

# Comparative Analysis of YOLOv8 and HSV Methods for Traffic Density Measurement

<sup>a1</sup>I Gede Pasek Suta Wijaya, <sup>a2</sup>Muhamad Nizam Azmi, <sup>a3</sup>Ario Yudo Husodo

<sup>a</sup>Jurusan Sistem Komputer dan Informatika, Universitas Mataram  
Mataram, Indonesia

<sup>1</sup>[gpsutawijaya@te.ftunram.ac.id](mailto:gpsutawijaya@te.ftunram.ac.id)

<sup>2</sup>[muhamadnizamazmi@gmail.com](mailto:muhamadnizamazmi@gmail.com)

<sup>3</sup>[ario@unram.ac.id](mailto:ario@unram.ac.id)

## Abstract

Traffic density measurement is a critical component in traffic management and urban planning. This study addresses the challenge of accurately measuring traffic density by comparing the performance of the YOLOv8 segmentation method with the traditional HSV method. At the beginning of the abstract, we clearly present the problem of accurately measuring traffic density. The primary objective is to highlight the strengths and limitations of each method in terms of accuracy and reliability in traffic density estimation. The choice of segmenting the asphalt area rather than vehicle objects is justified by the need to understand how different segmentation approaches affect traffic density measurements. The HSV method involves converting images to the HSV color space, creating masks for specific areas, and measuring traffic density based on the asphalt area. This method, while straightforward, may not accurately capture the dynamic nature of vehicle movement. In contrast, the YOLOv8 segmentation method utilizes a deep learning approach to detect and segment vehicles, providing potentially more precise measurements. Experimental results from three locations demonstrate varying levels of traffic density. The YOLOv8 method results in a graph with a wavy pattern, reflecting the more detailed detection of vehicles. Conversely, the HSV method produces a linear pattern, indicating a more consistent but potentially less detailed measurement. Quantitative analysis shows that Location 2 has a higher traffic density compared to Locations 1 and 3, as indicated by the average number of detected vehicles per frame. This study provides a comprehensive understanding of the differences between HSV and YOLOv8 segmentation methods for traffic density measurement. The findings suggest that while YOLOv8 offers more detailed and dynamic detection, the HSV method provides a simpler yet effective approach for certain applications.

**Keywords:** Computer Vision, Traffic Density, Vehicle Detection, YOLOv8 Segmentation Method, Normal HSV Method, Traffic Patterns.

## 1. Introduction

With the continuing increase in urbanization and people's mobility, traffic density analysis becomes crucial in urban transportation management. Traffic congestion occurs where, at certain times, vehicles traveling along a road segment stop for short or long periods. Congestion is evidence of the disorder in traffic management that occurs in urban areas, but congestion is not a new phenomenon. Almost all large cities, whether in developed or developing countries, still face congestion problems, at least during busy morning and evening hours [1][2]. The development of the number of vehicles in Indonesia in the years 2022 amounted to 151,424,276 units. Then, in January 2023, it reached 152,565,905 units, with the data sourced from the Traffic Corps of the Indonesian National Police [3][4]. However, traffic density does not only reflect the level of congestion on the road but is also influenced by the width of the road. In fact, the density level reflects the precision between the width of the road and the increase in the population, leading to an increase in the number of vehicles in that area[5]. Traffic density can lead to congestion, air pollution, accidents, and economic losses [6][7][8]. As the number of vehicles increases every year, especially in Indonesia, there is an increase of approximately 0.75% in the

number of vehicles in January 2023. However, the width of the road has not significantly increased, leading to traffic congestion. To address this issue, a traffic engineering strategy needs to be formulated to prevent congestion from worsening. Traffic engineering requires data related to road and vehicle conditions, including knowing the level of density on a road segment over time[9].

Currently, to count the number of vehicles passing through a road segment, manual recording of each type of vehicle passing by is done, or other methods such as installing sensors on the road that can detect vehicles based on pressure, infrared, ultrasonic, or other types of sensors are used. With the development of artificial intelligence, computer vision, and the affordability of cameras, the method of detecting and counting the number of vehicles has become more sophisticated. One effective method is to use vision-based sensors that can provide more information compared to other sensors[10][11]. However, to detect vehicles from video captured by cameras, sophisticated processing is required because the types of vehicles captured by cameras vary in shape, color, and perspective[12][13]. The quality of life for residents can be disrupted by traffic congestion, violations, and accidents[14][15]. Therefore, the approach of asphalt detection using computer vision emerges as a potential solution to improve the accuracy of traffic density analysis, especially at street intersections for further decision-making. Asphalt, as a dominant element on the road surface, can be used as the main reference for measuring traffic density. By integrating computer vision technology, we can identify and monitor changes in road conditions in more detail, especially when asphalt is not visible due to the presence of objects or vehicles[16].

From the several studies described earlier, it is evident that several effective methods such as CNN[17][18], YOLOv4[19], XSG Blockset[20], HOG[21][22], SAD[23], and even color segmentation approaches like HSL and HSV are used for real-time road density monitoring[24][25]. However, none of the previous studies have used an approach with asphalt detection as a parameter for measuring traffic density. Therefore, among the algorithms used for traffic density detection, the author proposes a method for monitoring traffic density with an asphalt detection approach. This study aims to develop a traffic density analysis system with the assistance of YOLOv8 for object segmentation detection and asphalt with an asphalt detection approach as a parameter to measure traffic density[26]. By doing this, it is hoped that it can make a significant contribution to the understanding and management of urban traffic. In the context of traffic density analysis using computer vision, the research problem formulation is how to implement image processing methods, such as edge detection and segmentation, to calculate the length and width of asphalt from images, as well as how to effectively detect asphalt considering factors such as color, texture, and filters in image processing. Additionally, this research also aims to utilize video processing techniques with optical flow algorithms to process videos in real-time.

You Only Look Once (YOLO) is one of the single-stage object detection methods, along with SSD (Single Shot Detector). Using YOLO, object detection is treated as a regression problem to spatially separate bounding boxes and the associated class probabilities for those bounding boxes. A neural network predicts bounding boxes and class predictions directly from the entire image in one evaluation. YOLO has several advantages compared to classifier-based systems. YOLO will produce prediction results with a single evaluation network unlike the R-CNN system which requires multiple networks to process one image. Therefore, YOLO is said to be thousands of times faster than R-CNN and hundreds of times faster than Fast R-CNN[27].

