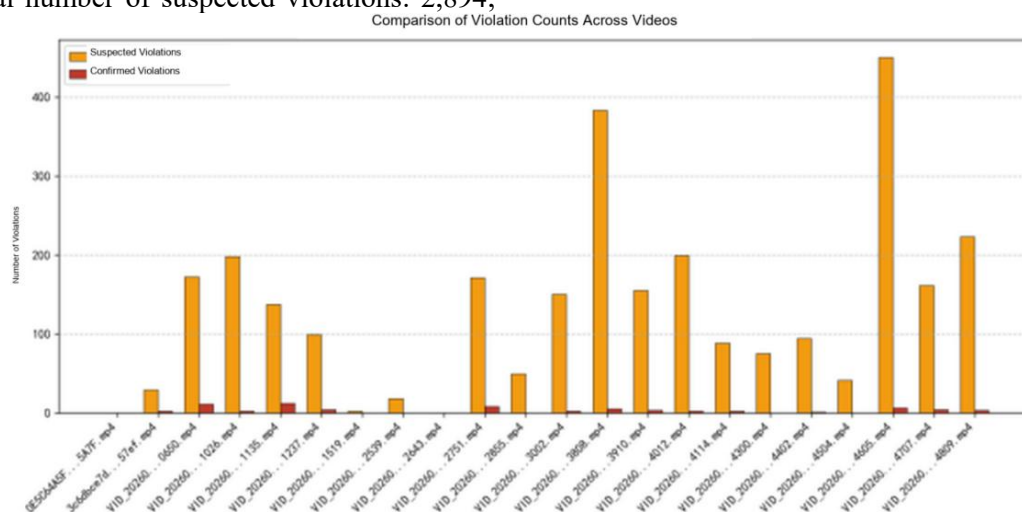


**Figure 18. Bar Chart Comparing the Numbers of Suspected and Confirmed Violations Across Videos**

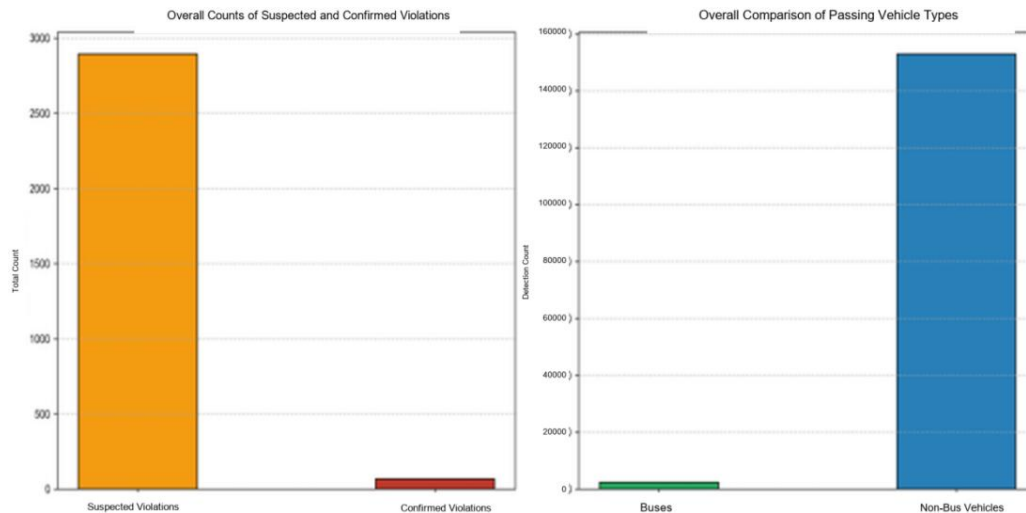


**Figure 19. Bar Chart Comparing Overall Suspected and Confirmed Violations with Overall Passing Vehicle Types**

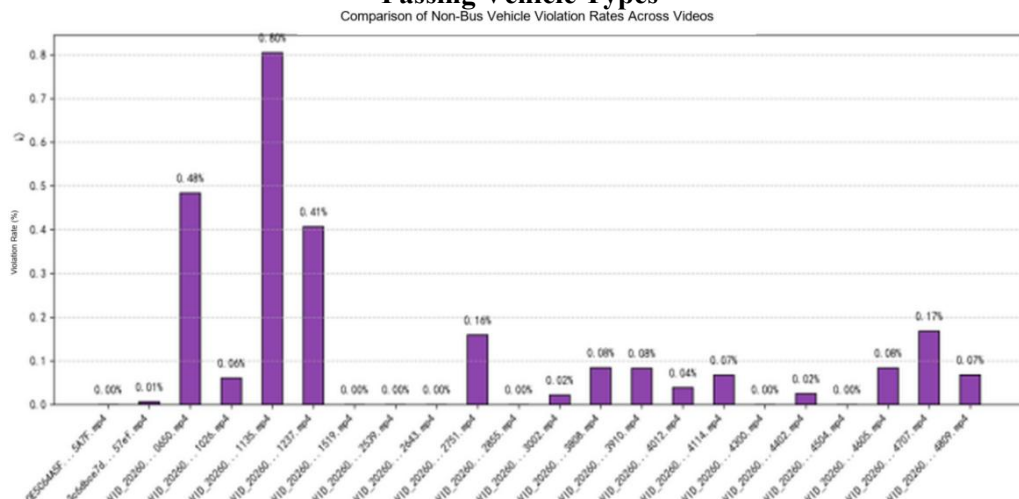


**Figure 20. Bar Chart Comparing Violation Rates of Non-bus Vehicles Across Videos**

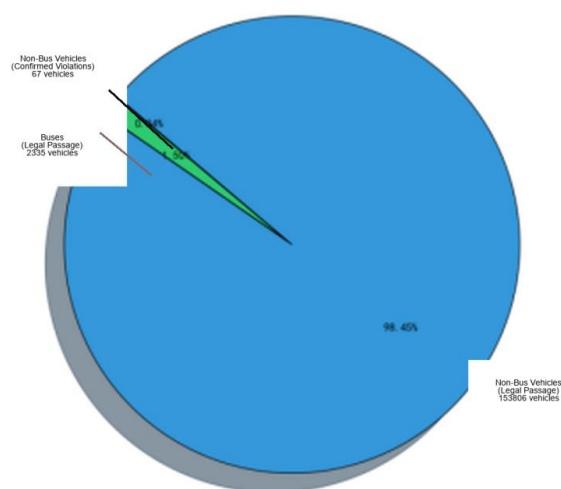


**Figure 21. Pie Chart of the Proportional Distribution of Vehicle Types and Violation Status**

#### 4.4.3 Manual verification

To verify the accuracy of the system in

monitoring and judging vehicle violations, manual inspection was conducted during video processing. Except for slight errors caused by insufficiently precise bus-lane recognition or bounding-region delineation, no obvious missed detections or false detections occurred. Overall, the system demonstrated good performance in video processing.

#### 4.4.4 Current system evaluation and limitations

First, regarding the overall system evaluation, this study designed and implemented a complete intelligent monitoring system for detecting illegal occupation of bus lanes. From the perspective of engineering implementation and practical testing performance, the system has several significant advantages. It establishes a full-chain workflow integrating visual perception, trajectory tracking, rule-based judgment, evidence output, and data visualization. The system not only verifies the theoretical algorithms, but also develops a

software hub with an interactive graphical user interface (GUI), forming an engineering prototype that can directly assist modern urban traffic law enforcement. By integrating advanced object detection algorithms and tracking algorithms currently available in the market, the proposed violation judgment rules demonstrate good accuracy.

Second, several deficiencies and limitations remain. Although the system performed well on the test data, it is still constrained by software and hardware conditions as well as the current algorithmic framework. In the construction of more complex real-world intelligent transportation infrastructure, the following limitations remain.

