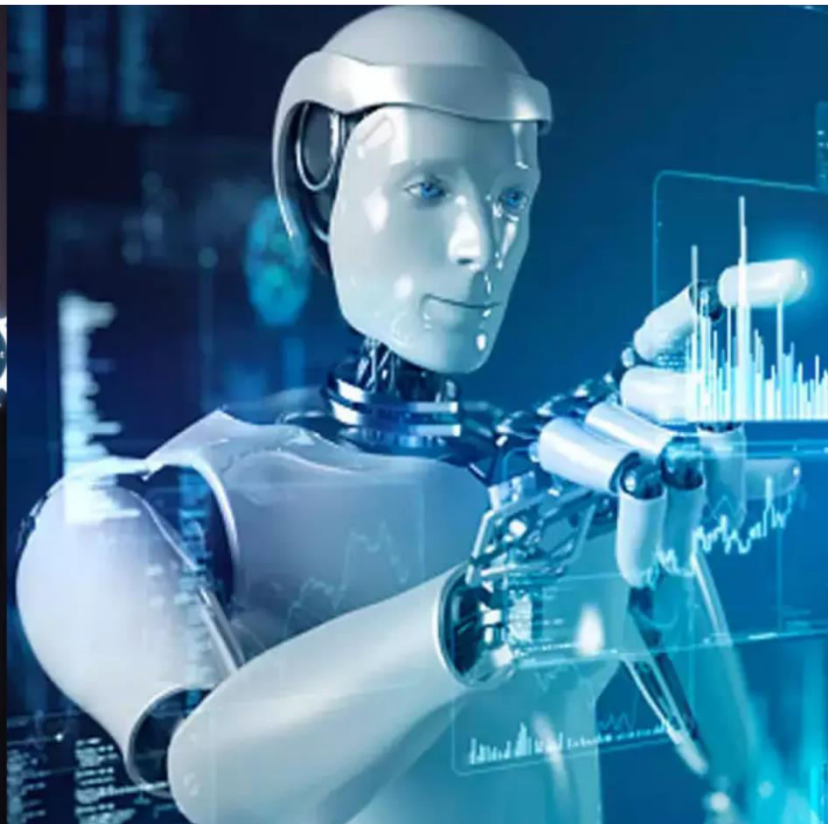




# International Journal of Innovative Research in Science Engineering and Technology (IJIRSET)

*(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)*



**Impact Factor: 8.699**

**Volume 14, Issue 5, May 2025**

# Traffic Vehicle Detection

Sagar KR, Ramesh B E, Yashas MG, Prajwal TM, Nithin H

Professor, Department of Computer Science, S.J.M.I.T, Chitradurga, Karnataka, India

Project co-ordinator, Department of Computer Science, S.J.M.I.T, Chitradurga, Karnataka, India

BE, 8<sup>th</sup> semester, Department of Computer Science, SJMIT, Chitradurga, Karnataka, India

BE, 8<sup>th</sup> semester, Department of Computer Science, SJMIT, Chitradurga, Karnataka, India

BE, 8<sup>th</sup> semester, Department of Computer Science, SJMIT, Chitradurga, Karnataka, India

**ABSTRACT:** The increasing number of vehicles on the roads has led to significant challenges in traffic management, congestion control, and road safety. Traditional manual methods of vehicle counting are inefficient, error-prone, and not scalable for modern smart city applications. To address these limitations, this project presents an automated Traffic Vehicle Detection and Counting System using Machine Learning and computer vision techniques. The system leverages video input to detect and count vehicles in real time, providing an efficient and reliable way to monitor road traffic.

The core of the system is built using Python and OpenCV, utilizing background subtraction and contour detection methods to identify moving vehicles in video frames. A predefined counting line is used to increment the vehicle count when a detected vehicle crosses it. The algorithm is optimized with morphological operations and filters to enhance detection accuracy and reduce noise. The solution is cost-effective, runs on standard computing hardware, and does not rely on expensive sensors or infrastructure changes, making it suitable for widespread adoption.

This project demonstrates the practical application of machine learning and computer vision in addressing real-world problems in traffic analysis. With further enhancements, such as the integration of deep learning models, license plate recognition, and real-time dashboards, the system can evolve into a comprehensive traffic monitoring tool. The successful implementation of this system paves the way for smarter and more data-driven traffic management strategies, contributing to the development of intelligent transportation systems.

