

A centroid-based algorithm for measuring and tracking vehicle speed from a monocular camera using the YOLOv8 object detector

Pankaj Kumar Gautam, Sanjeev Kumar

Department of Computer Science and Engineering, United University, Prayagraj, Uttar Pradesh, India

Article Info

Article history:

Received May 10, 2024

Revised Jun 21, 2025

Accepted Jul 4, 2025

Keywords:

Computer vision

Neural networks

Object detection

Speed estimation

Target tracking

YOLO

ABSTRACT

Accurate real-time vehicle speed measurement is crucial for enhancing road safety and advance intelligent transportation systems (ITS). This paper proposes a centroid-based tracking algorithm that integrates YOLOv8, a state-of-the-art object detector, with DeepSORT for robust multi-object tracking. By leveraging YOLOv8's anchor-free detection and DeepSORT's appearance-based association, the proposed method effectively mitigates occlusions and minimizes identity switches. Evaluations on the VS13 benchmark dataset reveal a 2–5% improvement in measurement accuracy as compared to existing solutions, while maintaining real-time performance at 30 FPS. The method demonstrates consistent reliability across different vehicle models, speeds, and lighting conditions, underscoring its adaptability to real-world traffic scenarios. Moreover, larger bounding boxes enhance tracking stability, reducing false detections. Overall, the approach's low computational overhead and high accuracy position it as a practical solution for ITS applications in constrained environments.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Pankaj Kumar Gautam

Department of Computer Science and Engineering, United University

Prayagraj, Uttar Pradesh, India

Email: pankajgautam257@gmail.com

1. INTRODUCTION

With the rapid urbanization and expansion of transportation networks, vehicular congestion has become a pressing challenge in modern cities. Increasing traffic density not only contributes to longer commute times but also raises serious concerns about road safety. According to study from Liu and Shetty [1], traffic congestion leads to significant delays, increased fuel consumption, and higher emission levels, making it a crucial area for research and policy intervention. More importantly, road accidents remain one of the leading causes of fatalities worldwide, incurring substantial economic losses. The Association for Safe International Road Travel (ASIRT) reports that traffic-related incidents account for a considerable percentage of premature deaths globally. As such, effective traffic management and speed control measures are essential for ensuring public safety and optimizing urban mobility. Traditional methods of speed estimation, such as radar-based and inductive loop detection systems, have been widely used in traffic monitoring. However, these techniques often suffer from high installation costs, limited coverage, and susceptibility to environmental interference. Jakus and Coe [2] explored speed measurement through doppler effect analysis in vehicular noise, highlighting an alternative approach to speed detection. With advancements in artificial intelligence (AI) and smart city initiatives, computer vision has emerged as a transformative tool for real-time traffic monitoring. Deep learning-based

object detection and tracking algorithms now enable accurate and efficient vehicle identification across video sequences.

In recent years, deep learning models such as convolutional neural networks (CNNs) have significantly improved object detection and tracking capabilities in real-world applications. Zhao *et al.* [3] reviewed deep learning-based object detection techniques, emphasizing the effectiveness of CNN architectures like AlexNet, VGG, and ResNet in recognizing and localizing objects. Among these advancements, the you only look once (YOLO) algorithm, has gained widespread adoption due to its real-time processing capabilities and high detection accuracy. The integration of YOLO with other sensor technologies further enhances vehicle tracking and speed estimation accuracy. Han *et al.* [4] proposed a fusion-based approach that combines LiDAR and camera data using an improved YOLO algorithm, demonstrating superior object detection performance in complex environments. Similarly, Lin and Sun [5] introduced a YOLO-based traffic counting system, showcasing its potential in intelligent traffic management solutions.

Given these advancements, this paper focuses on developing a real-time vehicle speed estimation system using YOLO-based object detection techniques. The proposed system leverages video-based tracking to estimate vehicle velocity accurately, contributing to safer and more efficient traffic management. The paper is organized as follows: section 2 provides comprehensive background information, reviews existing literature on traffic monitoring systems, highlights key advancements, and identifies research gaps. Section 3 provided the methods and algorithm, detailing state-of-the-art YOLO-based object detection and tracking approaches along with the datasets used for evaluation. Section 4 has proposed methodology followed by section 5 that describes the results and discussion, including the speed measurement techniques, and tracking mechanisms. Section 6 presents the experimental setup, performance evaluation, and obtained results. Finally, section 7 discusses the findings, compares them with previous studies, with implications for future research and practical implementations in intelligent traffic management followed by the conclusion.

2. THEORETICAL BACKGROUND

Automated vehicle detection, speed estimation, and tracking have become crucial in intelligent transportation systems. Various techniques, including deep learning-based object detection, traditional computer vision algorithms, and hybrid approaches, have been developed to enhance real-time performance and accuracy. The rise of YOLO, Kalman Filters, DeepSORT, faster R-CNN, and other machine learning models has significantly improved the ability to analyze traffic flow, detect vehicles, and estimate speed with high precision. UAV-based surveillance and single-camera tracking further expand the scope of real-time traffic monitoring. Recent advancements in vehicle detection and speed estimation have leveraged deep learning and computer vision techniques to improve traffic monitoring accuracy. Traditional methods, such as CVS-based motion analysis, suffer from noise sensitivity and are less effective in dynamic environments. The YOLO-based frameworks [6]-[8] have emerged as dominant solutions due to their real-time object detection and tracking capabilities. Researchers have further enhanced these models by integrating Kalman filters [9], DeepSORT tracking, and feature matching techniques [10] to improve the robustness of vehicle tracking. Studies such as [11]-[13] have utilized drone-based video analysis to estimate vehicle speed from an aerial perspective, while others have incorporated 3D convolutional networks and semantic fusion techniques to enhance detection precision across multiple camera angles.

Despite these advancements, existing models face challenges such as occlusion handling, varying illumination conditions, computational efficiency, and real-time deployment constraints. Methods relying on deep learning [14] are computationally intensive and require extensive training datasets for reliable performance. Occlusion and environmental factors [15], [16] impact real-time tracking accuracy, especially in dense urban traffic. Multi-camera approaches [17]-[19] improve accuracy but introduce synchronization challenges. Many studies have focused on optimizing feature extraction, multi-camera tracking, and sensor fusion to enhance accuracy. However, research gaps remain in robust real-time inference and scalability for large-scale deployments. Addressing these issues through efficient computational techniques, improved tracking algorithms, and dataset augmentation can further refine vehicle speed estimation systems, making them more suitable for real-world traffic management applications.

