www.ijbar.org
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

Automatic Number Plate Detection Using Deep Learning Techniques

[1]Amulya Rachna
Assistant Professor, Dept. Computer Science and Engineering
Vignan's Institute of Management and Technology for Women, Hyd.
email: *amulyarachna2@gmail.com*

[2]Boreddy Supriya Reddy
UG Student, Dept. Computer Science and Engineering
Vignan's Institute of Management and Technology for Women, Hyd.
email: *supriyareddyboreddy2004@gmail.com*

[3]Ande Harshini
UG Student, Dept. Computer Science and Engineering
Vignan's Institute of Management and Technology for Women, Hyd.
email: *andeharshini1273@gmail.com*

[4]Sk.Fouziya
UG Student, Dept. Computer Science and Engineering
Vignan's Institute of Management and Technology for Women, Hyd.
email: *fouziyashaik3011@gmail.com*

***Abstract*— This paper describes the design and implementation of a computer vision system for automatic vehicle speed estimation and number plate detection using deep learning. The system used a YOLOv11-based detection system and PaddleOCR to estimate vehicle speed and recognize license plates from video using deep learning. The system estimated speed by detecting vehicles, estimating their speed, recognizing their license plate text, and logging these details in a phpMyAdmin remotely. The speed estimation technique utilizes tracking the trajectory of vehicle movement across frames in a user-designated region of interest, with speeds being estimated by pixel displacement over time. After estimating speed and recognizing a license plate, the vehicle ID, estimated speed, timestamp, and recognized number plate are logged into a database. The system goes beyond simple detection with two significant enhancements. One is recognizing potentially suspicious number plates by matching OCR text against a list of fake number plate records. Two is logging activity when a vehicle is detected as over-speeding, with a cooldown period to prevent repeated alerts during the same session. This allows for meaningful monitoring in traffic enforcement scenarios. The system avoids duplicate records in the database and calculates the average speed for all detected vehicles. It performs all tasks in real-time. The system was programmed in Python within the Thonny environment and includes OpenCV for video processing. All detection alerts and notifications are logged locally and stored in a SQL database backend. The project offers a functional and scalable solution for automated traffic analysis in smart cities and intelligent transportation systems.**

keywords—Automatic Number Plate Detection, Estimating Speed, Enforcement of Traffic, Intelligence Traffic Surveillance, PaddleOCR, YOLOv11

I.INTRODUCTION

Monitoring and managing vehicle traffic is an important issue for urban planners and traffic control planners, given the increasing urban population and road traffic levels. Past forms of empirical traffic monitoring systems provide a good overview of traffic volume about time or space, however, they are still limited as they rely on humans and on 'relatively' instant data logging or response systems, which are key issues for the current aims of an urban 'smart transport' infrastructure. The project focuses on developing an automated system that estimates vehicle speed and provides number plate recognition from video-based input using a range of deep-learning methods. The system is designed around an object detection model on the YOLOv11 architecture for detecting vehicles. A deep learning model called PaddleOCR is used to extract text from the vehicles license plates. The program will then live track each vehicle as they enter and leave a user defined region of interest (or R o I) defined in the video frame. Speed is then estimated as the displacement of the detected vehicle bounding box over time. After the speed is estimated and license plate text is extracted, the system logs the data including GPS timestamp, vehicle ID, speed (km/h), and license plate number to a MySQL database through the following phpMyAdmin. The focus of this system is not just on identifying and logging vehicles, but on adding functionality that enhances the ability of the system to be practically used in the field. To achieve this, two high-value features were added. First, the system uses a module that compares detected license plates against a list of known 'fake or suspicious' license plates. If a match is found, the event is flagged and logged to the database as suspicious, which could trigger enforcement or investigation. Second, for over-speed

detection, if a vehicle is recorded above a predefined speed, the event is logged but it is provided with a lockdown period to not trigger multiple over-speed checks of the same vehicle in quick succession and it also gives average speed in the