(1) Strong dependence on the current environment and static cameras. Both the spatiotemporal mask fusion algorithm and the manually calibrated ROI mode in this system are based on the premise that the surveillance camera view remains absolutely static. In practical applications, once the camera undergoes zooming or rotation, the global memory matrix will become invalid. Additional image data must then be introduced for training. Moreover, the system depends heavily on the current single environment, and its monitoring performance may not remain equally effective when applied to other intersections.

(2) Generalization bottlenecks under extreme weather conditions. The YOLOv8-Seg instance segmentation model used in this study was mainly fine-tuned on image datasets collected under normal weather conditions. Under extreme visibility-obstructed conditions, such as heavy rain, snow covering lane markings, or strong glare from oncoming vehicle high beams at night, the accuracy of lane-mask extraction by the AI model may decrease significantly. Although the system can be downgraded to manual mode through the GUI in such cases, its degree of automation is restricted.

(3) Risk of ID switching under extreme congestion. ByteTrack performs well under normal traffic-flow conditions. However, in extreme scenarios caused by morning and evening peak congestion, where vehicles are densely queued and closely adjacent to each other, target detection boxes are highly prone to overlap. In such cases, the predicted trajectories generated by Kalman filtering may intersect, resulting in Track ID switching for a small number of vehicles and consequently

interrupting or resetting the residence-time counter.

(4) Singularity of the violation judgment rules. The current traffic state machine mainly relies on the static rule of “spatial lane crossing plus time threshold.” However, in real complex traffic scenarios, non-bus vehicles may occupy bus lanes to avoid ambulances or fire trucks performing emergency tasks, or to bypass traffic accidents ahead.

## 5. Conclusion

### 5.1 Summary of the Main Research Work

This paper conducts an in-depth study on the core topic of identifying illegal occupation of bus lanes by motor vehicles. With the rapid development of artificial intelligence technology, traditional traffic monitoring methods can no longer fully meet the increasingly complex demands of urban traffic governance. Based on previous studies, this paper integrates advanced computer vision algorithms with practical traffic management requirements. The main research work completed in this study is summarized as follows.

First, the theoretical foundation of the algorithms was established. At the initial stage of the study, this paper systematically reviewed and compared the development history and underlying mathematical theories of advanced computer vision algorithms, including lane-line recognition, object detection such as the YOLO series, and multi-object tracking such as ByteTrack. This laid a solid theoretical foundation for subsequent algorithm selection and system architecture design.

Second, a visual perception model was constructed. For the object detection task, this paper focused on real-world scenarios and collected a local traffic surveillance dataset from several intersections in Xi'an. On this basis, official pre-trained weights were loaded, and transfer learning and fine-tuning were carried out using the YOLOv8 algorithm. The perception model can not only accurately locate various types of traffic participants, but also effectively extract bus-lane regions through instance segmentation, thereby providing a reliable spatial benchmark for violation judgment.

Third, a spatiotemporal violation judgment logic was formulated. Image detection alone cannot define a “violation.” Therefore, this paper

introduced the ByteTrack algorithm to achieve continuous spatiotemporal tracking of vehicles. Furthermore, the bottom-center projection method, the ray casting method based on point-in-polygon topological inclusion, and the time-threshold state machine were innovatively integrated to construct a complete set of violation recognition and judgment rules. This effectively filters false alarms caused by perspective errors and lawful lane-changing behavior.

Fourth, a complete intelligent monitoring system was developed and implemented. Based on the above theories and algorithms, this paper designed and implemented an intelligent monitoring system for illegal bus-lane occupation, transforming the proposed method from theory into practical application. The system includes a user-friendly graphical user interface, a dynamic visualization window with anti-aliasing and occlusion-resistant processing, and the ability to output detailed violation evidence logs and multidimensional statistical analysis charts. This study not only completes the cross-validation of theoretical algorithms, but also builds an engineering prototype with the potential to directly assist modern urban traffic law enforcement.