2.1. Challenges and research gaps

Despite significant progress, multiple challenges hinder real-time and scalable deployment of vehicle detection and speed estimation systems. Firstly, occlusion and varying lighting conditions significantly impact

detection accuracy, especially in urban areas with high vehicle density. Techniques such as feature fusion and sensor-based augmentation are still underexplored for mitigating these issues. Additionally, models like YOLO and faster R-CNN require extensive computational resources, making them unsuitable for lightweight, embedded systems.

Additionally, real-time inference remains a bottleneck. While techniques like DeepSORT and Kalman filters improve object tracking, they still struggle with dynamic motion prediction and error accumulation over time. The lack of standardized datasets and benchmarking frameworks further limits the generalizability of existing solutions. Moreover, UAV-based traffic monitoring faces challenges in perspective correction and object localization, affecting speed estimation accuracy. Table 1 presents some techniques of speed estimation and their limitations and drawbacks.

Table 1. Comparative analysis of vehicle detection and speed estimation techniques

Author(s)	Techniques Used	Complexity	Methodology	Input type	Limitations	Application
C.-J. Lin <i>et al.</i> [6]	YOLO, Virtual Detection Zone	High	Object detection & region-based classification	CCTV, UAV	Limited accuracy in dense traffic	Vehicle counting, speed estimation
Z. Chen <i>et al.</i> [7]	YOLOv5-based detection	High	Deep learning-based object detection	High-resolution UAV images	Requires high computational power	Aerial vehicle detection
G. R. Arash <i>et al.</i> [8]	CVS-based detection	Medium	Motion vector analysis	Video sequences	Susceptible to noise	Traffic monitoring
A. H. Rais & R. Munir [9]	YOLO, Kalman Filter, Frame Sampling	High	Object tracking & state estimation	Traffic camera	Requires manual parameter tuning	Real-time speed tracking
J. Azimjonov & A. Özmen [10]	CNN, Feature Matching	High	Hybrid tracking and speed monitoring	Highway surveillance	High computational load	Traffic flow monitoring
T. Rahman <i>et al.</i> [11]	Deep Learning (YOLO), Drone Video	High	Frame-based speed estimation	Aerial footage	Accuracy depends on camera angle	UAV-based traffic analysis
P. H. Nguyen & M. B. Duy [12]	YOLOv4, DeepSORT	High	Object detection and multi-object tracking	Highway camera	Requires extensive dataset training	Speed tracking
H. Dong <i>et al.</i> [13]	3D ConvNets, Non-local Blocks	Very High	Feature extraction & motion detection	Traffic surveillance	High latency	Autonomous traffic systems
T.-H. Wu <i>et al.</i> [14]	YOLOv5, Distance Detection	High	Bounding box-based estimation	Surveillance footage	Sensitive to occlusion	Real-time vehicle tracking
P. Hurtik <i>et al.</i> [15]	Poly-YOLO (YOLOv3 variant)	Medium	Instance segmentation & detection	Urban traffic images	High false positives in night-time	Smart city monitoring
U. Mittal <i>et al.</i> [16]	Faster R-CNN + YOLO	High	Hybrid ensemble model	Road traffic video	Computationally expensive	Traffic density estimation
Z. Tang <i>et al.</i> [17]	Visual & Semantic Feature Fusion	Very High	Multi-camera tracking & 3D speed estimation	Inter-camera feed	Requires synchronization	Intelligent traffic systems
P. Giannakeris <i>et al.</i> [18]	Abnormality Detection from Surveillance	Medium	Feature extraction & anomaly analysis	Surveillance feed	Limited scalability	Traffic incident monitoring
S. Hua <i>et al.</i> [19]	Vehicle Tracking via Traffic Videos	High	Video-based tracking	Traffic footage	Requires high-quality video feed	Real-time highway monitoring

3. MATERIALS AND METHODS

The YOLO [20] framework revolutionized object detection by formulating it as a single-stage regression problem, where a CNN directly predicts bounding boxes and class probabilities from an input image in a single forward pass. This architecture contrasts with traditional multi-stage detection methods, such as Region-based CNNs (R-CNNs), which require separate region proposal and classification steps, making YOLO significantly faster while maintaining competitive accuracy.

3.1. YOLO Model and Its variant

Following its initial introduction, YOLO9000 was a pivotal upgrade by Redmon and Farhadi [21], incorporating a multi-scale detection approach and joint training on classification and detection datasets. Subsequent iterations, including YOLOv3, YOLOv4, and YOLOv5, have progressively improved accuracy, robustness, and computational efficiency through optimizations such as anchor-based predictions, feature pyramid networks (FPNs), spatial pyramid pooling (SPP), and cross-stage partial networks (CSPNet). The evolution of YOLO underscores its adaptability and impact on real-time applications, including autonomous driving, surveillance, and UAV-based monitoring. By enabling high-speed detection with minimal computational overhead, YOLO remains a cornerstone in advancing computer vision research and addressing the growing demands for dynamic, high-performance detection systems.

In the evolving landscape of object detection, YOLOv8 offers notable advancements over its predecessors, making it a compelling choice for current research. YOLOv8 improves accuracy through a more advanced backbone and neck architecture, enhances speed with optimized network design, and refines object detection capabilities with better multi-scale detection and bounding box regression. It also excels in handling complex

scenes with reduced false positives and missed detections, thanks to improved contextual understanding. Additionally, YOLOv8 integrates advanced training techniques, such as self-supervised learning and enhanced transfer learning, while being more model-efficient with reduced size and optimized resource usage. These advancements address the limitations of earlier versions, providing a robust and efficient solution for real-time object detection tasks. The YOLO object detection framework has emerged as a ground-breaking approach in computer vision, offering a unified, end-to-end solution for real-time object detection.

Table 2 provides a clear and comprehensive overview of the YOLO models, showcasing their evolution, architectural changes, and improvements over time. The YOLOv8 architecture consists of two main components: the backbone and the head. The backbone is a modified version of CSPDarknet53, incorporating 53 convolutional layers to extract high-level feature representations from input images. This feature extraction is crucial for effective object detection. The head, following the backbone, includes multiple convolutional layers and a series of fully connected layers responsible for bounding box (BBox) prediction, score estimation, and class probability determination for each object within an image. YOLOv8 processes input images of size 640x640 pixels. For loss functions, it utilizes complete intersection over union (IoU) and distribution focal loss (DFL) for accurate bounding box regression, while employing binary cross-entropy (BCE) for class prediction. This architecture enhances YOLOv8's ability to perform complex vision tasks with high accuracy and efficiency.