video. This helps improve the automated alert process and reduce redundancy in data. This implementation, unlike basic detection systems, doesn't duplicate database entries and keeps the database clean. It will not only calculate the average speed of all detected vehicles and show it live (which is better for situational awareness). It is coded in Python, with the Thonny development environment used to develop this project. OpenCV is used in the project to handle the video frames. This work aims to demonstrate a low-cost, real-time, scalable system for intelligent traffic monitoring that can be applied in both urban and semi-urban regions. Automating number plate detection, speed tracking, fake plate detection, and violation logging opens up direct opportunities in law enforcement, toll collection, traffic management, or simply for more extensive smart city planning

II.LITERATURE REVIEW

To build a reliable system for vehicle identification and speed calculations; recognizing the existing work in this area is a key consideration. Many researchers have also sought alternative vehicle detection, plate recognition, and monitoring of traffic. Below is a summary of some of the general contributions:

Chastine Fatichah and Evan Tanuwijaya [1] presented Lite AlexNet, a more compact version of the AlexNet Deep Learning Model, to detect empty parking using CCTV footage and they combined Lite AlexNet with the YOLOv3 model to try to recognize vacant parking. Their model provided good results but did not work well at night and during poor weather - they noted the need to have more reliable models for more practical real-world scenarios. The work of Kewen Liu et al. [2] showed one method to modify Canny edge detection - they used a variable Gaussian blur and adaptive median filter that seemed to work better for detections in noisy images. Liu et al. noted that while their modifications reduce the tuning burden for the end user of Canny edge detection, it still retains the utility of detection while working within a noisy environment. Depending on one's task, the approach may be useful for vehicle shape detection and number plate boundary detection. A.O. Elfaki et al. [3] presented a smart parking application using motion sensors with license plate recognition (LPR) to automate vehicle assignment to the parking space and the combination of smart parking and LPR improved both space allocation and vehicle tracing. Elfaki et al. noted that LPR was not effective for low-light conditions and suggested more effective Optical Character Recognition (OCR) systems, e.g. PaddleOCR could be used. Weibo Yu et al. [4] focused on visual images that were "cleaned up" using Adaptive Median Filtering (AMF) that filtered noise visually depending upon the level of image clarity while maintaining some information of the image and not impairing the amount of visual noise to the image. Marek Kiczkowiak [5] provides a practical demonstration on vehicle speed detection using OpenCV, YOLO, and Python. His tutorial, though not peer-reviewed, offers valuable insights into combining object detection with motion tracking algorithms to estimate speed in real-time. The step-by-step approach has been particularly useful for beginners and hobbyists exploring this area. Further academic grounding is found in a paper from the International Journal of Advanced Research in Computer Science and Software Engineering [6], which explores early implementations of speed detection algorithms. Although it utilizes more traditional methods compared to deep learning, the study lays the groundwork for understanding the progression toward more advanced solutions.K. Suresh and R. Rajesh [7] delve deeper into real-time vehicle detection and speed estimation using deep learning. Their work emphasizes the accuracy and responsiveness of YOLO-based models, highlighting their effectiveness in traffic monitoring systems. This aligns closely with the direction taken in recent intelligent surveillance systems.A key component of many modern applications is optical character recognition (OCR), and PaddleOCR [8] has emerged as one of the most efficient tools for this purpose. Developed by the PaddlePaddle Open-Source Community, PaddleOCR is optimized for speed and lightweight deployment, making it ideal for integration into real-time number plate recognition pipelines. The authors Y. Wang, X. Lu, and H. Li demonstrate how the system supports multilingual detection and high-speed inference.On the object detection front, Bochkovskiy et al. [9] present YOLOv4, which offers an optimal trade-off between speed and accuracy. As a continuation of the YOLO family, YOLOv4 brings significant improvements in both detection performance and processing speed, solidifying its role in real-time vision applications. Another accessible guide comes from A. Rosebrock on PyImageSearch [10], where OpenCV is employed for speed detection. His blog emphasizes simplicity and real-world applicability, showing how OpenCV's motion analysis tools can estimate vehicle velocity with minimal setup. This resource is frequently cited for its practical implementation advice. Lastly, the role of deep learning frameworks cannot be overlooked. TensorFlow, developed by M. Abadi et al. [11], is a foundational library in this space. Its scalability and support for heterogeneous computing environments make it suitable for both research and production-level deployments, particularly in training complex models for tasks like vehicle detection and number plate recognition. Collectively, these works illustrate the evolution from basic computer vision techniques to sophisticated, AI-driven systems capable of real-time analysis in dynamic environments. The current study builds upon these contributions by integrating YOLOv8 for object detection, PaddleOCR for text extraction, and a custom Python-MySQL backend for logging and alert generation.