YOLOv8, or "You Only Look Once version 8," is the latest popular object detection model in the field of computer vision. With the "You Only Look Once" approach, YOLOv8 is able to detect objects by looking at an image only once, distinguishing it from other methods that require multiple stages. Its complex model architecture consists of various layers and is designed to improve object detection performance, including accuracy and speed. This model supports various object categories, such as humans, cars, and animals, and has been applied in various applications, including security surveillance, autonomous vehicles, and medical image analysis. YOLOv8 requires training on a suitable dataset and can be fine-tuned for specific purposes. As an open-source project, YOLOv8 allows contributions from the community, while its integration with modern technologies, such as GPU usage, enhances the efficiency of the inference process. It is

important to refer to official sources or the latest documentation for the most accurate information[28].

## 2. Research Methods

Briefly, the research process for traffic density analysis using the asphalt detection approach using computer vision is presented in Figure 1. The explanation of each process in Figure 1 is as follows:

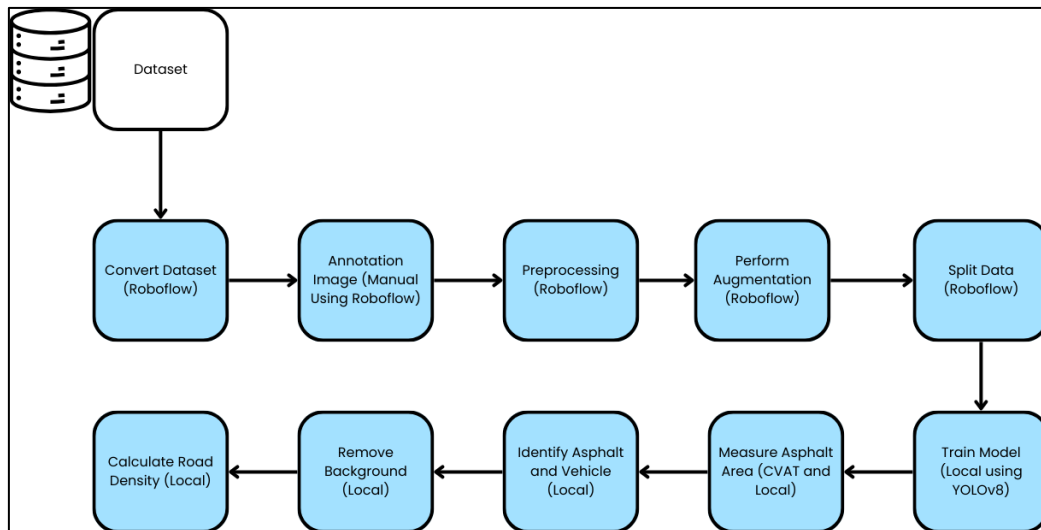


Figure 1. Research Flow

### a) Dataset

The first step in this thesis project is to collect a dataset through the Mataram City Transportation Agency. This dataset should include diverse traffic situations, including dense conditions and light changes. Here is the link to the dataset: <https://bit.ly/DatasetThesis>

### b) Convert Dataset (Roboflow)

This step involves converting the video into a series of static images known as frames. In the context of developing an object recognition model, videos are usually split into a series of images to form a more extensive training dataset. Each image is considered as a separate entity and will be used in the model training process.

### c) Annotation Image (Manual Using Roboflow)

Image annotation is the process of preparing a dataset for training an object recognition model. This process involves identifying and labeling relevant objects in each image. Some annotation tools options include LabelImg, VGG Image Annotator, and Roboflow. The best reason to use Roboflow over other tools is that the platform provides a comprehensive and intuitive experience for annotating data. Roboflow is designed to be easy to use for users at various levels, even those with no prior experience in data annotation. Its full features such as integration with various file formats, data visualization tools, and support for various annotation tasks make Roboflow a strong choice. Its ability to easily scale according to project needs, whether small or large, is also a major advantage. Additionally, its ability to collaborate with teams and strong integration with other platforms and tools make Roboflow a comprehensive and efficient solution for data annotation in Machine Learning and Computer Vision projects. Data annotation using Roboflow is done manually by marking objects using segmentation and providing appropriate labels. The labels given by the 2 labels are segmentation to recognize asphalt (Asphalt) and objects above asphalt are vehicles (Vehicle). This step is a prerequisite for building an accurate model.

**d) Preprocessing (Roboflow)**

This process involves resizing images in the dataset to ensure consistent size during model training. Consistent image sizes allow the model to learn better and produce more consistent results. Typically, images are resized to a predetermined size, such as 1200x720 pixels, to fit the training requirements.

**e) Perform Augmentation (Roboflow)**

Image augmentation is a technique used to create variations in the training dataset by slightly modifying the image features. This can include rotation, cropping, shifting, and changing colors. The goal is to produce a more diverse dataset and allow the model to learn from various situations and lighting conditions.

**f) Split Data (Roboflow)**

Dividing the dataset into different subsets is important for managing data efficiently during model training. Typically, the dataset is divided into three main subsets: training, validation, and testing. The training subset is used to train the model, the validation subset is used to measure the model's performance during training, and the testing subset is used to test the model's performance after training is complete. Proper data division ensures that the model can learn well and be tested properly.

**g) Train Model (Local using YOLOv8)**

The model training process involves providing the training dataset to the model to learn patterns and features in the data. The model uses machine learning techniques to adjust parameters to recognize and differentiate the desired objects in the images. Training the model requires sufficient time and resources to achieve optimal results.

**h) Measure Asphalt Area (CVAT and Local)**

This step involves measuring the area of asphalt in the image without any objects or vehicles on it. This measurement is usually done using pixels in the image to determine the area that will be used as a reference for calculating vehicle density. CVAT stands for Computer Vision Annotation Tool, which is used to annotate and label objects within images. The measurement of the asphalt area is done by calculating the difference between the green and red pixel counts in the image, where the asphalt pixels are predefined. The equation for this measurement is

$$\text{Asphalt area} = \sum_{i=1}^m \sum_{j=1}^n (\text{Green Pixels}(i, j) - \text{Red Pixels}(i, j)) \quad (1)$$

Where:

- $m$  is the number of rows in the image.
- $n$  is the number of columns in the image.
- $\text{Green Pixels}(i, j)$  is the count of green pixels at position  $(i, j)$
- $\text{Red Pixels}(i, j)$  is the count of red pixels at position  $(i, j)$

This equation calculates the asphalt area by subtracting the red pixel count from the green pixel count for each pixel position  $(i, j)$  in the image, ensuring that only the asphalt area is measured. By defining the asphalt pixels and using the difference between green and red pixel counts, we can accurately measure the asphalt area that will be used as a reference for vehicle density calculations. This method provides a clear and precise approach to segmenting and quantifying the asphalt area in traffic images.

**i) Identify Asphalt and Vehicle (Local)**

After the model is trained, the images in the dataset can be processed to identify asphalt areas as well as objects or vehicles on them. The model will predict the location and type of objects based on the learning it has done during the training process.