Table 2. Overview of the YOLO models

Version	Release year	Key features	Architecture	Improvements
Initial YOLO	2015	24 convolutional layers, 2 fully connected layers, based on GoogleNet	GoogleNet	Issues with accurate positioning and recall rate
YOLO v2	2016	Batch normalization, anchor boxes, high-resolution classifiers	DarkNet-16	Faster detection, improved accuracy
YOLO v3	2018	Multi-scale classification, independent logistic classifiers, DarkNet-53 backbone	DarkNet-53	Enhanced detection, more efficient processing
YOLO v4	2020	Mosaic data augmentation, new loss function, anchor-free detection, "Bag-of-Freebies" and "Bag-of-Specials"	DarkNet	Fastest and most accurate among YOLO versions
YOLO v5	2020	Hyperparameter optimization, auto-learning bounding box anchors	PyTorch	Similar size to YOLO v4, improved speed
YOLO v6	2022	Variants: Nano, Tiny, Small for reduced memory and improved speed	Head, Bottleneck, Backbone Network	Suited for various applications
YOLO v7	2022	Pose estimation on COCO key points datasets	HBB Network	Enhanced object detection with pose information
YOLO v8	2023	Advanced flexibility and performance for vision-based AI tasks	HBB Network	Improved detection accuracy and model adaptability

3.2. Object tracking

Vehicle tracking in video sequences is essential for monitoring positions and trajectories, enabling speed estimation, direction analysis, and behavioral assessment. The efficiency of tracking algorithms depends on factors such as frame rate (FPS), traffic density, and vehicle size. While optical flow and pixel-based methods offer real-time insights, Kalman filter-based trackers, despite their effectiveness in motion prediction, suffer from scalability issues due to their $O(n^3)$ computational complexity. As the number of vehicles increases, Kalman filters struggle with maintaining accuracy, often leading to false positives and tracking inconsistencies.

To address these limitations, simple and real-time tracker (SORT), introduced by Bewley *et al.* [22], combines Kalman filtering with the Hungarian algorithm to optimize frame-to-frame object matching. SORT tracks objects based on their center coordinates and bounding box dimensions, offering fast and efficient trajectory estimation. However, it lacks appearance-based features, making it vulnerable to ID switching when multiple objects with similar motion characteristics are present. This limitation reduces its robustness in long-term multi-object tracking, particularly in crowded and occluded environments where maintaining consistent identity across frames is critical.

In (1) shows the state vector of the detected object, where h and s represent the center coordinates of the detected object. s and r represent the bounding box (BBox) area size. \hat{h} and \hat{v} represent the vertical and horizontal axes movement of the object center, and \hat{a} represents the change in BBox area.

$$X = [h, v, s, r, \hat{h}, \hat{v}, \hat{a}] \quad (1)$$

The DeepSORT [23] algorithm represents an enhancement over SORT by incorporating appearance information to reduce ID switches by approximately 45%. It improves the matching process by replacing the Intersection over Union (IoU) cost matrix-based mechanism with a combination of cascade matching and

IoU-based matching, enabling it to effectively track objects that have been occluded for extended periods. Additionally, StrongSORT [24] and ByteSORT [25], which are based on real-time YOLO object detectors, demonstrate strong performance in tracking applications.

BBox-based tracking algorithms, which rely on tracking two consecutive frames using BBox vector information, involve a process where BBox data—including the set of all bounding boxes in the current frame with their respective height and width—is used. The movement of these bounding boxes between successive frames is tracked by first measuring similarity between consecutive frames using Euclidean distance, creating a 2D distance array. This array is then sorted row-wise and minimized by column IDs. Trackers update vehicle BBox information based on these IDs, with the distance threshold for minimization being user-defined and dependent on observed vehicle speed, whether high or low. Table 3 provides the critical comparison of tracking algorithms.

Table 3. Critical examination of the tracking algorithms SORT, DeepSORT, StrongSORT, and ByteTrack based on key evaluation criteria

Algorithm	Complexity	Speed	Performance	Limitations	Positive aspects	Parameters considered
SORT [22]	$O(n^3)$ (Hungarian algorithm)	Very Fast (~260 FPS)	Moderate accuracy in low-density scenarios	High ID switching, lacks appearance features	Efficient for real-time applications, simple and lightweight	IoU-based matching, BBox tracking
DeepSORT [23]	Higher than SORT (due to feature extraction)	Fast (~50 FPS)	~45% reduction in ID switching	Computationally expensive due to feature embedding	Incorporates appearance features, improving re-identification	Cascade matching, Mahalanobis distance, CNN embeddings
StrongSORT [24]	Higher than DeepSORT	Moderate (~100 FPS)	Improved accuracy in crowded scenes	Increased computational cost due to advanced re-ID models	Enhanced feature matching, occlusion handling	Gated Recurrent Units (GRU), Re-ID network, IoU matching
ByteTrack [25]	Moderate (Optimized Association Algorithm)	Fast (~160 FPS)	High accuracy in associating all detection boxes	May be sensitive to low-quality detections	Handles missing detections, robust to occlusion, better performance on MOT benchmarks	Confidence thresholding, motion-based tracking

ByteTrack emerges as the most robust algorithm, balancing speed and accuracy efficiently. However, StrongSORT provides superior identity preservation in dense environments. While DeepSORT remains a strong choice for person re-identification, SORT is best suited for lightweight, high-speed tracking with minimal hardware requirements. The selection of this algorithm is based on the trade-offs between speed, accuracy, and computational efficiency depending on the application context.

In (2)–(4) shows the bounding box representation of two consecutive frames i and j , where m and n are the sets of all bounding boxes of objects in the current frame, with width W and height H , respectively. D_{ij} represents the movement of the bounding box in successive frames i and j .

$$B_i = \begin{bmatrix} X_0^{center}, Y_0^{center}, W_0, H_0 \\ \vdots \\ X_n^{center}, Y_n^{center}, W_n, H_n \end{bmatrix} \quad (2)$$

$$B_j = \begin{bmatrix} X_0^{center}, Y_0^{center}, W_0, H_0 \\ \vdots \\ X_m^{center}, Y_m^{center}, W_m, H_m \end{bmatrix} \quad (3)$$

$$D_{ij} = \sqrt{\sum_{i=0}^m \sum_{j=0}^n (B_i - B_j)^2}, \quad i \in [0 : m], \quad j \in [0 : n] \quad (4)$$

3.3. Available datasets

The Table 4 provides an overview of various datasets commonly used for traffic monitoring, object tracking, and video-based vehicle detection research. These datasets vary in terms of resolution, frame rate, duration, and specific focus areas, making them suitable for different applications such as real-time traffic analysis, urban surveillance, and object detection in dynamic environments.