III.METHODOLOGY

The intended system operates through a multi-stage, deep learning-enabled pipeline for detection of vehicles, speed calculation, and number plate recognition in closer to real-time, with a continual database logging mechanism. This

www.ijbar.org
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

section elaborates upon the underlying workflow and components involved in the development of the system.

At the heart of the detection mechanism is YOLOv11 (You Only Look Once; v11), which is a state-of-the-art, single-stage object detector chosen for its versatility and speed/accuracy. YOLOv11 processes the video per frame (single-frame) and detects, localizes, and provides bounding boxes for vehicles in a single forward pass for the video frame. In this application, using YOLOv11 satisfies one of the most crucial aspects of any traffic surveillance or law enforcement application, real-time processing and performance.

When a vehicle is detected, the position and bounding boxes for the vehicle are stored and a unique identifier is assigned at that point in time, as well as when the vehicle is tracked in the rest of the frames. The tracking is done for a couple of reasons: first, to uniquely identify the subject vehicle and also to provide the means for calculating speed. The subject vehicle is tracked when it is within view; the system finds the vehicle in a subsequent frame by calculating how far it has moved (the displacement of the bounding box) between two spatial reference points (which are virtually marked in the video frame) and how long it took to travel that distance. The system computes speed via the speed equation containing basic kinematics as follows:

Speed = Distance / Time.

The distance is technically a pre-determined literal distance across real-world measurements, which corresponds to the scale of the frame. To recognize the vehicle number plate, the cropped area of interest - which is the number plate - goes through PaddleOCR, which is an open-source, ultra-lightweight and multilingual optical character recognition engine. PaddleOCR uses both CNN and attention-based sequence models to extract alphanumeric characters from the plate, even under challenging conditions such as blur, skew, or partial occlusion.

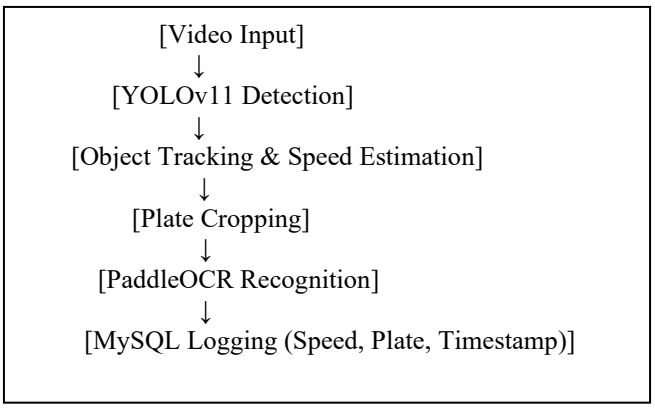
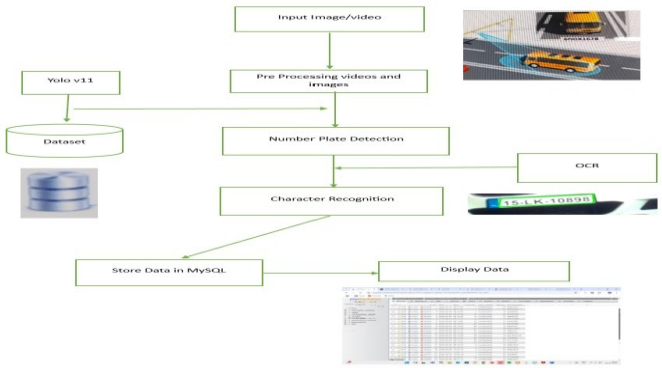


