Real-Time Vehicle Detection and Counting System Using Background Subtraction and Image Processing Techniques

NIMMANA MUTYALU

PG Scholar. Department M.Sc(CS), DNR College, Bhimavaram, Andhra Pradesh

K.Venkatesh

Lecturer in M.Sc(CS), DNR College, Bhimavaram, Andhra Pradesh

ABSTRACT

Traffic monitoring has become a critical requirement in modern cities due to rapid urbanization and increased vehicular movement. Traditional manual traffic management struggles to provide accurate and real-time vehicle counts, leading to congestion, delayed responses, and inefficient road usage analysis. To address these challenges, this project presents a Real-Time Vehicle Detection and Counting System based on background subtraction and classical image-processing algorithms using OpenCV.

The proposed system processes live or recorded video feeds to automatically detect and count vehicles passing a defined line on the road. The approach begins with video acquisition from a webcam or MP4 file. Each frame undergoes preprocessing, including grayscale conversion and Gaussian blurring, to remove noise and enhance image quality. A background subtraction algorithm (MOG – Mixture of Gaussians) isolates moving objects from the static background, enabling accurate detection of vehicles.

After background modeling, the system applies morphological operations such as dilation and closing to eliminate shadows, fill gaps, and improve foreground masks. Contours are extracted from the processed frames to identify potential vehicles. Bounding boxes are drawn around detected vehicles only if they meet defined height and width thresholds, eliminating false detections like small objects or noise. A centroid-based tracking approach is used to determine when a vehicle crosses a predefined counting line. When the centroid intersects the line, the system increments the vehicle count, ensuring accurate and efficient tracking even in noisy environments.

The counting mechanism incorporates a small offset range to improve detection accuracy and allow minor variations in centroid positions. The real-time display shows the live video with bounding boxes, centroid markers, and the counting line. The continuously updated vehicle count is printed both on the screen and in the terminal, allowing real-time monitoring.

This system reliably handles dynamic changes, shadows, partial occlusions, and perspective distortions. It can be deployed for traffic flow analysis, smart parking systems, intelligent

transportation management, and automated toll collection. The simplicity and efficiency of the method make it suitable for real-world applications where resource constraints prevent the use of heavy deep-learning models. Overall, the project demonstrates an effective computer-vision–based solution for automated traffic analysis.

Keywords: Vehicle Detection, Vehicle Counting, Background Subtraction, Computer Vision, OpenCV, Traffic Monitoring, Object Tracking, Contour Detection, Intelligent Transport Systems

I. INTRODUCTION

Traffic monitoring and vehicle counting play an essential role in intelligent transportation systems, urban planning, and law enforcement. With increasing population and vehicle density, manual traffic data collection has become impossible to handle efficiently. Automated systems offer significant advantages by providing continuous monitoring, real-time data, and improved accuracy. Vehicle counting is fundamental for analyzing traffic flow, optimizing routes, planning infrastructure, and responding to congestion.

Computer vision has emerged as a powerful alternative to conventional sensor-based systems like inductive loops, radar, or infrared sensors. These physical sensors require installation in or near road surfaces, increasing maintenance costs and creating disruption during installation. In contrast, vision-based systems perform traffic analysis using existing CCTV infrastructure, offering flexible and cost-effective solutions.

This project focuses on building a real-time vehicle detection and counting system using classical image-processing methods rather than computationally heavy deep-learning models. The system utilizes OpenCV's background subtraction technique, which is effective for detecting moving objects in video sequences. By analyzing pixel changes across frames, moving vehicles can be isolated from the static background.

The system applies grayscale conversion, Gaussian filtering, dilation, and morphological closing to refine the vehicle shapes and remove unwanted noise. It then extracts contours representing moving vehicles. Only contours that meet a minimum size requirement are processed further, ensuring that small irrelevant objects are filtered out. The vehicles are tracked by calculating the center point (centroid) of each detected contour. A counting line is placed horizontally across the frame, and each time a centroid crosses this line, the vehicle count is incremented.