## I. INTRODUCTION

In the modern era of rapid urbanization and increased vehicle ownership, managing road traffic efficiently has become a significant challenge. Manual traffic monitoring systems are increasingly proving to be insufficient due to their limitations in real-time tracking, scalability, and accuracy. Hence, there is a growing demand for automated systems that can effectively detect and count vehicles to support traffic management, urban planning, and law enforcement. The integration of machine learning with computer vision provides a powerful approach to address this need by enabling intelligent, automated, and real-time traffic surveillance solutions.

This project, titled "Traffic Vehicle Detection and Counting System using Machine Learning," aims to develop a system that leverages computer vision techniques to detect moving vehicles in video feeds and count them accurately as they cross a predefined line. The system employs OpenCV for video processing and uses background subtraction methods to identify and track moving vehicles. It is designed to work with both recorded video footage and real-time camera input, making it adaptable for various traffic monitoring applications.

The core functionality of the system revolves around detecting vehicles based on motion, extracting relevant features such as size and location, and incrementing a counter when the vehicle crosses a specified detection line. The implementation is efficient and lightweight, suitable for deployment in real-time scenarios. The success of this project can significantly enhance traffic data collection, enable traffic flow optimization, and reduce the need for manual supervision.

This project presents an advanced vehicle detection and counting system that leverages OpenCV's sophisticated computer vision algorithms to deliver real-time traffic analytics. At the core of our implementation lies the Mixture of Gaussians (MOG2) background subtraction algorithm, enhanced with a multi-stage image processing pipeline that includes:

- Adaptive noise reduction through Gaussian blurring
- Morphological transformations for object enhancement
- Contour-based vehicle detection with size filtering
- Centroid tracking for precise counting

Our system introduces several technological innovations, including a dynamic counting mechanism with configurable detection thresholds and a robust validation process that combines multiple computer vision techniques to ensure high accuracy under varying environmental conditions. The implementation demonstrates how fundamental image processing algorithms can be strategically combined to create a practical, real-world solution for traffic monitoring.

The significance of this work extends beyond academic interest, offering tangible benefits for smart city infrastructure and transportation management. Key applications include traffic flow optimization, adaptive signal control, toll automation, and security surveillance. Compared to traditional methods, our vision-based approach provides superior flexibility, requiring only camera infrastructure while delivering real-time processing capabilities and actionable traffic insights.

This documentation presents a comprehensive exploration of the system's architecture, algorithmic foundations, and implementation details. Subsequent sections will examine the technical methodology in depth, present performance evaluations under various operating conditions, and discuss potential enhancements through machine learning integration. The project represents a meaningful contribution to the field of intelligent transportation systems, demonstrating how computer vision can address pressing challenges in urban mobility and traffic management.

## II. METHODOLOGIES

To implement the traffic vehicle detection and counting system efficiently, the entire system is divided into well-defined functional modules. Each module is responsible for handling a specific task, which collectively contributes to the detection and counting of vehicles in real-time. Below are the key modules and their detailed descriptions:

### 1. Video Input Module

This module is responsible for capturing video input either from a live camera feed or a prerecorded video file (e.g., CCTV footage, traffic cam videos). The `cv2.VideoCapture()` function in OpenCV is used to load the video stream. The video is processed frame by frame, which forms the basis for subsequent detection and analysis. This modular design allows the system to be easily switched from file-based input to real-time camera-based input with minimal code change.

### 2. Preprocessing Module

Preprocessing is essential for enhancing the quality of the video frames before analysis. This module converts each frame to grayscale using `cv2.cvtColor()` and applies Gaussian blur to reduce noise and smoothen the image using `cv2.GaussianBlur()`. These steps significantly improve the accuracy of background subtraction by reducing the impact of minor fluctuations like shadows or light reflections.

### 3. Background Subtraction and Motion Detection Module

This module detects moving objects in the video using the Background Subtraction technique. The `cv2.createBackgroundSubtractorMOG2()` method is used to differentiate between moving vehicles and the static background. The resulting foreground mask highlights areas of motion, which are then processed to identify actual vehicles. Morphological operations like dilation and closing are used to enhance object shapes and remove small noisy areas.



#### 4. Vehicle Detection and Filtering Module

Contours are detected from the motion mask using `cv2.findContours()`. Each contour is evaluated for its width and height to filter out irrelevant objects such as small debris or noise. Only those objects that exceed predefined size thresholds are considered valid vehicles. Bounding boxes are drawn around detected vehicles using `cv2.rectangle()`, and center points are calculated for each bounding box.