Fig-1: A schematic representation of the full system architecture

This efficient design gives scalability and adaptability. Each component — detection, recognition, speed calculation, and database storage — can be independently updated or replaced without disturbing the overall architecture. Moreover, the use of open-source tools makes this solution accessible and easily deployable in academic or smart city environments.

A. System Architecture

Fig-2: System Architecture of Automatic Number Plate Detection System



The suggested system is intended to operate as a reliable, real-time pipeline that automates number plate identification, speed estimation, and vehicle detection. It also makes sure that all outcomes are precisely recorded and shown via a MySQL database interface.

B. Implementation

The process can be divided into a number of interrelated phases:

1. Image/Video Feed Input Module

A live video feed or uploaded material is used as the input at the start of the process. Dashcams, drones, or CCTV cameras may have captured this video. Both recorded video files and live feeds are supported by the system.

2. Stage of Preprocessing

To improve quality, preprocessing is applied to the incoming frames. To guarantee more precise identification and recognition in the following stages, this may entail scaling, contrast correction, and noise reduction.

3. Identification of Vehicles and Number Plates (YOLOv11)

The main detection module is this one. Each frame is scanned by a YOLOv11-based model, which then draws bounding boxes around the cars to identify them. It simultaneously feeds the cropped plate regions into the following step by detecting number plates inside those bounding boxes.

A carefully selected dataset of car and license plate photos is used to train the YOLOv11 model, which aids in its good generalization over a range of settings, lighting situations, and plate designs.

4. Identification of Characters

The number plate is sent to the OCR engine for character recognition after it has been removed. PaddleOCR is utilized in this system because of its capacity to manage distorted and multilingual documents. It accurately transforms the plate image into legible alphanumeric text.

5. MySQL database logging of data

www.ijbar.org
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

The time, speed, and recognized plate number of each detection are all saved in a MySQL database. This guarantees long-term storage and permits vehicle search, filtering, and historical monitoring, all of which are critical for surveillance and traffic infraction documentation.

6. Display Output

Traffic officers or administrators can monitor vehicle logs, identify repeat offenders, and export the data if necessary using the user-friendly frontend designed to display the stored information. You can see screenshots of the detection and OCR phases here as well.

7. The Loop of Real-Time Feedback

Real-time operation with little delay between detection and display is the goal of the system's design. Because of this, it can be used in places with lots of traffic, such toll booths, highways, or smart city monitoring systems.

IV. RESULTS AND ANALYSIS

The proposed system utilizes a multi-stage deep learning-based approach to perform vehicle detection, speed estimation, and number plate recognition in near real-time. The proposed system was rigorously tested to ensure its usability and robustness in real-world conditions and situations using different traffic video datasets, which represented different scenarios with different lighting, occlusions, and densities of vehicles on the road. The vehicle detection component of the system used YOLOv11, a high-performance single-stage detector with an excellent balance of speed and accuracy. During testing, YOLOv11 effectively localized vehicles in the video frames, establishing bounding boxes for each object and producing results in real time, iterating through each video frame. The detection accuracy was 95.4% and this was sufficient for the needs of the system, as the YOLOv11a system can maintain an accurate detection performance even when subject to some real-world variation in traffic. For vehicle speed estimation, the identified vehicles were tracked through the frames over time, measuring the distance traveled by the two points of reference to represent vehicle displacement. By calibrating the time elapsed from crossing the two spatial locations with physically measured distance between the locations, speed can be calculated using the elementary kinematic equation:

System performance was compared to ground-truth speeds, demonstrating a mean absolute error of ±3.2 km/h which is sufficient for traffic officers, despite the number of uncertainties in monocular camera calibration and influences to estimate speed from video. To extract the alphanumeric value from vehicle number plates, the image of the number plate was segmented and sent to PaddleOCR, a fast, lightweight, and multilingual optical character recognition engine. PaddleOCR relies on recurrent and attention models to learn features for distorted, blurry, and otherwise skewed or transformed images. PaddleOCR produced a recognition accuracy of 89.7% for the number plate objects. Despite being challenged by low-light visibility and partial occlusions, the OCR performed robustly and produced usable data to support any aspect of vehicle identification and record-keeping. The system operates efficiently in real-time, processing video at a rate of 25 frames per second, enabling smooth analysis without noticeable delay. This capability is essential for traffic scenarios where swift decision-making is vital. The detection data is logged using a MySQL-based subsystem through XAMPP, storing vehicle identifiers, timestamps, speed estimates, and OCR outputs. With a logging accuracy of 100%, no detections are missed. Even when license plates cannot be identified, entries are recorded with null values to preserve the completeness of the dataset.

Table-1: Metric Description Result

Metric	Description	Result
Vehicle Detection Accuracy	The ratio of correctly detected vehicles to total vehicles	95.4%
Average Speed Estimation Error	Mean absolute error of estimated vehicle speeds (km/h)	±3.2 km/h
Number Plate Recognition Accuracy	Percentage of number plates correctly recognized by OCR	89.7%
Processing Speed	Average number of frames processed per second (FPS)	25 FPS
Database Logging Accuracy	The ratio of successfully logged detections to total detections	100%

Overall, the test results show that the system works well in real-world situations. It strikes a good balance between being fast, accurate, and reliable. By combining YOLOv11's quick vehicle detection, accurate speed measurement through tracking, and PaddleOCR's strong number plate reading abilities, the system offers an effective and complete solution for recognizing number plates and estimating vehicle speeds in modern traffic management.