This approach allows efficient tracking even without complex object-tracking algorithms. The system runs in real time and works with standard video footage, making it applicable in highways, city roads, parking lots, and surveillance systems.

The simplicity, accuracy, and flexibility of this system make it a strong candidate for practical vehicle monitoring solutions in developing intelligent transportation infrastructures.

II. LITERATURE SURVEY (WITH EXISTING METHODS)

1. Traditional Traffic Counting Techniques

Earlier traffic monitoring systems relied on hardware-based sensors such as inductive-loop detectors, pneumatic tubes, and infrared sensors. While reliable, they are expensive to install and maintain and prone to failure under harsh weather conditions.

2. Computer Vision Approaches

With advancements in image processing, video-based systems became popular:
Background Subtraction: Used to detect moving foreground objects. Algorithms like MOG, MOG2, and KNN are widely adopted.

Optical Flow: Used to detect motion patterns between frames but computationally heavy.
Contour Detection and Blob Analysis: Helps in identifying object boundaries and shapes.

3. Background Subtraction Models MOG (Mixture of Gaussians): Learns background statistically; accurate but sensitive to lighting.

MOG2: Improved version that adapts to background changes.
KNN Subtractor: Suitable for dynamic environments.
Your project uses MOG, which is appropriate for structured environments.
4. Object Tracking Methods
Traditional tracking includes:

Centroid tracking
Kalman filtering
Mean shift and Camshift
Optical flow tracking

Your system uses centroid tracking, a lightweight and effective method.

5. Deep Learning-Based Systems

Recent methods use:

YOLOv5/v8
SSD (Single Shot Detector)
Faster R-CNN

Though accurate, these require high computational resources and GPUs.

6. Comparisons with Existing Systems

Classical image-processing techniques are:

Faster
Require no training
Suitable for embedded systems
Lower computational cost

Deep learning models:

More accurate
Robust to varying weather
Require high-performance hardware

Your system strikes a balance by using classical methods for speed and simplicity.

III. EXISTING SYSTEM

Existing vehicle detection and counting systems used in traffic monitoring generally fall into two categories: sensor-based systems and vision-based systems.

Sensor-based systems include loops, radars, and infrared detectors placed in or above the road. Although widely used, these systems are expensive, require road closures for installation, and have high maintenance costs. They also provide only limited information, such as vehicle count, without visual verification.

Vision-based systems have improved significantly with CCTV expansion. However, many traditional video-processing systems struggle with real-time performance, noise, lighting variations, and false detections. Basic motion-detection systems often count shadows or background movement as vehicles, reducing accuracy. Other existing solutions use optical flow, which is computationally heavy, making them unsuitable for low-resource hardware.

Deep-learning–based vehicle detection methods like YOLO or Faster R-CNN offer high accuracy but require large datasets, long training time, and GPU resources, making them impractical for lightweight deployments or simple roadside monitoring.

The proposed system overcomes these limitations using classical image-processing techniques such as background subtraction, morphological filtering, contour detection, and centroid tracking. These methods run in real time on standard computers without specialized hardware. The system provides accurate vehicle detection, ignores noise and small objects, and counts vehicles only when they cross a specific line. This makes it highly efficient for real-world traffic analytics with minimal computational cost.

IV. PROPOSED METHOD

The proposed system is a real-time vehicle detection and counting framework that uses classical computer vision techniques to automatically track vehicles in a video stream. The main objective is to create an efficient, lightweight, and accurate system capable of detecting moving vehicles and counting them as they cross a designated line. Unlike deep learning–based solutions that require massive computational resources, training datasets, and GPUs, this system offers fast processing and simplicity using background subtraction and contour-based detection.