**j) Remove Background (Local)**

After performing segmentation to identify vehicle and road areas, coloring is applied to the detected areas. Areas classified as vehicles are colored red, while areas classified as roads are colored green. This coloring aims to provide a clear visual representation of traffic density in the observed area. Additionally, areas not included in the segmentation are colored black to improve the accuracy of density detection results. With this visualization, the traffic density picture can be analyzed more deeply and easily understood.

**k) Calculate Road Density (Local)**

The final step involves calculating the number of vehicles based on the amount of visible asphalt area in the image. Using the HVS color system, vehicles can be identified and counted based on information about the proportion of asphalt area occupied by the vehicles. This provides an estimate of vehicle density in a specific area recorded in the image.

### 3. Result and Discussion

#### 3.1 Convert Dataset (Roboflow)

The third dataset conversion process from video format with a duration of 10 minutes and 19 seconds into 100 frames (as shown in Figure 2) is done to simplify the dataset and obtain a more representative representation of the video content. Converting each video into 100 frames allows the dataset to cover various situations and conditions that may occur during the video, thus enriching the dataset with the necessary variations to train the model well. Additionally, by reducing the complexity of the video into individual frames, the model training process becomes more efficient by reducing the amount of data that needs to be processed. Converting the dataset into image format also ensures data consistency and ease of use in the model, as most models accept images as input more easily than videos. Thus, converting the dataset into 300 frames in image form helps improve the quality and efficiency of model training.



**Figure 2.** Convert Dataset Visualization

#### 3.2 Annotation Images (Manual Using Roboflow)

In the segmentation process using Roboflow, images are annotated using Roboflow to identify and mark road/asphalt areas and objects above them. Annotation is done by assigning the appropriate label to each pixel in the image, indicating whether it is part of the road/asphalt or an object above it. Then, the segmentation process is performed to separate the road/asphalt from the objects above it in the image. The segmentation model is trained to identify and isolate road/asphalt areas with high accuracy, enabling further analysis of those areas. An example of annotated image results can be seen in Figure 3.



Figure 3. Image Annotation Visualization

### 3.3 Preprocessing (Roboflow)

In the preprocessing stage, the auto-orient process is performed to ensure that the orientation of frames in the dataset is consistent, making it easier for further processing and analysis. Additionally, resizing the frames to a smaller size like 1200x720 pixels aims to adjust the normal length and width of frames in general. The initial length and width of the frames are 1280x720. Then, Auto-Adjust Contrast: Using Contrast Stretching is applied to enhance the contrast in the images, making unclear or blurry details clearer. Auto-Adjust Contrast: Using Contrast Stretching is applied to enhance the contrast in the images. The formula used for Contrast Stretching is given in Eq. (2).

$$New\ Pixel = (old\ pixel - Min) * \frac{newMax - NewMin}{Max - Min} + NewMin \quad (2)$$

In this formula, New Pixel is the new pixel value that has been adjusted for contrast, Old Pixel is the original pixel value, Min and Max are the minimum and maximum pixel values in the image, while NewMin and NewMax are the desired pixel value ranges after contrast adjustment. This process will make unclear or blurry details in the image clearer. Thus, the image can be more prepared for further analysis after going through this preprocessing stage.

### 3.4 Perform Augmentation (Roboflow)

Data augmentation is an important strategy in the development of machine learning models aimed at improving performance and generalization. In the context of road data processing, this augmentation involves several techniques such as horizontal flipping, saturation adjustment, brightness adjustment, blurring, and adding noise. Horizontal flipping is used to create mirror image variations, allowing the model to understand objects from different perspectives, which is important in road data analysis. Saturation and brightness adjustments aim to address variations in lighting that may occur in road environments, which can affect the image. Blurring with a 1-pixel boundary helps reduce noise and blurriness in the image, while adding noise at a level of 0.38% of pixels can help improve the model's resistance to minor disturbances in the image. By applying this augmentation, the training dataset becomes more diverse and representative of conditions that may occur in the real world, thus improving the model's performance in recognizing and classifying objects on the road.

Furthermore, augmentation has also been proven effective in preventing overfitting. With additional variations in the dataset, the model tends not to rely too much on specific patterns that may exist in the original dataset, thus improving the model's ability to generalize to new data. Evaluation results show that models trained with augmented datasets are able to perform better on test data than models trained with original datasets without augmentation.

### 3.5 Split Data (Roboflow)

In the preprocessing stage, the dataset is divided into three main subsets: the train set, validation set, and test set. The train set accounts for approximately 87% of the total dataset, which is



equivalent to around 624 images. The train set is used to train the model to recognize patterns and features in the data. The validation set, which consists of around 60 images, is used to measure the performance of the model during the training process and to assist in adjusting the model's parameters. Meanwhile, the test set, which comprises about 30 images, is used to test the performance of the model after training is completed. This division of the dataset is important to ensure that the model can learn effectively from the training data and can generalize well to new data that it has not seen before.

### 3.6 Train Model (Local using YOLOv8)

In this experiment, training was conducted using the YOLOv8 architecture for segmentation tasks, utilizing a local computer as the resource. The model used was yolov8n-seg.pt with the dataset configuration defined in the config.yaml file. Training was carried out for 10 epochs with a batch size of 16. The images used for training were sized 1200x720 pixels. During training, optimization was performed using automatic optimization methods with an initial learning rate (lr0) of 0.01. Optimization was also performed with a momentum of 0.937 and weight decay of 0.0005 to reduce overfitting. Additionally, other parameters were adjusted, such as momentum during the warm-up stage (warmup\_momentum), and training during the warm-up stage for 3 epochs (warmup\_epochs) with a bias learning rate (warmup\_bias\_lr) of 0.1. The choice of a batch size of 16 was made to balance the trade-offs between training speed and memory constraints. A batch size of 16 was found to be the most optimal for this experiment, providing a reasonable compromise between computational efficiency and the ability to generalize from the training data. Using this batch size, the model was able to effectively learn from the data without running into memory issues or requiring excessive computation time. Regarding the training parameters, they were chosen based on common practices and preliminary experiments to ensure effective training. The initial learning rate (lr0) of 0.01 is a standard starting point that allows the model to learn efficiently while avoiding drastic weight updates that could destabilize training. The momentum of 0.937 helps accelerate gradients vectors in the right directions, leading to faster converging. The weight decay of 0.0005 helps reduce overfitting by penalizing large weights, encouraging the model to generalize better. The warm-up parameters, including a warm-up momentum (warmup\_momentum) and training during the warm-up stage for 3 epochs (warmup\_epochs) with a bias learning rate (warmup\_bias\_lr) of 0.1, were chosen to gradually introduce the model to the learning process, preventing large updates at the start that could lead to poor convergence. While these parameters were effective in our experiments, further fine-tuning and experimentation could potentially yield even more optimal results. Continuous evaluation and adjustment of these parameters based on the model's performance and specific dataset characteristics are essential to achieving the best possible outcomes.