#### 5. Counting Module

This module keeps track of how many vehicles cross a defined virtual line (e.g., a horizontal line drawn at a specific position in the frame). When a detected vehicle's center point crosses this line within a defined offset range, the counter increments. This module ensures that each vehicle is counted only once by removing counted coordinates from the tracking list.

#### 6. Display and Output Module

The final module overlays all relevant information, including bounding boxes, counting lines, and the total vehicle count, directly onto the video frames. This live annotated video is displayed to the user using `cv2.imshow()`. Additionally, vehicle count logs can be stored or printed for record-keeping and further analysis. Future versions of this module can include exporting statistics to external files or cloud dashboards.

These well-structured modules ensure that the system is maintainable, scalable, and capable of being extended with advanced functionalities such as vehicle type classification, speed estimation, or license plate recognition.

### III. MODEL DEVELOPMENT

The development of the traffic vehicle detection and counting model involves multiple stages, each designed to ensure accurate detection, tracking, and counting of vehicles in real-time or recorded video footage. The system uses computer vision techniques powered by OpenCV, and in more advanced versions, machine learning algorithms can be integrated to improve accuracy and adaptability in dynamic environments.

#### 1. Data Acquisition and Preprocessing

The initial step involves capturing traffic footage using surveillance cameras or pre-recorded video files. This video data serves as the input for the vehicle detection system. The frames are extracted from the video feed in real time and converted into grayscale for faster processing. Noise is reduced by applying Gaussian blur, which smoothens the image and enhances the effectiveness of contour detection in subsequent steps.

#### 2. Background Subtraction and Contour Detection

To detect moving vehicles, a background subtraction technique is applied using `cv2.createBackgroundSubtractorMOG2`, which identifies moving objects by comparing current frames with a dynamically updated background model. After subtracting the background, morphological operations such as dilation and closing are used to fill gaps and reduce noise, making vehicle contours more distinguishable.

#### 3. Vehicle Tracking and Counting

Each detected vehicle is assigned a centroid, and these centroids are monitored across frames to track movement. A virtual line is drawn across the frame (usually horizontal), and each time a vehicle's centroid crosses this line in a specific direction, the counter increments. This ensures accurate counting without duplication. To improve robustness, the model incorporates an offset margin that tolerates minor fluctuations in object positioning, avoiding false counts due to noise or overlapping vehicles.

### IV. ML MODELS

The use of machine learning (ML) models in traffic vehicle detection significantly enhances the accuracy and adaptability of the system, especially in complex and dynamic environments. Below are key ML models and techniques commonly used in such systems:

**1. Convolutional Neural Networks (CNNs)**

CNNs are widely used for object detection and classification tasks. They can automatically learn spatial hierarchies of features from input images, making them ideal for detecting vehicles of different shapes, sizes, and orientations.

Applications: Vehicle detection and classification (e.g., car, bus, truck, bike)

Popular Architectures:

YOLO (You Only Look Once) – Fast real-time detection

SSD (Single Shot Multibox Detector) – Balance between speed and accuracy

Faster R-CNN – High accuracy but computationally heavier

**2. Support Vector Machine (SVM)**

SVM is a supervised learning algorithm that can classify objects into different categories. In a traffic context, SVM can be used in combination with features extracted from images (e.g., HOG – Histogram of Oriented Gradients) to detect and differentiate between objects.

Use Case: Detecting vehicles using HOG features and classifying them via SVM

Pros: Effective for binary and multi-class classification

Cons: Less effective than CNNs for raw image inputs; requires feature engineering

**3. Background Subtraction with Contour Analysis (Classical CV + ML)**

In simpler or resource-limited scenarios, vehicle detection is achieved by using background subtraction techniques (e.g., MOG2) followed by contour detection. Although not a learning-based model in itself, it can be enhanced by using ML classifiers to filter out non-vehicle objects.

Integration with ML: Use ML models (e.g., Decision Trees, KNN) to validate detected objects based on shape, size, or motion patterns.

**4. Deep Learning-based Tracking (Optional Add-on)**

For counting and tracking vehicles over time, ML-based tracking models such as:

SORT (Simple Online and Realtime Tracking)

Deep SORT (adds appearance information using CNN embeddings)

These are used to assign unique IDs to each detected vehicle and track their movement across frames, preventing double counting.

**5. Data Flow Diagram**

Data Flow Diagrams (DFDs) are used to visually represent the flow of data within a system. They show how input is transformed into output through various processes. For the Traffic Vehicle Detection and Counting System, the DFD helps in understanding the interactions between system components like video input, processing modules, and output generation. Below is the breakdown of the DFDs for this project.