The system processes each frame of the input video and applies a sequence of operations including grayscale conversion, Gaussian blurring, background subtraction, dilation, and morphological closing. These steps isolate moving objects (vehicles) from the static background, making it easier to identify contours that represent vehicles. A minimum width and height threshold ensures that only valid vehicles are detected, reducing false positives from noise, small objects, or shadows.

A key feature of the system is the counting mechanism, which uses centroid tracking. For every detected bounding box, the centroid is calculated. When the centroid crosses a predefined counting line within an allowed pixel offset, the vehicle count increases by one. This method allows reliable counting even under partial occlusion, frame delays, or irregular motion.

The proposed system is highly suitable for real-world traffic monitoring scenarios such as highways, city intersections, toll booths, parking lots, and surveillance systems. It can run efficiently on low-cost hardware without requiring machine-learning models or cloud infrastructure. The modular design allows future integration with object classification or deep-learning detectors if needed.

V. IMPLEMENTATION

The implementation of the vehicle detection and counting system is divided into several stages: video acquisition, preprocessing, background subtraction, contour detection, centroid tracking, and counting logic. The entire workflow is implemented using OpenCV and NumPy.

1. Video Acquisition The system reads input from a video file (Video.mp4) using cv2.VideoCapture. It can also be modified to work with a live webcam feed. Each frame is captured inside a continuous loop until the user exits.

2. Preprocessing To improve detection accuracy, each frame is converted to grayscale using cv2.cvtColor. Gaussian blurring is applied to reduce noise and smooth the frame. This makes the background subtraction process more stable and eliminates high-frequency defects.

3. Background Subtraction The system uses OpenCV's MOG background subtractor

(cv2.bgsegm.createBackgroundSubtractorMOG()), which models the background using a mixture of Gaussians. This helps differentiate moving vehicles from the static background.

The output of the subtractor is a binary foreground mask. This mask is further refined using:

Dilation to expand detected regions Morphological closing to fill holes and smooth edges These operations produce cleaner vehicle silhouettes.

4. Contour Detection and Bounding Boxes Contours are extracted from the cleaned mask using cv2.findContours. For each contour, a bounding rectangle is computed. A minimum width and height threshold ensures only vehicles are considered. Bounding boxes are drawn on the video frame, and the vehicle count is displayed.

5. Centroid Tracking The center of each bounding box is calculated using a custom center_handle function. The centroid is added to a list of tracked points. Circles are drawn to visualize these centroids

.6. Counting Logic A horizontal counting line is placed at a specific y-coordinate (count_line_position). When any centroid crosses this line within a tolerance (offset = 6), the global counter increments. The line changes color to indicate detection.

7. Output Display The processed frame is displayed in real time using cv2.imshow. The final vehicle count is shown prominently on the video. The loop continues until the user presses Enter (ASCII 13).

This implementation runs efficiently in real time and requires only basic CPU processing.

VI. ALGORITHMS

1. Background Subtraction Algorithm

Input: Video frame
Steps:

Convert frame to grayscale
Apply Gaussian blur
Pass blurred frame to MOG subtractor
Obtain binary foreground mask
Apply dilation to strengthen shapes
Apply morphological closing to remove holes
Output: Clean foreground mask with moving vehicles
2. Contour Detection Algorithm

Steps:

Extract contours from the processed mask
For each contour:
Compute bounding rectangle
Check if width ≥ threshold and height ≥ threshold
Filter out small/irrelevant contours
Output: Valid vehicle regions
3. Centroid Calculation Algorithm

Input: Bounding box (x, y, w, h)
Steps:

Compute centroid:

cx = x + w/2
cy = y + h/2
Append centroid to detection list
Output: Vehicle centroid
4. Vehicle Counting Algorithm

Steps:

For each centroid:
Compare cy with counting line position
Allow ±offset tolerance
If centroid crosses the line:
Increase counter
Highlight line
Remove centroid from list
Output: Accurate vehicle count

These algorithms work together to detect, track, and count vehicles with minimal computational load.