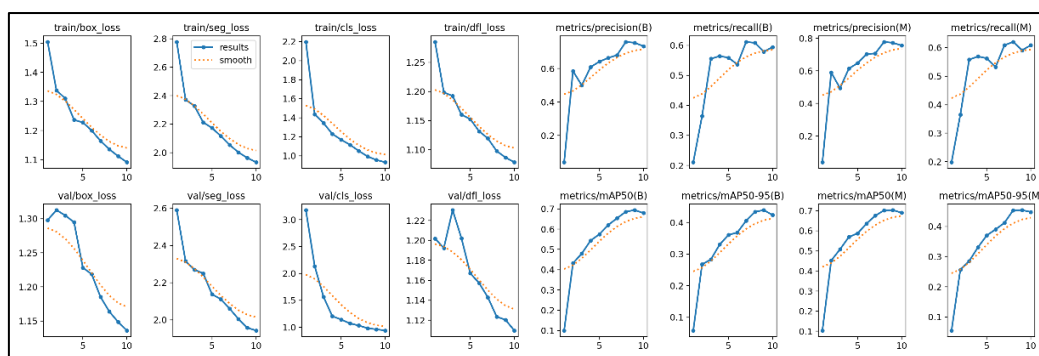


Figure 4. YOLOv8 Training Result plot Visualization

The training results show that the model successfully trained the dataset, with a box\_loss of 1.091, seg\_loss of 1.932, cls\_loss of 0.9323, and dfl\_loss of 1.079, as shown in Figure 4. Evaluation was performed at the end of training using precision (P), recall (R), and mean Average Precision (mAP) metrics for sharp-edged objects (Box) and blurry-edged objects (Mask). The evaluation results show that the model achieved a mAP50 of 0.679 and mAP50-95 of 0.424 for sharp-edged objects, as well as a mAP50 of 0.755 and mAP50-95 of 0.609 for blurry-edged objects. Overall, the model achieved an mAP of 0.69 for sharp-edged objects and 0.448 for blurry-

edged objects, using a dataset of 60 images and 1287 instances. The training process took approximately 5.382 hours on a local computer. The training process for this research was conducted on a local computer with specific hardware and software configurations to ensure optimal performance and efficiency. The hardware used was an HP Pavilion Gaming Laptop 15 ec2010AX, equipped with an AMD Ryzen 5 processor, an NVIDIA GeForce RTX 30 graphics card, and 16GB of RAM. This setup provided the necessary computational power and memory to handle the training tasks effectively. In terms of software, the local computer operated on a 64-bit version of Windows 11. The programming language used for developing and training the model was Python 3.8.0, a widely adopted language in machine learning and data science due to its extensive libraries and ease of use. The development environment was Visual Studio Code, a versatile and user-friendly integrated development environment (IDE) that facilitated coding, debugging, and running the training scripts. These hardware and software specifications collectively ensured that the training of the YOLOv8 model could be executed efficiently and effectively, providing a solid foundation for the research and allowing for the accurate evaluation of the model's performance. The visualization of the training results for each batch can be seen in Figure 5.

The achieved loss values provide insights into the model's performance. The box loss of 1.091 suggests that the model has a moderate level of accuracy in predicting object boundaries. The segmentation loss of 1.932, which is relatively higher, indicates that the model may have struggled more with precise segmentation tasks. The classification loss of 0.9323, being relatively low, suggests good classification performance, while the Distribution Focal Loss (DFL) of 1.079 indicates moderate accuracy in predicting probability distributions.

The evaluation metrics reveal further details about the model's performance. For sharp-edged objects, the model achieved a mAP50 of 0.679, indicating that it correctly identifies 67.9% of these objects at a 50% Intersection over Union (IoU) threshold, which suggests fairly good performance. The mAP50-95 of 0.424 for sharp-edged objects highlights some difficulty at higher precision levels across different IoU thresholds. The overall mAP of 0.69 for sharp-edged objects indicates a high accuracy in general. For blurry-edged objects, the model performed better, with a mAP50 of 0.755 and a mAP50-95 of 0.609, suggesting better precision across various IoU thresholds. The overall mAP of 0.448 for blurry-edged objects, while lower than the mAP50, still demonstrates a reasonable level of accuracy.

The training process, which took approximately 5.382 hours on a local computer, indicates the model's efficiency given the hardware constraints and dataset size. The detailed visualization of training results in Figure 5 provides insights into how the model's performance evolved over each batch, which can aid in further refinement and tuning of the model. Overall, while the model shows balanced performance in both sharp-edged and blurry-edged object detection, with a slight edge in detecting blurry-edged objects, there is room for improvement in precise segmentation tasks. Further optimization of model parameters and additional training on a more extensive dataset could enhance the model's accuracy and efficiency.

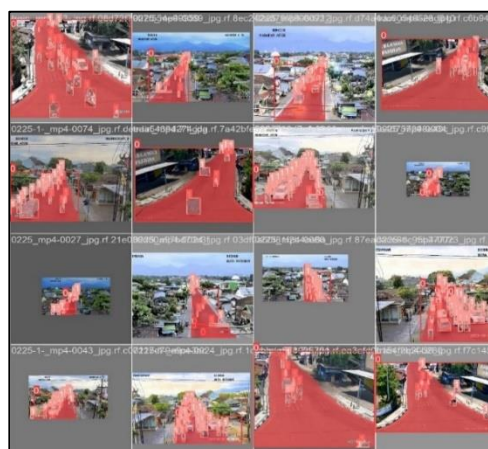


Figure 5. YOLOv8 batch Training Result Visualization



The confusion matrix is used to evaluate the model's performance in the segmentation task, where there are 3 predicted classes: asphalt, vehicle, and background. Background represents parts of the image that are not included in the segmentation of the objects of interest. From the confusion matrix results, it is known that the predicted vehicle class has the highest correct predictions, with a total of 876. This indicates that the model is able to effectively identify and separate areas that are vehicles in the image. The confusion matrix results can be seen in Figure 6. However, evaluation metrics for segmentation tasks are not sufficient if only using the confusion matrix. To provide a more comprehensive evaluation of the segmentation model, the following additional metrics should be presented:

1. Mean Average Precision (mAP) Mask at a Threshold of 50-95: This metric evaluates the model's precision and recall across different intersection-over-union (IoU) thresholds, providing a more nuanced understanding of the model's performance in accurately segmenting objects of various sizes and shapes.
2. Model Parameters: Reporting the number of parameters in the model helps to understand its complexity and potential trade-offs between performance and computational efficiency.
3. Model Complexity in FLOPs (Floating Point Operations per Second): This metric measures the computational cost of running the model, providing insight into the model's efficiency and suitability for deployment in different environments.