Some datasets, like artificial intelligence CIRT and BrnoCompSpeed, focus on real-world traffic scenarios, while others, such as ImageNet VID, are designed for broader object detection tasks. The inclusion of annotated datasets, such as Davis and CDnet2014, further supports moving object detection research, while VS13 audio/video annotated provides a controlled setting for vehicle tracking studies.

Table 4. Overview of traffic monitoring and object detection datasets with key characteristics

Dataset	Resolution	FPS	Duration/size	Description	Special notes
AI City Challenge [25], [26]	Varies	Varies	Varies	Proposed by NVIDIA, annual updates	Regular traffic scenarios; sample sizes vary
BrnoCompSpeed [27], [28]	1920×1080	Varies	Not specified	18 videos with over 20,000 tagged cars	Real-time traffic scenarios
UTFPR [29]	Varies	30	5 hours	Covers 3 lanes in various weather conditions	Speed up to 60 km/hr; low meter-to-pixel ratio
QMUL Junction [30]	360×288	25	1 hour	Video snippets from multiple intersections	Various traffic scenarios
ImageNet VID [31]	Varies	25-30	Not specified	3862 training, 555 validation snippets	Essential for video object detection
Davis & CDnet2014 [32]	Varies	Varies	Not specified	Used for moving object detection	Annotated video snippets
VS13 Audio/Video [33]	Varies	30	10 sec/video	13 vehicle models; 400 snippets	Speed range 30–105 km/hr; controlled setup

4. PROPOSED APPROACH

YOLOv8 offers state-of-the-art object detection capabilities with enhanced accuracy and real-time performance, making it ideal for vehicle detection tasks. It employs an anchor-free detection mechanism, allowing for improved localization of objects, faster inference, and reduced computational overhead. Complementing this, simple online and realtime tracker with a deep association metric (DeepSORT) enhances object tracking by associating detections across frames using deep appearance features and Kalman filtering.

This combination leverages YOLOv8's superior detection accuracy and DeepSORT's robust tracking methodology, ensuring that vehicles are correctly identified and persistently tracked even in occluded or complex traffic scenarios. By applying this hybrid approach to the VS13 dataset, which provides well-annotated vehicle trajectories in controlled environments, we can achieve high-performance tracking with reliable speed estimations and multi-object tracking (MOT) accuracy.

4.1. Proposed algorithm and flow chart

- Extract frames from VS13, normalize, and augment images for robustness.
- Train YOLOv8 on VS13 with COCO pre-trained weights and fine-tune for vehicle detection.
- Apply Kalman filtering for motion prediction and deep association metrics for ID matching.
- Compute mAP, detection accuracy, tracking performance, and ID switches.
- Implement in Python, optimize for real-time speed, and validate on VS13.

In a real-time scenario, vehicle speed is calculated by determining the distance S traveled between two reference lines r_1 and r_2 drawn on the frames. The time t it takes for the vehicle to move between these two lines is calculated using the following formulas given in (5) and (6):

$$t = \frac{N}{fps} \quad (5)$$

$$v = \frac{Ds}{t} \quad (6)$$

where N is the time elapsed between frames, and fps is the frames per second. Here, v represents the vehicle speed, D denotes the pixel displacement distance, and s is the scaling factor used to convert pixel distances into real-world measurements. This scaling factor is essential for transforming pixel measurements into accurate real-world distances.

4.2. Algorithm and flow chart

In this study, we propose a centroid-based tracking algorithm that integrates the DeepSORT tracker with the YOLO v8 object detection system. This approach leverages the strengths of DeepSORT for robust tracking and YOLO v8 for advanced object detection to enhance overall tracking accuracy and efficiency. Algorithm 1 provides a structured overview of the steps involved in the proposed speed estimation algorithm, while Figure 1 illustrates the process flow diagram of the proposed speed estimation algorithm.

Algorithm 1. Structured overview of the steps involved in the proposed speed estimation algorithm using YOLOv8

1. **Input:** Vehicle Video (Frames: N , Distance: $DIST$)
2. **Initialize:** Load YOLOv8, Tracker (TRK), Set $i = 0$
3. **Process:** While $i < N$:
 - (a) **Frame Processing:** For each frame f_i :
 - i. Load tracker TRK , define reference lines REF_1 (entry) and REF_2 (exit)
 - ii. Resize frame to 640×480 , detect objects using YOLOv8
 - iii. Assign unique ID (UNQ_ID) to each detected object
 - iv. Draw bounding boxes (BBox) around detected objects
 - v. Store and update (UNQ_ID , $BBox$) in TRK
 - (b) **Tracking & Speed Calculation:** For each UNQ_ID in TRK :
 - i. Compute BBox centroid, record timestamps at REF_1 (ENT_TIME) and REF_2 (EXT_TIME)
 - ii. Calculate elapsed time: $ELAPSED_TIME = EXT_TIME - ENT_TIME$
 - iii. Compute speed: $SPEED = DIST / ELAPSED_TIME$
 - iv. Store (UNQ_ID , $SPEED$) in a CSV file
4. **Output:** Compare computed speed with benchmark, calculate error.

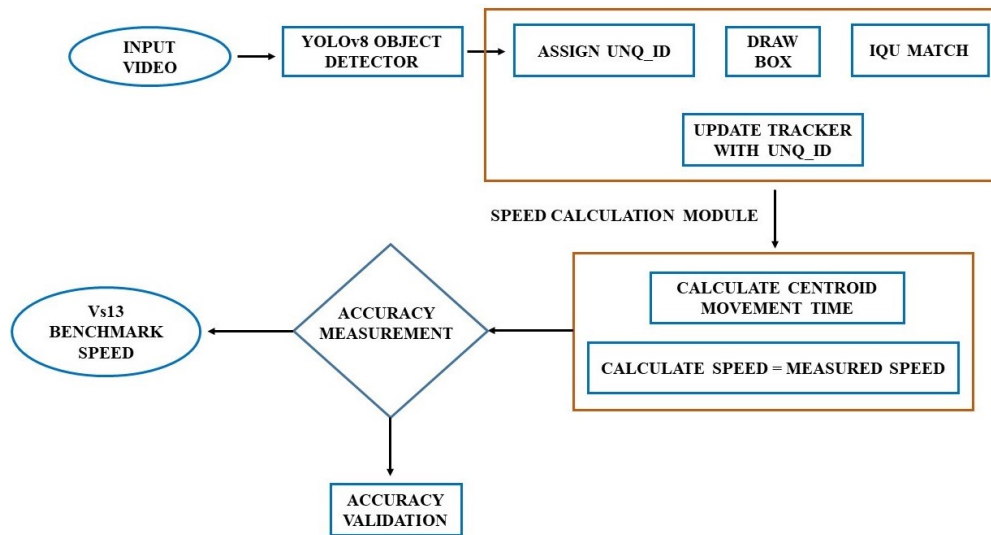


Figure 1. Process flow diagram of proposed speed estimation algorithm

4.3. Datasets and assumptions

Figure 2 illustrates two key elements: panel Figure 2(a) displays screenshots of the road frames, capturing the roadway in various conditions, while panel Figure 2(b) depicts the reference lines used for alignment and analysis. Together, these images provide a comprehensive view of both the road context and the guiding lines for assessment. The VS13 dataset was recorded on a 622-meter stretch of road situated away from the main city thoroughfare.