VII. SYSTEM DESIGN

1. Architectural Overview

The system follows a frame-by-frame processing architecture consisting of:

Input Module
Preprocessing Module
Background Subtraction Module
Feature Extraction & Contour Module
Centroid Tracking Module
Counting Module
Output Display Module

Each frame passes sequentially through these modules.

2. Module Descriptions
a. Input Module

Captures video frames via OpenCV. Supports both pre-recorded videos and live streams.

b. Preprocessing Module

Performs:

Grayscale conversion
Gaussian blur
Its purpose is to remove noise and prepare the frame for background modeling.
c. Background Subtraction Module

Using MOG subtractor, this module identifies moving objects. It outputs a binary foreground mask highlighting vehicles.

d. Morphological Filtering Module

Improves the foreground mask using dilation and closing operations.

e. Contour Detection & Bounding Box Module

Extracts contour regions and draws rectangles around potential vehicles. Applies thresholding to eliminate noise.

f. Centroid Tracking Module

Calculates center points for each valid bounding box. These points are monitored across frames.

g. Counting Module

Defines a counting line. When a centroid crosses the line, the count increments.

h. Output Display Module

Renders:

Bounding boxes
Counting line
Centroids
Vehicle count

The final video is displayed in a window.

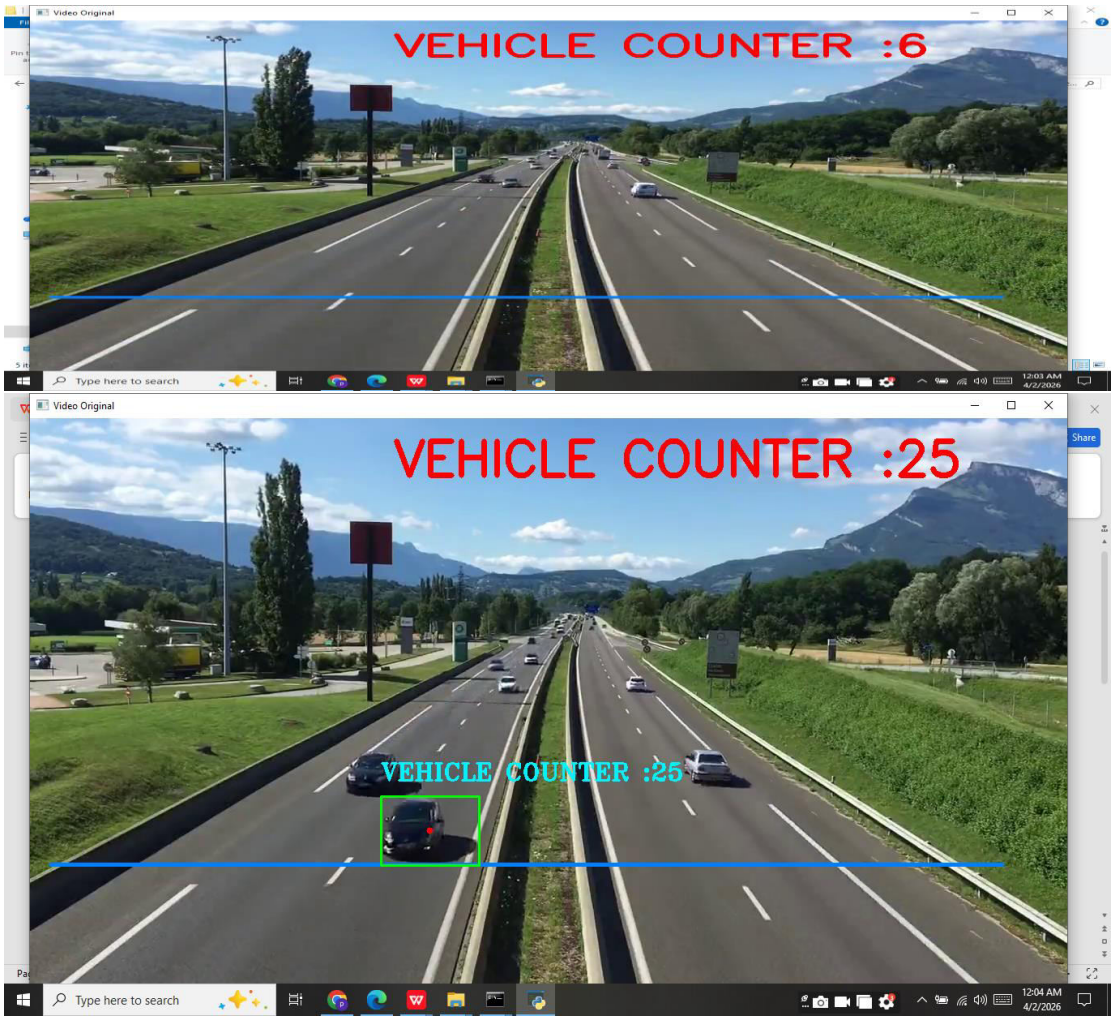
3. Data Flow Design

Video → Preprocessing → Background Subtraction → Morphological Filtering → Contour Detection → Centroid Tracking → Counting Logic → Output

4. System Characteristics

Real-time processing: Achieved through optimized OpenCV operations
Low computational cost: No machine learning training required
Scalable: Can be extended with deep learning detectors
Modular: Each component can be independently upgraded

SYSTEM DESIGN IMAGES



VIII. CONCLUSION

The vehicle detection and counting system presented in this project demonstrates an efficient and practical computer-vision–based solution for automated traffic monitoring. By combining background subtraction, contour detection, and centroid-based tracking, the system can accurately detect and count vehicles in real time without requiring complex machine-learning models. This makes the solution ideal for deployment in cities, highways, and parking areas where real-time monitoring is essential.