By including these additional metrics, a more comprehensive evaluation of the model's performance can be achieved, ensuring that the segmentation results are robust, accurate, and efficient. Figure 6.

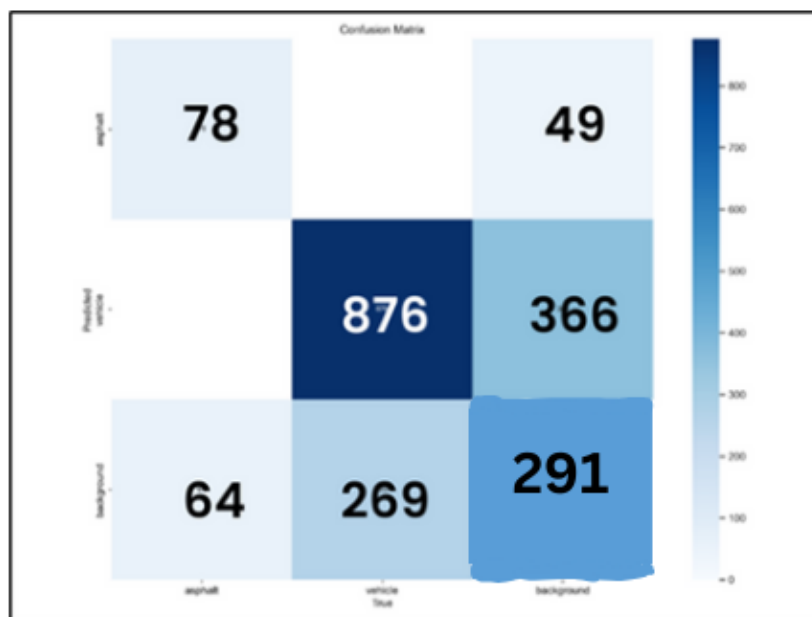


Figure 6. Confusion Matrix Visualization




In the given confusion matrix, the model's performance is evaluated for three classes: asphalt, vehicle, and background. For the asphalt class, the model shows an accuracy of 55%. The precision for this class is 54.9%, meaning that out of all the predictions the model made for asphalt, only 54.9% were actually asphalt. However, the recall for the asphalt class is 61.4%, indicating that the model was able to detect 61.4% of the total instances of asphalt present in the dataset. For the vehicle class, the model performs better, with an accuracy of 71%. The precision for vehicle detection is 70.5%, meaning that out of all the predictions the model made for vehicles, 70.5% were correct. The recall for this class is 76.5%, showing that the model effectively identified 76.5% of the total instances of vehicles in the dataset. For the background class, the model achieves an accuracy of 50%. However, the precision for the background class is 50%, meaning that out of all the predictions the model made for background, only 50% were actually background. The recall for background is 44.3%, indicating that the model was able to identify 44.3% of the

total instances of background present in the dataset. Overall, the model shows better performance in detecting the vehicle class compared to the asphalt and background classes. However, the model requires further improvement in detecting the asphalt and background classes to achieve more balanced and accurate results across all classes.

### 3.7 Measure Asphalt Area (CVAT and Local)

At this stage, the percentage of pixels identified as asphalt from 3 locations in Mataram is identified. The process begins with capturing image samples from different locations using a digital camera. Each captured image is then uploaded to CVAT (Computer Vision Annotation Tool) for byclass segmentation. Byclass segmentation allows for more accurate recognition of pixels that are asphalt and non-asphalt. CVAT has several advantages that make it the right choice for asphalt segmentation. One of the strong reasons is its ability to export segmentation results in the form of segmentation masks. This feature allows users to easily view and manipulate the segmentation results, as well as integrate them into further analysis processes using Python or other programming languages. Additionally, CVAT provides an intuitive and user-friendly interface, making it easier for users to perform accurate byclass segmentation. After the segmentation process is completed, analysis is performed using Python to calculate the percentage of asphalt pixels in each image. This analysis involves counting the number of asphalt pixels and total pixels in each image, and then calculating the percentage of asphalt pixels. The asphalt measurement results can be seen in Table 1. The analysis results indicate that from 3 different experimental locations in Mataram, the Airlangga location has the highest percentage of asphalt pixels, followed by the Pagesangan and Rembiga locations.

**Table 1.** Asphalt Measurement Result

No	image	Intersection Name	Total image pixel	Asphalt pixel	Asphalt pixel percentage
1		Airlangga (experiment point 1)	864000	377638	43.7%
2		Pagesangan (experiment point 2)	864000	148943	17.2%
3		Rembiga (experiment point 3)	864000	79664	9.2%

### 3.8 Identify Asphalt and Vehicles (Local)

In this section, the identification of asphalt and vehicles is performed using the previously fine-tuned YOLOv8 model. The base model used to detect both classes is YOLOv8\_Seg.pt. This identification process involves sending input images to the YOLOv8\_Seg.pt model. The model will make predictions on the input images and identify the location and class of asphalt and vehicles in the images. The prediction results include segmentation, marking the location of each detected object, along with its class label. The results of asphalt and vehicle identification using the YOLOv8 segmentation model can be seen in Figure 7.

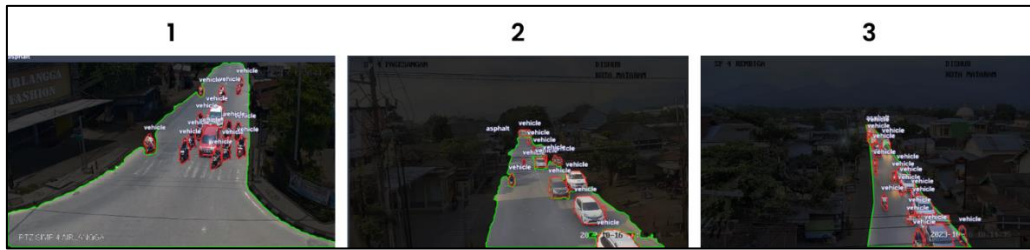


Figure 7. Asphalt and Vehicle Segmentation using YOLOv8

### 3.9 Background Remover (Local)

In this stage, the background or masking of each detected class is separated, where asphalt is marked with green and vehicles with red. Classes other than these two classes will be given black color. The masking process is done to facilitate the identification of asphalt and vehicles in road images. The main reason for masking is to improve the ease of identifying asphalt and vehicles in road images. By giving green color to asphalt and red color to vehicles, these objects will be more easily distinguished from the background. Masking also allows for better extraction of main objects in the image, enabling further analysis such as calculating asphalt area, identifying vehicle types, and estimating traffic volume. Thus, masking is an important step in the image processing process for better road and traffic analysis.