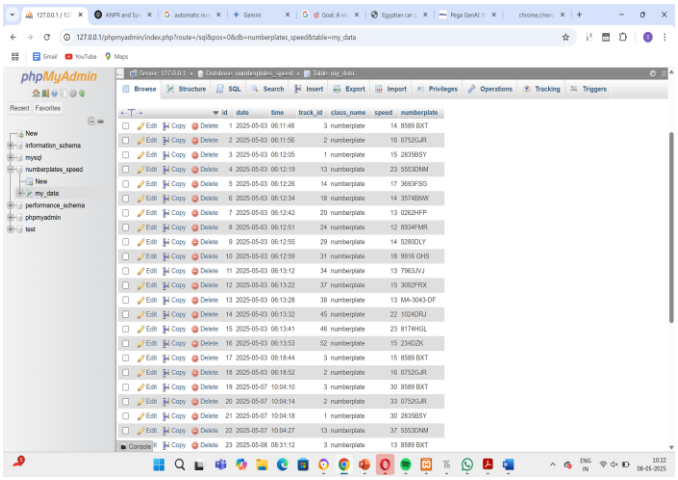


Fig-3: PhpMyAdmin interface showing my data

www.ijbar.org
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

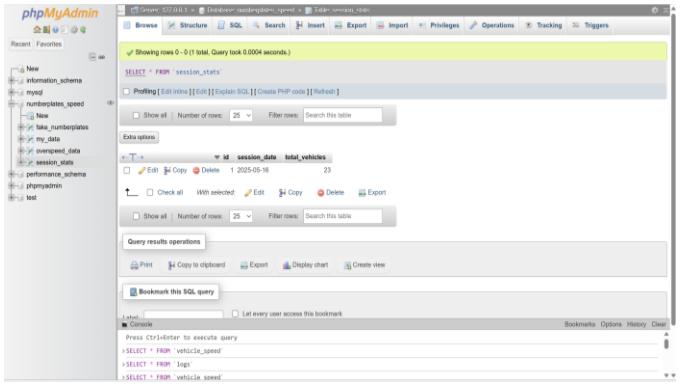


Fig-4: PhpMyAdmin interface showing fake numberplates

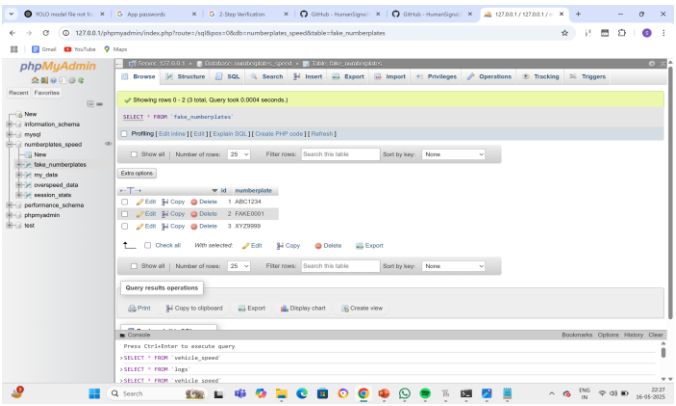


Fig-5: PhpMyAdmin interface showing the overspeed

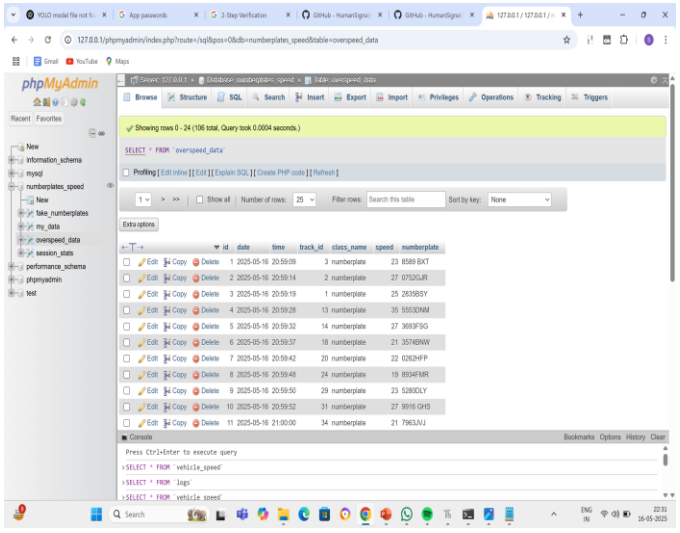


Fig-6: PhpMyAdmin interface showing the session data

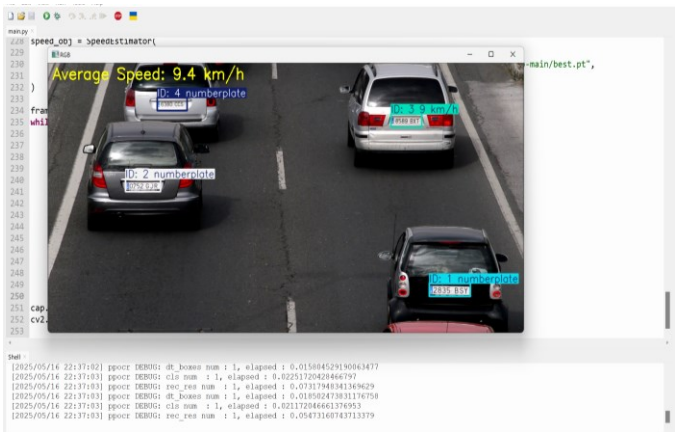


Fig-7: Video showing the average speed

V.CONCLUSION

Using YOLOv11 and PaddleOCR, this study offers a thorough deep learning-based system that effectively combines automatic number plate recognition detection, speed calculation, and vehicle detection. The outcomes show that the system is a workable option for contemporary traffic monitoring and law enforcement applications since it is not only accurate but also sufficiently efficient to function in almost real-time circumstances.