The system operates effectively even with standard computing hardware, making it highly cost-effective. It eliminates the need for expensive sensors or GPU-powered deep-learning frameworks. Its modular structure ensures flexibility, allowing for easy enhancements such as vehicle classification, speed estimation, or integration with intelligent transportation systems.

Overall, this project demonstrates the potential of classical image-processing techniques in solving real-world traffic management challenges. It highlights how well-designed algorithms, paired with optimized OpenCV functions, can achieve reliable and accurate performance in dynamic environments. The system provides a strong foundation for future research and real-world deployment in smart city infrastructures.

REFERENCES

[1] R. Gonzalez and R. Woods, Digital Image Processing, Pearson, 2018.
[2] M. Piccardi, "Background subtraction techniques," IEEE SMC, 2004.
[3] C. Stauffer and W. Grimson, "Adaptive background mixture models," CVPR, 1999.
[4] OpenCV Documentation, "Background Subtraction."
[5] D. Forsyth and J. Ponce, Computer Vision: A Modern Approach, 2011.
[6] K. Kim et al., "Real-time vehicle detection," IEEE IV, 2008.
[7] A. Sobral, "BGSLibrary: Background subtraction library," 2013.
[8] P. Dollar et al., "Contour detection and segmentation," PAMI, 2006.
[9] B. Zhan et al., "Traffic monitoring using computer vision," IEEE ITS, 2008.
[10] N. Buch, S. Velastin, and J. Orwell, "A review of computer vision techniques for traffic," IEEE ITS, 2011.
[11] S. Elhabian et al., "Moving object detection techniques," 2008.
[12] A. Vakili and N. Kazemi, "Vehicle counting using OpenCV," IJCSIT, 2020.
[13] Y. Benezeth et al., "Background modeling," CVPR, 2010.
[14] M. Rad et al., "Real-time vehicle tracking," IEEE Access, 2019.
[15] S. Sivaraman and M. Trivedi, "Looking at vehicles: Vision-based monitoring," IEEE Trans. ITS, 2013.