Each recording features a single vehicle at a time, with over 10 different vehicles, varying in model and specifications, utilized throughout the dataset collection. The recordings were captured using a GoPro Hero5 camera mounted on a tripod at a distance of 0.5 meters from the roadside. The videos are encoded in MP4 format and recorded at 30 frames per second (fps) while the vehicle is in cruise control mode to ensure consistent speed measurements.

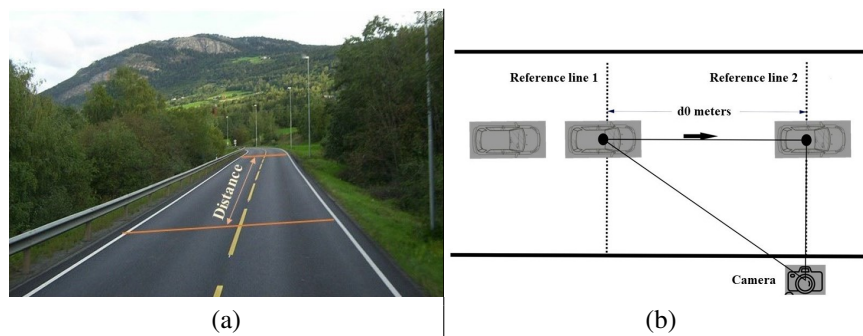


Figure 2. Illustrates two key elements; (a) screenshots of the road frames capturing different road conditions and (b) reference lines used for alignment and analysis

5. RESULTS AND DISCUSSION

We evaluated the proposed model on a computer setup with an 8 GB RAM, an Intel Core i5 processor (2.50 GHz), in a 64-bit Windows environment. The results of calculating the speed of a vehicle passing between the reference line by the proposed model by measuring the timestamp depicted on various model of car has shown below. It is pertinent to mention here that all samples are recoded at 30 FPS. The co-ordinate of the reference lines has been manually adjusted to justify the distance on road. The approach has been on randomly selected samples of VS13 dataset and the results with screen shot presented below showing the reference line and elapsed time between frames. Below presented the measured speed and benchmark speed on 6 randomly selected model and various speed range. To depict the accuracy, we have calculated the RMSE value of each car model speed range.

Simulation results: show the vehicle being tracked passing through reference line and timestamp captured. Figure 3 presents the simulation results of vehicle monitoring using the proposed approach. The results illustrate various stages of vehicle tracking, speed measurement, and reference line crossing. The cell-image of Figures 3(a)-(h) depict vehicles crossing the entry point, and demonstrate speed measurement and exiting reference line detection. These images collectively validate the effectiveness of the proposed method in accurately tracking vehicle movement and measuring speed across designated reference lines.



Figure 3. Simulation results of vehicle monitoring: (a)-(h) speed measurement and reference line crossing using the proposed approach

6. RESULT ANALYSIS

This experiment applied the proposed YOLOv8 + DeepSORT (centroid approach) speed estimation algorithm to the VS13 dataset [33], which contains annotated audio-video snippets of vehicles traveling at speeds ranging from 30 to 105 km/h. To ensure diversity in vehicle types and road conditions, we randomly selected 30 video samples featuring various car models, such as Toyota Corolla, Honda Civic, Nissan X-Trail, Ford Mustang, Tesla Model 3, and Mercedes-Benz C-Class.

Each sample was chosen to represent different speeds, lighting conditions, and environmental factors. We assessed the algorithm's performance using key metrics like mean absolute error (MAE) to measure average deviation from ground-truth speeds, root mean square error (RMSE) to evaluate error spread, percentage accuracy to gauge how closely predicted speeds align with actual values, and the system's processing speed (FPS) to confirm real-time feasibility. This approach allowed for a thorough evaluation of the proposed solution's accuracy, robustness, and computational efficiency.

Comparison with benchmark dataset and validation: to validate the effectiveness of our proposed method, we compared the predicted speeds against the VS13 dataset's ground-truth values. The resulting performance plot showed close alignment between actual and estimated speeds, with minor deviations largely attributed to occlusions and perspective distortions—factors that DeepSORT's robust tracking helps mitigate. The choice of YOLOv8 is driven by its high-speed processing and accurate vehicle detection, while DeepSORT ensures consistent object association across frames, thereby minimizing identity switches.

The Figure 4 compares the algorithm's estimated speeds with the ground-truth values from the VS13 dataset. By plotting both sets of data, we can assess how accurately the YOLOv8 + DeepSORT (centroid approach) measures vehicle speed under various car model in VS13 datasets. This direct comparison highlights the method's capability to align closely with benchmark data across different vehicle models and speeds.