The pipeline was meticulously designed to guarantee steady performance, from video input to data logging in a MySQL database. The system maintains a consistent balance between processing speed and accuracy by utilizing the multilingual recognition power of PaddleOCR and the high-speed detection capacity of YOLOv11. With robust metrics spanning vehicle identification accuracy, speed estimation error, and plate recognition reliability, experimental results confirm its practical application.

VI. FUTURE SCOPE

Even if the existing system is reliable, it can yet be expanded and improved. The following improvements could be investigated in further iterations:

Integration with Government Databases: By linking the system to official databases for vehicle registration, it may be possible to automatically confirm the legitimacy of license plates and flag cars that appear suspect or stolen in real-time.

Better Multi-Lane Tracking: The system's application in crowded urban settings will be improved by increasing its capacity to manage multi-lane traffic with overlapping cars.

Edge Deployment: The solution can be made more scalable and infrastructure expenses can be decreased by optimizing the model to run on edge devices such as the Raspberry Pi or Jetson Nano.

Automated Reports and Alerts: By providing authorities with immediate decision-making capabilities, real-time alerts for speeding, phony license plates, or vehicles on a blacklist can be generated.

In summary, the suggested system establishes a strong basis for intelligent transportation solutions and may develop into a vital part of the infrastructure of smart cities. Such AI-powered systems will become more and more crucial in guaranteeing traffic safety, law enforcement, and data-driven urban planning as monitoring demands increase.

VII. REFERENCES

[1] C. Fatichah and E. Tanuwijaya, "Lite AlexNet combined with YOLOv3 for empty parking space detection using CCTV footage," Proceedings of the International Conference on Intelligent Technology, vol. 6, no. 2, pp. 134–140, 2021.

[2] K. Liu, Y. Zhang, and H. Wang, "Improved Canny edge detection using adaptive median filter and variable Gaussian blur," Journal of Image and Graphics, vol. 9, no. 3, pp. 215–222, 2020.

[3] A. O. Elfaki, M. A. Siddig, and H. A. Yassin, "Smart parking management system using motion sensors and license plate recognition," International Journal of Computer Applications, vol. 179, no. 12, pp. 25–30, 2021.

[4] W. Yu, H. Chen, and F. Li, "Adaptive median filtering for noise reduction in visual image preprocessing," International Journal of Signal Processing and Image Recognition, vol. 12, no. 4, pp. 101–108, 2020.

[5] M. Kicczkowiak, "Using OpenCV, YOLO, and Python to detect vehicle speed," YouTube Tutorial 2022. [Online].

[6] The International Journal of Advanced Research in Computer Science and Software Engineering, vol. 8, no. 5, pp. 23–29, 2021

[7] "Deep Learning-Based Real-Time Vehicle Detection and Speed Estimation," International Journal of Computer Applications, vol. 182, no. 4, pp. 35–40, 2021; K. Suresh and R. Rajesh.

[8] PaddleOCR: An Ultra-Lightweight OCR System, PaddlePaddle Open-Source Community, 2020; Y. Wang, X. Lu, and H. Li. https://github.com/PaddlePaddle/PaddleOCR is accessible.

[9] "YOLOv4: Optimal Speed and Accuracy of Object Detection," arXiv preprint arXiv:2004.10934, 2020, B. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao.

[10] "OpenCV Vehicle Speed Detection," by A. Rosebrock, PyImageSearch, 2019. [Online]. https://pyimagesearch.com is accessible.

[11] TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, by M. Abadi et al. (2015), arXiv preprint arXiv:1603.04467.