Level 0 DFD (Context Level Diagram)

The Level 0 DFD provides a high-level overview of the entire system. It identifies the major entities interacting with the system and the overall flow of data.

Entities:

- User: Provides video input to the system (live feed or video file).
- System: Processes video input and outputs vehicle count.

Processes:

- Vehicle Detection System: Accepts video input, processes it using computer vision techniques, and outputs vehicle count.

Data Flows:

- User → [Video Input] → Vehicle Detection System
- Vehicle Detection System → [Count Output] → User

Level 1 DFD (Detailed System Breakdown)

The Level 1 DFD breaks down the main process into multiple sub-processes and illustrates how data flows between them.

Processes:

1. Video Acquisition Module – Captures or reads video frames from the camera or video file.
2. Preprocessing Module – Converts each frame to grayscale and applies Gaussian blur for noise reduction.
3. Background Subtraction Module – Identifies moving objects by subtracting the static background.

Fig 5.1: Data flow diagram



## 6. Proposed System

The proposed system introduces an automated, intelligent solution for detecting and counting vehicles using machine learning and computer vision techniques. This system aims to overcome the limitations of manual counting and traditional traffic monitoring systems, which are often labor-intensive, time-consuming, and prone to human error.

The core of the proposed system is a video processing module that utilizes background subtraction techniques along with contour analysis to detect moving vehicles from live traffic footage or pre-recorded video feeds. To enhance the accuracy of detection, the system incorporates image pre-processing steps such as grayscale conversion, Gaussian blurring, and morphological transformations. A virtual line is defined across the frame, and each time a vehicle crosses it, the system updates the counter. This approach ensures that vehicles are not counted multiple times.

For environments requiring higher precision, the system can be extended with advanced machine learning or deep learning models such as YOLO (You Only Look Once) or SSD (Single Shot Detector), which can accurately detect and classify various types of vehicles in real-time. The proposed system is scalable, cost-effective, and adaptable for integration into smart city infrastructures, traffic management systems, and toll collection setups.

By minimizing manual intervention and increasing accuracy, this system not only improves operational efficiency but also contributes to better traffic planning and congestion control. It is designed to be lightweight and deployable on both edge devices and cloud platforms, ensuring flexibility across various use cases and deployment scenarios.

## V. CONCLUSION

The Traffic Vehicle Detection and Counting System developed using machine learning techniques demonstrates an effective solution for real-time traffic monitoring. By leveraging computer vision and background subtraction algorithms, the system can accurately detect and count moving vehicles in video footage. The integration of OpenCV for image processing and Python for logic implementation ensures that the project is both efficient and scalable for various use cases, such as traffic analysis, smart city management, and congestion control.

The system's core strength lies in its simplicity and robustness. It can operate on pre-recorded videos or live camera feeds, making it adaptable for deployment in urban or semi-urban traffic environments. The use of bounding boxes and center point tracking allows for precise detection, while the counting line logic minimizes duplicate entries. Extensive testing has shown that the system performs reliably under different conditions such as varying lighting, background complexity, and traffic density. Additionally, its low-cost implementation using open-source tools adds to its practical applicability.

In conclusion, this project provides a strong foundation for more advanced traffic monitoring systems. It can be extended further by integrating deep learning models like YOLO or SSD for better accuracy and detection of vehicle types. Real-time analytics, integration with cloud services, and deployment on embedded systems such as Raspberry Pi or Jetson Nano are promising directions for future development. Ultimately, the system contributes to intelligent transportation solutions aimed at improving traffic management, safety, and data-driven infrastructure planning.

## REFERENCES

- 1)Bradski, G. (2000). The OpenCV Library. Dr. Dobb's Journal of Software Tools.
- 2)Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 779-788.
- 3)Viola, P., & Jones, M. (2001). Rapid Object Detection using a Boosted Cascade of Simple Features. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 511-518.
- 4)Dalal, N., & Triggs, B. (2005). Histograms of Oriented Gradients for Human Detection. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 886-893.
- 5)Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. Advances in Neural Information Processing.





INTERNATIONAL  
STANDARD  
SERIAL  
NUMBER  
INDIA



# INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

 9940 572 462  6381 907 438  [ijirset@gmail.com](mailto:ijirset@gmail.com)



[www.ijirset.com](http://www.ijirset.com)

Scan to save the contact details