In the segmentation process to differentiate between asphalt and vehicles, a masking method is used, taking the prediction results from the trained model. In the initial stage, the input image is processed through the model, producing predictions in the form of masks and mask coordinates. Next, iteration is done on each mask and its coordinates, where each mask is represented as an array and the mask coordinates are adjusted according to the input image size. Then, the class of each mask is checked, with green assigned to represent asphalt if the mask belongs to the 'asphalt' class, and red to represent vehicles if the mask does not belong to the 'asphalt' class. Finally, the pixels in the segmented image that fall within the asphalt or vehicle areas based on the mask values are selected, and these pixels are colored according to their class. By using masking in this way, the segmentation results can clearly differentiate between asphalt and vehicles in the input image, facilitating traffic density analysis. The masking result visualization, showing the differentiation between asphalt and vehicles in the segmented image, can be seen in Figure 8.

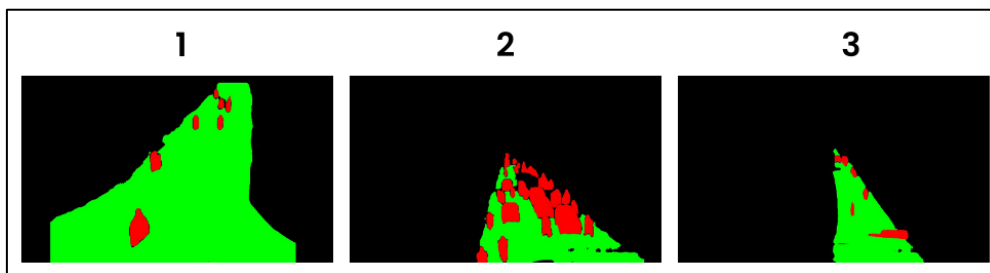


Figure 8. Masking Result Visualization

### 3.10 Calculate Road Density (Local)

Based on the experiment results, traffic density was measured from 3 experimental locations using the number of detected vehicles as a key metric. In this analysis, the average number of detected vehicles from each experimental point for each frame was calculated to gain a deeper understanding of the traffic density at each location. The calculation results showed that the average number of vehicles in frame 9008.500000 was 15.725966 for Location 1, 32.255352 for Location 2, and 20.987616 for Location 3. Thus, it can be concluded that Location 2 has a higher traffic density compared to Location 1 and Location 3. Furthermore, a graph depicting the development of the number of detected vehicles from each experimental point for each frame is presented. This graph provides a visual insight into traffic patterns over time and allows for a

better comparison of traffic density between locations. The plot showing the number of vehicles can be seen in Figure 9.

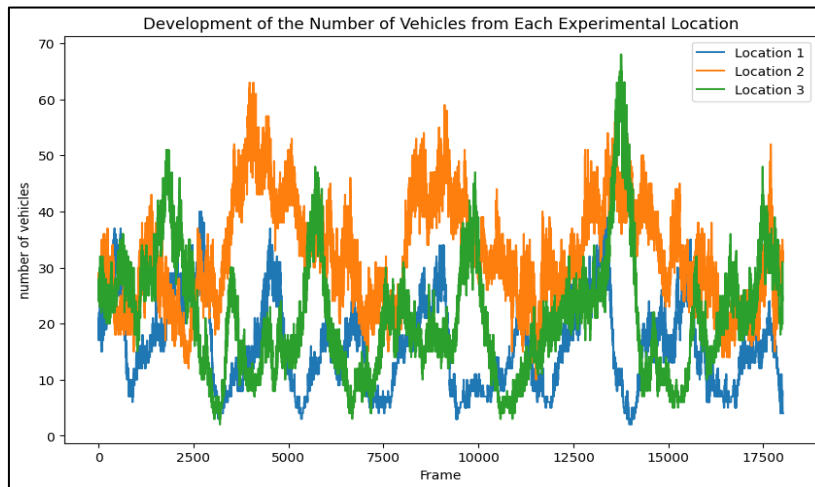


Figure 9. Number of Vehicles Plot

In this study, a comparison was made between traffic density measurements using the YOLOv8 segmentation method and the normal HSV method[25]. In the normal HSV method, several steps were taken, including conversion to the HSV color space[29], creating a mask on the desired area[30], and measuring traffic density based on the asphalt area. In the first stage of the manual HSV method, the lower asphalt parameters used were (0, 19, 0) and the upper asphalt parameters were (230, 250, 230), with the application of bitwise and operation and normal masking operations. In the second experimental stage, the parameters were changed to lower asphalt (100, 19, 20) and upper asphalt (230, 250, 230), with the application of bitwise and operation and bitwise not mask. In the third stage, the parameters used were lower asphalt (110, 5, 50) and upper asphalt (230, 250, 230), with the application of bitwise and operation and bitwise not mask.

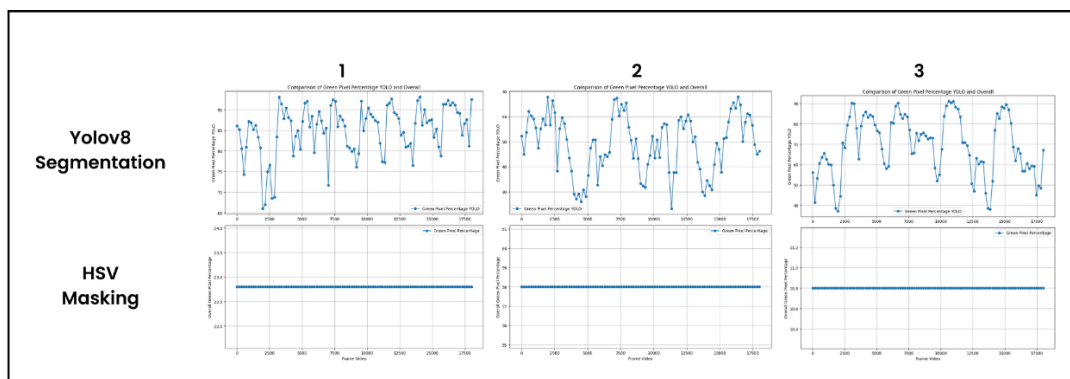


Figure 10. Traffic Density Measurement Plot

Furthermore, the results of the traffic density measurement can be seen in Figure 10. The plot results depict that YOLOv8 segmentation model showed a graph with a wavy pattern. This is because of the use of HSV color-based density detection method. On the other hand, the normal HSV method produced a graph with a more linear pattern. The graphs display the comparison of green pixel percentages between YOLOv8 segmentation and HSV masking methods across three locations. The top row shows the results for YOLOv8 segmentation, while the bottom row displays the results for HSV masking. For Location 1, YOLOv8 segmentation shows significant fluctuations in the green pixel percentage, indicating varying levels of traffic density. Peaks in the graph correspond to higher traffic density, while troughs represent lower density, suggesting that the traffic flow is dynamic, with periods of congestion followed by periods of lighter traffic. In contrast, the HSV masking graph shows a flat line at approximately 2.3%, indicating a constant