## 5.2 Limitations and Future Prospects

Although the system designed in this paper achieved good recognition performance and engineering robustness in the test scenarios, this study still has several limitations due to constraints in software and hardware testing conditions, dataset scale, and the existing algorithmic framework. These limitations also indicate directions for future research.

First, the dataset scale and scenario generalization capability remain limited. During model training and system testing, the dataset used in this paper mainly relied on surveillance videos collected from specific intersections and specific camera angles in Xi'an. As a result, when the model is applied to bus lanes with different forms in other cities or to static or dynamic PTZ cameras installed at different heights, its environmental robustness and feature generalization capability may be insufficient. In the future, efforts will be made to collect large-scale traffic datasets across regions, viewing angles, and time periods. Unsupervised techniques will also be introduced so that the model can quickly adapt to traffic monitoring

nodes in new cities without requiring large-scale re-annotation.

Second, the violation judgment rules lack global contextual awareness. The current violation state machine mainly relies on the static and rigid rule of "spatial lane crossing plus time threshold." However, in real complex traffic scenarios, non-bus vehicles may occupy bus lanes to avoid special vehicles performing emergency tasks, such as ambulances and fire trucks, or to bypass traffic accidents ahead. The current system does not yet possess the cognitive ability to exempt such special cases. In the future, intelligent transportation systems should evolve toward a cognitive paradigm. By constructing spatiotemporal semantic graph networks, the system will not only track vehicle coordinates, but also analyze vehicle interactions and scene context. This will enable the system to autonomously distinguish between active malicious lane occupation and passive emergency lane borrowing, thereby achieving more humanized and precise intelligent traffic law enforcement.

## References

- [1] Chen Meng. Research on a violation detection algorithm for lane occupation based on in-vehicle video footage [D]. Huazhong University of Science and Technology, 2015.
- [2] Dawood B, Mujtaba Z, Akbari P, et al. Automated enforcement of bus-only lanes and crossings with policy and implementation insights from a calgary case study[J]. *Case Studies on Transport Policy*, 2025: 101630.
- [3] Sun Xiyun. Recognition of motor vehicles illegally using bus lanes based on deep learning [D]. China People's Public Security University, 2025
- [4] Li Zhiyuan, Wang Guanghui. Research on Lane Line Recognition Based on Traditional Edge Operators [J]. *Modern Electronic Technology*, 2024,47(07):61–65.
- [5] Ge Z, Ma C, Fu Z, et al. End-to-end lane detection with convolution and transformer[J]. *Multimedia Tools and Applications*, 2023, 82(19): 29607-29627.
- [6] Ren S, He K, Girshick R, et al. Faster r-cnn: Towards real-time object detection with region proposal networks[J]. *Advances in neural information processing systems*, 2015, 28.

- [7] JOCHER G, CHAURASIA A, QIU J. Ultralytics YOLOv8: version 8.0.0[CP/OL]. 2023[2026-07-24]. <https://github.com/ultralytics/ultralytics>.
- [8] Zhang Y, Sun P, Jiang Y, et al. Bytetrack: Multi-object tracking by associating every detection box[C]//European conference on computer vision. Cham: Springer Nature Switzerland, 2022: 1-21.
- [9] GB 5768.1-2025: Road Traffic Signs and Markings – Part 1: General Principles [S]. Beijing: China Standards Press, 2025.
- [10] Qin X, Zhang Z, Huang C, et al. U2-Net: Going deeper with nested U-structure for salient object detection[J]. Pattern recognition, 2020, 106: 107404.
- [11] Pan S J, Yang Q. A survey on transfer learning[J]. IEEE Transactions on knowledge and data engineering, 2009, 22(10): 1345-1359.
- [12] Lin T Y, Maire M, Belongie S, et al. Microsoft coco: Common objects in context[C]//European conference on computer vision. Cham: Springer International Publishing, 2014: 740-755.
- [13] Bolya D, Zhou C, Xiao F, et al. Yolact: Real-time instance segmentation[C]//Proceedings of the IEEE/CVF international conference on computer vision. 2019: 9157-9166.