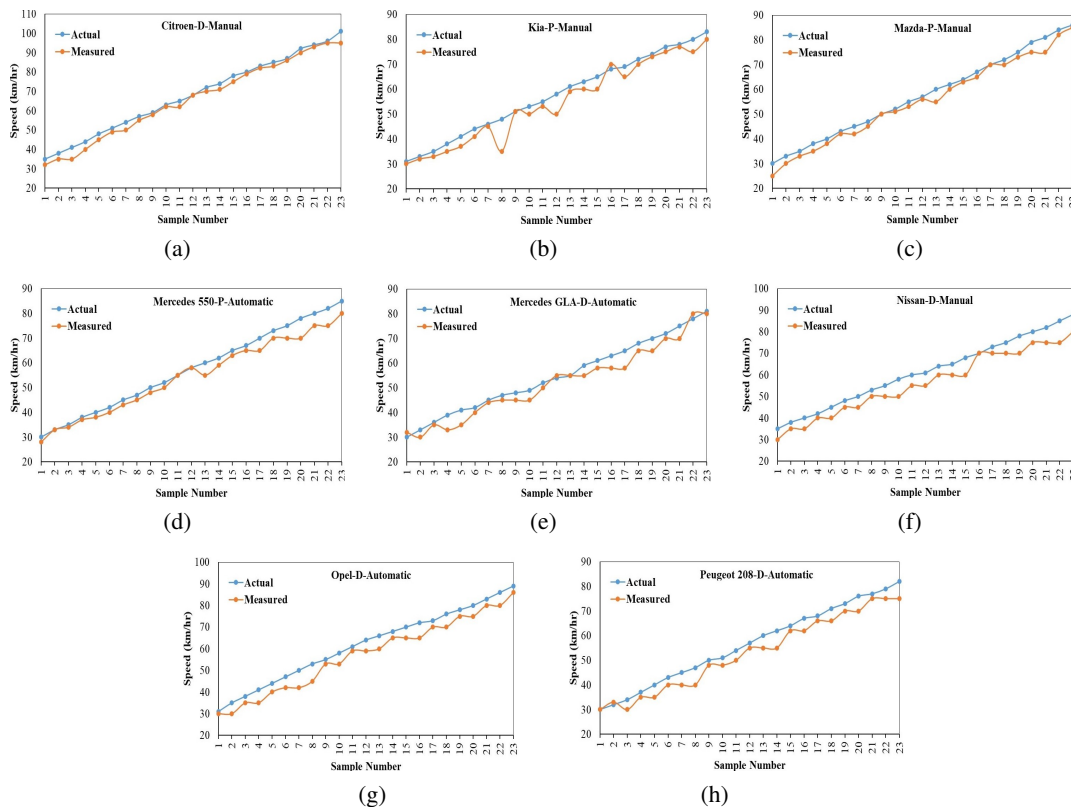


Figure 4. The charts from (a)–(h) depict calculated vs. benchmark speeds on selected vehicles from the VS13 dataset

7. DISCUSSION AND COMPARATIVE ANALYSIS

In this section, we present a comparative analysis of our proposed speed estimation model against various existing speed measurement techniques. Table 5 outlines the integration of our speed estimation model with object detection and tracking algorithms, specifically utilizing the YOLO framework. The analysis reveals that the accuracy of vehicle speed measurement in previous studies heavily relies on the precision of vehicle detection and tracking, which can be computationally intensive.

Table 5. Comparative analysis of vehicle detection & speed estimation techniques

Authors	Object detection and tracking	Speed measurement approach	Limitations	Performance range (RMSE)
Bell <i>et al.</i> [34]	YOLO v3+SORT	Pixel-to-Distance approach	Speed affected by number of detections in frame. No benchmark dataset.	1-5 km/hr
Rodríguez <i>et al.</i> [35]	YOLO v3+KBF	Measure time and coordinate, then regression model	No benchmark dataset. Sampling and selecting ideal location for dataset sampling.	1-5 km/hr
Yang <i>et al.</i> [36]	YOLO v4+SSD	Use motion of logo and light and then measure speed and distance	Needs hardware device such as industrial camera.	1-4 km/hr
Fernandez <i>et al.</i> [37]	MSER detector	Use the detection time between two detections	Needs multiple camera calibration.	1-4 km/hr
Azimjonov <i>et al.</i> [38]	YOLO v4+CNN Classifier	Motion-based method	No benchmark dataset.	1-4 km/hr
Nguyen <i>et al.</i> [39]	YOLO v4+DSORT	Pixel-to-Distance approach	No benchmark dataset.	1-6 km/hr
Wu <i>et al.</i> [40]	YOLO v5	Simulated software-based approach	Virtual environment.	1-5 km/hr
Our work	YOLO v8+C-BBox	Centroid-based tracking, measure time and distance	Calibration needed in reference lines.	2-5 km/hr

7.1. Summary of key findings

This paper presents a centroid-based tracking algorithm for vehicle speed estimation that leverages YOLOv8 (an advanced object detection architecture) integrated with DeepSORT for robust multi-object tracking. Evaluations on the VS13 dataset—which includes diverse car models traveling at speeds ranging from 30 to 105 km/h—demonstrated that the proposed method improves accuracy by 2–5% compared to existing techniques. With a consistent frame rate of around 30 FPS and minimal computational overhead, the approach effectively balances real-time processing and high detection performance. Our findings indicate that our speed estimation model significantly outperforms others when evaluated on a benchmark dataset. Moreover, our algorithm demonstrates superior performance in real-time video samples, particularly in constrained environments, compared to some existing models.

7.2. Context within previous studies

Prior research has highlighted the efficacy of YOLO-based frameworks for real-time detection, while tracking algorithms like SORT and DeepSORT address identity preservation and occlusion issues. Despite these advancements, many methods struggle with handling complex traffic environments or maintaining accuracy under occlusions. By incorporating YOLOv8 (for improved detection accuracy) and a centroid-based approach (for simplified velocity calculation), this study builds on earlier works that employed older YOLO versions or less efficient trackers. The combination successfully reduces ID switches and accommodates varied environmental conditions, marking a step forward in vehicle speed measurement. Previous studies, such as those by Bell *et al.* [34], [35] developed custom datasets for speed assessment, achieving an accuracy of 1–5 km/h, while Fernandez *et al.* [37] applied camera calibration techniques for speed measurement, reporting similar accuracy levels. Unlike these approaches, our method integrates YOLOv8 with centroid-based tracking, requiring minimal calibration and improving real-time speed estimation accuracy to 2–5 km/h, demonstrating adaptability in both constrained and open-road environments.

7.3. Future research and key take-away

Future work could involve broadening the dataset to include additional benchmark sources featuring extreme weather, nighttime scenarios, and high-density traffic. Sensor fusion—such as combining LiDAR or radar with video-based methods—may further enhance reliability under adverse conditions. Moreover, multi-camera coordination using advanced synchronization and multi-view geometry could refine speed accuracy and manage occlusion challenges. Overall, by merging YOLOv8's precise object detection with DeepSORT's robust tracking, this research offers a practical, high-accuracy framework for real-time vehicle speed estimation, underscoring how modern computer vision can advance intelligent transportation and road safety initiatives.

8. CONCLUSION

In this study, we proposed a centroid-based algorithm for vehicle speed estimation that integrates the YOLOv8 object detector with DeepSORT centroid based tracking. Evaluations were performed on the benchmark VS13 dataset, demonstrating 2–5% higher measurement accuracy compared to existing approaches. Notably, the method achieves real-time performance at 30 FPS and maintains a low computational footprint, making it viable for resource-constrained environments. Our results indicate that larger bounding boxes enhance tracking stability, and the approach remains robust across varying vehicle speeds and models. Furthermore, the algorithm exhibits a lower RMSE than other state-of-the-art methods, underscoring its suitability for intelligent transportation systems (ITS). Future research should aim to establish comprehensive benchmark datasets and explore advanced sensor integration (e.g., LiDAR or radar) to further refine speed estimation accuracy and extend applicability to a broader range of traffic scenarios.