green pixel percentage. This constant value suggests that the HSV method is not sensitive to changes in traffic density, likely due to its reliance on color thresholds which do not effectively capture the dynamic nature of traffic flow. For Location 2, YOLOv8 segmentation similarly shows fluctuations in the green pixel percentage, reflecting variations in traffic density, with notable peaks and troughs indicating periods of high and low traffic density respectively. The HSV masking graph remains flat at around 2.3%, again not reflecting the variations in traffic density, suggesting its limited effectiveness in capturing dynamic traffic conditions. For Location 3, YOLOv8 segmentation shows considerable fluctuations in the green pixel percentage, indicating dynamic changes in traffic density. The pattern is consistent with Locations 1 and 2, showing clear periods of higher and lower traffic density. Conversely, the HSV masking graph is flat at around 2.3%, showing no variation in the green pixel percentage, indicating that the HSV method consistently fails to capture the dynamic nature of traffic density at this location as well.

The fluctuating green pixel percentages across all three locations for YOLOv8 segmentation demonstrate its ability to detect and represent variations in traffic density accurately. The wavy patterns reflect the real-time changes in traffic flow, making YOLOv8 a more suitable method for dynamic traffic density measurement. In contrast, the constant green pixel percentage across all locations for the HSV masking method indicates its ineffectiveness in capturing the dynamic changes in traffic density. The reliance on color thresholds causes the HSV method to miss subtle changes in traffic flow, resulting in flat, uninformative graphs. This occurred because traffic density measurement was based on color, and in the observed video, there was a lot of noise such as tree houses, sky, and vehicles that had colors that were similar to asphalt as in Figure 11. This analysis provides an understanding of the differences between the YOLOv8 segmentation method and the normal HSV method in measuring traffic density. The video results for detecting traffic density can be accessed through the following link: <https://bit.ly/ResultExperiment>



Figure 11. Traffic Density Normal HSV Visualization

#### 4. Conclusion

This research highlights the importance of using computer vision techniques for traffic management, especially in vehicle detection and traffic density estimation. By using advanced algorithms like YOLOv8 for object segmentation and asphalt detection, this study successfully demonstrates the effectiveness of this approach in measuring traffic density. The comparison clearly shows that YOLOv8 segmentation is more effective in capturing and representing the dynamic nature of traffic density compared to the HSV masking method. YOLOv8's ability to show fluctuations in the green pixel percentage provides a more accurate and detailed understanding of traffic conditions, while the HSV method fails to reflect these variations, leading to less informative results. Emphasizing the importance of selecting the right technique for road data analysis. Future research directions include optimizing segmentation models and exploring additional image processing methods to improve traffic management system efficiency. Based on the experimental results, the analysis of traffic density at three experimental locations reveals different vehicle detection rates. Location 2 shows the highest traffic density with an average of 32.255352 vehicles detected per frame, followed by Location 1 with 15.725966 vehicles and Location 3 with 20.987616 vehicles. This indicates that Location 2 experiences higher traffic volume compared to the other two locations. The graphical presentation of the number of detected vehicles over time also illustrates traffic patterns and density differences between experimental locations. In future research, a model optimization approach should be suggested, such as experimenting with the best model parameters and modifying the YOLO model, in addition to the image processing approach. By continuously improving and refining these techniques, we can



contribute to more effective traffic management strategies and ultimately enhance urban transportation systems.

## References

- [1] D. Ariyoga, R. Rahmadi, and R. A. Rajagede, "Penelitian Terkini Tentang Sistem Pendeteksi Pelanggaran Lalu Lintas Berbasis Deep Learning: Sebuah Kajian Pustaka," 2021.
- [2] A. Stanley and R. Munir, "Vehicle Traffic Volume Counting in CCTV Video with YOLO Algorithm and Road HSV Color Model-based Segmentation System Development," *Proceeding 15th Int. Conf. Telecommun. Syst. Serv. Appl. TSSA 2021*, vol. C, pp. 1–5, 2021, doi: 10.1109/TSSA52866.2021.9768231.
- [3] "Korlantas Polri - Informasi Lalu Lintas Indonesia." Accessed: Dec. 21, 2022. [Online]. Available: <https://korlantas.polri.go.id/>
- [4] J. de Winter, J. Hoogmoed, J. Stapel, D. Dodou, and P. Bazilinskyy, "Predicting perceived risk of traffic scenes using computer vision," *Transp. Res. Part F Traffic Psychol. Behav.*, vol. 93, no. August 2022, pp. 235–247, 2023, doi: 10.1016/j.trf.2023.01.014.
- [5] M. Michael, F. Tanoto, E. Wibowo, F. Lutan, and A. Dharma, "Pengenalan Plat Kendaraan Bermotor dengan Menggunakan Metode Template Matching dan Deep Belief Network," *MATRIK J. Manajemen, Tek. Inform. dan Rekayasa Komput.*, vol. 19, no. 1, pp. 27–36, 2019, doi: 10.30812/matrik.v19i1.475.
- [6] W. Ci, T. Xu, R. Lin, and S. Lu, "A Novel Method for Unexpected Obstacle Detection in the Traffic Environment Based on Computer Vision," *Appl. Sci.*, vol. 12, no. 18, 2022, doi: 10.3390/app12188937.
- [7] H. S. Bedi, P. K. Malik, R. Singh, R. Singh, and N. Bisht, "Traffic Surveillance Using Computer Vision and Deep Learning Methods," *IOP Conf. Ser. Earth Environ. Sci.*, vol. 1285, no. 1, 2024, doi: 10.1088/1755-1315/1285/1/012018.
- [8] I. M. Sukarsa, N. Kadek, D. Rusjayanthi, M. Srinitha, M. Utami, and N. Wayan, "The Use of XGBoost Algorithm to Analyse the Severity of Traffic Accident Victims," *LONTAR Komput.*, vol. 14, no. 1, pp. 36–47, 2023.
- [9] F. Rofii, G. Priyandoko, M. I. Fanani, and A. Suraji, "Peningkatan Akurasi Perhitungan Jumlah Kendaraan dengan Membangkitkan Urutan Identitas Deteksi Berbasis Yolov4 Deep Neural Networks," *TEKNIK*, vol. 42, no. 2, pp. 169–177, Aug. 2021, doi: 10.14710/teknik.v42i2.37019.
- [10] M. Boudissa, H. Kawanaka, and T. Wakabayashi, "Semantic Segmentation of Traffic Landmarks Using Classical Computer Vision and U-Net Model," *J. Phys. Conf. Ser.*, vol. 2319, no. 1, 2022, doi: 10.1088/1742-6596/2319/1/012031.
- [11] W. Riyadi and J. Jasmir, "Performance Prediction of Airport Traffic Using LSTM and CNN-LSTM Models," *MATRIK J. Manajemen, Tek. Inform. dan Rekayasa Komput.*, vol. 22, no. 3, pp. 627–638, 2023, doi: 10.30812/matrik.v22i3.3032.
- [12] Muhamad Irfan Hermawan., Iwan Iwut Tritoasmoro., and Nur Ibrahim., "Pengaturan Lampu Lalu Lintas Berdasarkan Kepadatan Kendaraan Menggunakan Metode YOLO," Feb. 2021.
- [13] M. V. Yashina, A. S. Bugaev, I. A. Kuteynikov, and M. A. Kuznetsov, "Evaluation of Recovery Accuracy of Vehicles Flow Characteristics by Video Sensor Views," *2022 Syst. Signals Gener. Process. F. Board Commun. SOSG 2022 - Conf. Proc.*, pp. 1–4, 2022, doi: 10.1109/IEEECONF53456.2022.9744271.
- [14] K. Darwhekar, A. Patil, S. Ghodke, R. Bawkar, and S. Rudrawar, "Computer Vision based Intelligent Traffic Management System," *6th Int. Conf. Electron. Commun. Aerosp. Technol. ICECA 2022 - Proc.*, no. Iceca, pp. 1051–1056, 2022, doi: 10.1109/ICECA55336.2022.10009105.
- [15] I. Imanuddin, F. Alhadi, R. Oktafian, and A. Ihsan, "Deteksi Mata Mengantuk pada Pengemudi Mobil Menggunakan Metode Viola Jones," *MATRIK J. Manajemen, Tek. Inform. dan Rekayasa Komput.*, vol. 18, no. 2, pp. 321–329, 2019, doi: 10.30812/matrik.v18i2.389.
- [16] A. Kurniasari and Jalinus, "Pendeteksian Tingkat Kepadatan Jalan Menggunakan Metode Canny Edge Detection," *J. Ilm. Teknol. dan Rekayasa*, vol. 25, no. 3, pp. 239–248, 2020,



- doi: 10.35760/tr.2020.v25i3.3419.
- [17] M. Hasan, S. Das, and M. N. T. Akhand, "Estimating Traffic Density on Roads using Convolutional Neural Network with Batch Normalization," *2021 5th Int. Conf. Electr. Eng. Inf. Commun. Technol. ICEEICT 2021*, pp. 1–6, 2021, doi: 10.1109/ICEEICT53905.2021.9667860.
- [18] M. Umair Arif, M. U. Farooq, R. H. Raza, Z. U. A. Lodhi, and M. A. R. Hashmi, "A Comprehensive Review of Vehicle Detection Techniques under Varying Moving Cast Shadow Conditions Using Computer Vision and Deep Learning," *IEEE Access*, vol. 10, no. October, pp. 104863–104886, 2022, doi: 10.1109/ACCESS.2022.3208568.
- [19] M. Umair, M. U. Farooq, R. H. Raza, Q. Chen, and B. Abdulhai, "Efficient Video-based Vehicle Queue Length Estimation using Computer Vision and Deep Learning for an Urban Traffic Scenario," *Processes*, pp. 1–20, 2021.
- [20] K. H. Nam Bui, H. Yi, and J. Cho, "A multi-class multi-movement vehicle counting framework for traffic analysis in complex areas using CCTV systems," *Energies*, vol. 13, no. 8, 2020, doi: 10.3390/en13082036.
- [21] S. Javadi, *Computer Vision for Traffic Surveillance Systems Methods and Applications*. 2021.
- [22] R. J. A. F. Utaminingrum, and A. Setia, "Helmet Monitoring System using Hough Circle and HOG based on KNN," *LONTAR Komput.*, vol. 12, no. 1, pp. 13–23, 2021.
- [23] Z. Liu, C. Shen, X. Fan, G. Zeng, and X. Zhao, "Scale-aware limited deformable convolutional neural networks for traffic sign detection and classification," *IET Intell. Transp. Syst.*, vol. 14, no. 12, pp. 1712–1722, 2020, doi: 10.1049/iet-its.2020.0217.
- [24] X. Chen, Y. Chen, and G. Zhang, "A computer vision algorithm for locating and recognizing traffic signal control light status and countdown time," *J. Intell. Transp. Syst. Technol. Planning, Oper.*, vol. 25, no. 5, pp. 533–546, 2021, doi: 10.1080/15472450.2021.1871611.
- [25] A. A. Mahersatillah, Z. Zainuddin, and Y. Yusran, "Unstructured Road Detection and Steering Assist Based on HSV Color Space Segmentation for Autonomous Car," *2020 3rd Int. Semin. Res. Inf. Technol. Intell. Syst. ISRITI 2020*, pp. 688–693, 2020, doi: 10.1109/ISRITI51436.2020.9315452.
- [26] D. Reis, J. Kupec, J. Hong, and A. Daoudi, "Real-Time Flying Object Detection with YOLOv8," 2023, [Online]. Available: <http://arxiv.org/abs/2305.09972>
- [27] R. Ekhlakov and N. Andriyanov, "Multicriteria Assessment Method for Network Structure Congestion Based on Traffic Data Using Advanced Computer Vision," *Mathematics*, vol. 12, no. 4, 2024, doi: 10.3390/math12040555.
- [28] T. Wu and Y. Dong, "YOLO-SE: Improved YOLOv8 for Remote Sensing Object Detection and Recognition," *Appl. Sci.*, vol. 13, no. 24, p. 12977, 2023, doi: 10.3390/app132412977.
- [29] S. Jeong, S. Roh, and K. Sohn, "Multi-Regime Analysis for Computer Vision- Based Traffic Surveillance Using a Change-Point Detection Algorithm," *IEEE Access*, vol. 9, pp. 40980–40995, 2021, doi: 10.1109/ACCESS.2021.3064603.
- [30] K. H. K. Manguri and A. A. Mohammed, "A Review of Computer Vision-Based Traffic Controlling and Monitoring," *UHD J. Sci. Technol.*, vol. 7, no. 2, pp. 6–15, 2023, doi: 10.21928/uhdjst.v7n2y2023.pp6-15.