ACKNOWLEDGMENTS

The authors would like to acknowledge the Department of CSE, United University, Prayagraj, for providing the necessary laboratory facilities and resources that contributed to the successful completion of this research.

FUNDING INFORMATION

Authors state no funding involved.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Pankaj Kumar Gautam	✓	✓	✓	✓	✓	✓		✓	✓	✓			✓	
Sanjeev Kumar		✓				✓		✓	✓	✓	✓	✓		

C : Conceptualization	I : Investigation	Vi : Visualization
M : Methodology	R : Resources	Su : Supervision
So : Software	D : Data Curation	P : Project administration
Va : Validation	O : Writing - Original Draft	Fu : Funding acquisition
Fo : Formal analysis	E : Writing - Review & Editing	

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

Data availability is not applicable to this paper as no new data were created or analyzed in this study.

REFERENCES




- [1] H. Liu and R. R. Shetty, "Analytical Models for Traffic Congestion and Accident Analysis," 2021.
- [2] K. A. R. L. Jakus and D. S. Coe, "Speed measurement through analysis of the Doppler effect in vehicular noise," *IEEE Transactions on Vehicular Technology*, vol. 24, no. 3, pp. 33-38, 1975, doi: 10.1109/T-VT.1975.23865.
- [3] Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 11, pp. 3212-3232, 2019, doi: 10.1109/TNNLS.2018.2876865.
- [4] J. Han, Y. Liao, J. Zhang, S. Wang, and S. Li, "Target fusion detection of LiDAR and camera based on the improved YOLO algorithm," *Mathematics*, vol. 6, no. 10, p. 213, 2018, doi: 10.3390/math6100213.
- [5] J.-P. Lin and M.-T. Sun, "A YOLO-based traffic counting system," in *Proc. 2018 Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, 2018, pp. 82-85, doi: 10.1109/TAAI.2018.00023.
- [6] C.-J. Lin, S.-Y. Jeng, and H.-W. Liao, "A Real-Time Vehicle Counting, Speed Estimation, and Classification System Based on Virtual Detection Zone and YOLO," *Mathematical Problems in Engineering*, 2021. Available: <https://doi.org/10.1155/2021/6631765>.
- [7] Z. Chen, L. Cao, and Q. Wang, "YOLOv5-Based Vehicle Detection Method for High-Resolution UAV Images," *Mobile Information Systems*, 2022, doi: 10.1155/2022/9533747.

- [8] G. R. Arash, A. D. Abbas, and M. R. K., "Vehicle Speed Detection in Video Image Sequences Using CVS Method," *International Journal of Physical Sciences*, vol. 5, no. 17, pp. 2555-2563, 2010.
- [9] A. H. Rais and R. Munir, "Vehicle Speed Estimation Using YOLO, Kalman Filter, and Frame Sampling," in *Proc. 8th International Conference on Advanced Informatics: Concepts, Theory and Applications (ICAICTA)*, 2021, pp. 1-6, doi: 10.1109/ICAICTA53211.2021.9640273.
- [10] J. Azimjonov and A. Özmen, "A Real-Time Vehicle Detection and a Novel Vehicle Tracking System for Estimating and Monitoring Traffic Flow on Highways," *Advanced Engineering Informatics*, vol. 50, p. 101393, 2021, doi: 10.1016/j.aei.2021.101393.
- [11] T. Rahman, M. A. L. Siregar, A. Kurniawan, S. Juniastuti, and E. M. Yuniarno, "Vehicle Speed Calculation from Drone Video Based on Deep Learning," in *Proc. 2020 International Conference on Computer Engineering, Network, and Intelligent Multimedia (CENIM)*, 2020, pp. 229-233, doi: 10.1109/CENIM48368.2020.9272685.
- [12] P. H. Nguyen and M. B. Duy, "An Algorithm Using YOLOv4 and DeepSORT for Tracking Vehicle Speed on Highway," *Indonesian Journal of Electrical Engineering and Informatics (IJEI)*, vol. 10, no. 1, pp. 90-101, 2022, doi: 10.11591/ije.i.v10i1.1284.
- [13] H. Dong, M. Wen, and Z. Yang, "Vehicle Speed Estimation Based on 3D ConvNets and Non-Local Blocks," *Future Internet*, vol. 11, no. 6, p. 123, 2019, doi: 10.3390/fi11060123.
- [14] T.-H. Wu, T.-W. Wang, and Y.-Q. Liu, "Real-Time Vehicle and Distance Detection Based on Improved YOLO v5 Network," in *Proc. 2021 3rd World Symposium on Artificial Intelligence (WSAI)*, 2021, pp. 24-28, doi: 10.1109/WSAI52458.2021.9473675.
- [15] P. Hurtik, V. Molek, J. Hula, M. Vajgl, P. Vlasanek, and T. Nejezchleba, "Poly-YOLO: Higher Speed, More Precise Detection and Instance Segmentation for YOLOv3," *Neural Computing and Applications*, vol. 34, no. 10, pp. 8275-8290, 2022, doi: 10.1007/s00521-021-06228-2.
- [16] U. Mittal, P. Chawla, and R. Tiwari, "EnsembleNet: A Hybrid Approach for Vehicle Detection and Estimation of Traffic Density Based on Faster R-CNN and YOLO Models," *Neural Computing and Applications*, 2022, pp. 1-20, doi: 10.1007/s00521-022-07693-w.
- [17] Z. Tang, G. Wang, H. Xiao, A. Zheng, and J.-N. Hwang, "Single-Camera and Inter-Camera Vehicle Tracking and 3D Speed Estimation Based on Fusion of Visual and Semantic Features," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2018, pp. 108-115, doi: 10.1109/CVPRW.2018.00022.
- [18] P. Giannakeris, V. Kaltsa, K. Avgerinakis, A. Briassoulis, S. Vrochidis, and I. Kompatsiaris, "Speed Estimation and Abnormality Detection from Surveillance Cameras," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2018, pp. 93-99, doi: 10.1109/CVPRW.2018.00017.
- [19] S. Hua, M. Kapoor, and D. C. Anastasiu, "Vehicle Tracking and Speed Estimation from Traffic Videos," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2018, pp. 153-160, doi: 10.1109/CVPRW.2018.00028.
- [20] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779-788, doi: 10.1109/CVPR.2016.91.
- [21] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 7263-7271, doi: 10.1109/CVPR.2017.690.
- [22] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple Online and Realtime Tracker," in *Proc. 2016 IEEE International Conference on Image Processing (ICIP)*, 2016, pp. 3464-3468, doi: 10.1109/ICIP.2016.7533003.
- [23] N. Wojke, A. Bewley, and D. Paulus, "Simple Online and Realtime Tracking with a Deep Association Metric," in *Proc. 2017 IEEE International Conference on Image Processing (ICIP)*, 2017, pp. 3645-3649, doi: 10.1109/ICIP.2017.8296962.
- [24] Y. Du, Z. Zhao, Y. Song, Y. Zhao, F. Su, T. Gong, and H. Meng, "StrongSORT: Make DeepSORT Great Again," *arXiv preprint arXiv:2202.13514*, 2022.
- [25] Y. Zhang, P. Sun, Y. Jiang, D. Yu, Z. Yuan, P. Luo, and W. Liu, "ByteTrack: Multi-Object Tracking by Associating Every Detection Box," *arXiv preprint arXiv:2110.06864*, 2021.
- [26] M. Naphade et al., "The 7th AI City Challenge," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 5538-5548, 2023.
- [27] J. Sochor, R. Juránek, and A. Herout, "BrnoCompSpeed: Review of Traffic Camera Calibration and Comprehensive Dataset for Monocular Speed Measurement," *arXiv preprint arXiv:1702.06441*, 2017.
- [28] V. Kocur and M. Ftáčnik, "Detection of 3D Bounding Boxes of Vehicles Using Perspective Transformation for Accurate Speed Measurement," *Machine Vision and Applications*, vol. 31, no. 7, p. 62, 2020.
- [29] A. J. Jakub, D. Bařina, and P. Zemřík, "BrnoCompSpeed: Video dataset for vehicle speed measurement," in *Proc. 14th IEEE International Conference on Advanced Video and Signal-Based Surveillance (AVSS)*, 2017, pp. 1-6.
- [30] F. de A. P. Macêdo, A. H. B. de Gusmão, and R. C. S. de Oliveira, "Traffic analysis using the UTFPR dataset," in *Proc. 2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2018, pp. 1-6.
- [31] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009, pp. 248-255.
- [32] A. Khoreva, R. Benenson, M. Omran, J. Hosang, and B. Schiele, "Lucid data dreaming for object tracking," in *Proc. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 3053-3061.
- [33] M. G. Xu, J. Li, and T. D. Ngo, "VS13: A dataset for vehicle speed estimation using audio and video," in *Proc. 2013 IEEE International Conference on Multimedia and Expo (ICME)*, 2013, pp. 1-6.
- [34] D. Bell, W. Xiao, and P. James, "Accurate Vehicle Speed Estimation from monocular Camera Footage," in *Proc. XXIV ISPRS Congress*, Newcastle University, 2020, doi: 10.5194/isprs-archives-XLIII-B2-2020-41-2020.
- [35] H. Rodríguez-Rangel, L. A. Morales-Rosales, R. Imperial-Rojo, M. A. Roman-Garay, G. E. Peralta-Peñuñuri, and M. Lobato-Báez, "Analysis of Statistical and Artificial Intelligence Algorithms for Real-Time Speed Estimation Based on Vehicle Detection with YOLO," *Applied Sciences*, vol. 12, no. 6, p. 2907, 2022, doi: 10.3390/app12062907.
- [36] L. Yang, J. Luo, X. Song, M. Li, P. Wen, and Z. Xiong, "Robust Vehicle Speed Measurement Based on Feature Information Fusion for Vehicle Multi-Characteristic Detection," *Entropy*, vol. 23, no. 7, p. 910, 2021, doi: 10.3390/e23070910.
- [37] D. F. Llorca, A. H. Martinez, and I. G. Daza, "Vision-Based Vehicle Speed Estimation: A Survey," *IET Intelligent Transport Systems*, vol. 15, no. 8, pp. 987-1005, 2021, doi: 10.1049/itr2.12046.




- [38] J. Azimjonov and A. Özmen, "A Real-Time Vehicle Detection and a Novel Vehicle Tracking System for Estimating and Monitoring Traffic Flow on Highways," *Advanced Engineering Informatics*, vol. 50, p. 101393, 2021, doi: 10.1016/j.aei.2021.101393.
- [39] P. H. Nguyen and M. B. Duy, "An Algorithm Using YOLOv4 and DeepSORT for Tracking Vehicle Speed on Highway," *Indonesian Journal of Electrical Engineering and Informatics (IJEI)*, vol. 10, no. 1, pp. 90–101, 2022, doi: 10.11591/ijeiei.v10i1.1284.
- [40] T.-H. Wu, T.-W. Wang, and Y.-Q. Liu, "Real-Time Vehicle and Distance Detection Based on Improved YOLO v5 Network," in *Proc. 2021 3rd World Symposium on Artificial Intelligence (WSAI)*, IEEE, 2021, pp. 24–28, doi: 10.1109/WSAI52458.2021.9473675.

BIOGRAPHIES OF AUTHORS



Pankaj Kumar Gautam    is a researcher and educator in the field of computer science and engineering. His research interests include computer vision, image analysis, and object tracking. He received a B.Tech. degree in Information Technology from Kamla Nehru Institute of Technology (KNIT), Sultanpur, and an M.Tech. degree from Sardar Vallabhbhai National Institute of Technology (SVNIT), Surat. He is currently pursuing a Ph.D. in Computer Science and Engineering at United University, Jhalwa, Prayagraj, India. With 11.5 years of teaching experience, including 8.5 years in the Board of Technical Education (BTE), he specializes in Java, C, and Information Security. He has published research papers in international conferences and journals and actively participates in faculty development programs. He can be contacted at pankajgautam257@gmail.com.



Sanjeev Kumar    is an Associate Professor in the Department of Computer Science and Engineering at United University, Jhalwa Prayagraj, India. He holds a Ph.D. in Computer Engineering from Motilal Nehru National Institute of Technology (MNNIT Allahabad) with a specialization in Computer Vision. His research areas include computer vision, object tracking, image processing, biometrics, medical image analysis, and pattern recognition. With 9 years of teaching experience, he specializes in Java, C, and Information Security. He has published research in international conferences and journals and actively participates in faculty development programs. He can be contacted at sanjeev.kumar@uniteduniversity.edu.in.